



# TIME TO WIN

all updates are on <http://www.geocities.com/SiliconValley/Way/7409/>

This is the help file for 'TIME TO WIN' for VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT} and MSOffice 95.

**TIME TO WIN for VB 4.0 (32-Bit) is COMPATIBLE with VB 5.0 and VB 5.0 CCE.**  
**Normally, TIME TO WIN for VB 4.0 (32-Bit) is compatible with all softwares supporting VBA 5.0 (MSOffice 97, Visio, ...)**

ANY REGISTERED USERS CAN ASK ME TO ADD SOME FUNCTIONNALITIES (non graphical routines).

## Overview

Current version                      Registration on CompuServe or on Internet

New features                        Revision history

Installation                        License agreement                      Distribution note

Technical support

Acknowledgement                      References

Other products

# DESencryptFile, DESdecryptFile

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[DESencryptFile](#) copy one file to an another file but with DES encryption.

[DESdecryptFile](#) copy one file to an another file but with DES decryption.

## Declare Syntax :

```
Declare Function cDESencryptFile Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, Key As String) As Long
```

```
Declare Function cDESdecryptFile Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, Key As String) As Long
```

## Call Syntax :

```
test& = cDESencryptFile(FileIn, FileOut, Key)
```

```
test& = cDESdecryptFile(FileIn, FileOut, Key)
```

## Where :

FileIn\$	is the source file.
FileOut\$	is the destination file.
Key	is the key to use for encryption/decryption.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

If the returned code is a negative value, it take the following value :

```
Public Const CRYPTO_KEY_TOO_SMALL = -1
Public Const CRYPTO_CANT_INIT_KEY = -2
Public Const CRYPTO_CANT_INIT_BUFFER = -11
Public Const CRYPTO_CANT_OPEN_FILEIN = -21
Public Const CRYPTO_CANT_CREATE_FILEOUT = -22
Public Const CRYPTO_ERROR_READING_FILEIN = -31
Public Const CRYPTO_ERROR1_WRITING_FILEOUT = -41
Public Const CRYPTO_ERROR2_WRITING_FILEOUT = -42
Public Const CRYPTO_ERROR1_WRITING_LASTBYTE = -51
Public Const CRYPTO_ERROR2_WRITING_LASTBYTE = -52
Public Const CRYPTO_BAD_LASTBYTE = -61
```

## Comments :

The Key is case sensitive.

The length of Text can be any size.

The length of Key must be greater or equal to 8 characters.

The encrypted file is always a multiple of 8 characters + 1 character.

## Examples :

```
Dim TestAs Long
```

```
Test = cDESencryptFile("c:\autoexec.bat", "c:\autoexec.tb1", "Time To Win")
```

```
Test = cDESdecryptFile("c:\autoexec.tb1", "c:\autoexec.tb2", "Time To Win")
```

**See also :** [Encryption](#)

TIME TO WIN for VB 3.0 : TIME2WIN.DLL : **9.52** (08/21/1997)  
TIME TO WIN for VB 4.0 (16-Bit) : T2WIN-16.DLL : **9.52** (08/21/1997)  
TIME TO WIN for VB 4.0 (32-Bit) : T2WIN-32.DLL : **5.43** (05/21/1998)  
TIME TO WIN for MSOffice 95 : T2WOFFIC.DLL : **2.02** (08/21/1996)

Select the following product :

TIME TO WIN for VB 3.0

TIME TO WIN for VB 4.0 (16-Bit)

TIME TO WIN for VB 4.0 (32-Bit)

TIME TO WIN for MSOffice 95

# TIME TO WIN for VB 3.0 : Installation

## **Demonstration version :**

The files TIME2WIN.DLL and TIME2WIN.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT\SYSTEM32 directory.

## **Registered version :**

The files TIME2WIN.DLL, TIME2WIN.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT\SYSTEM32 directory.

The file TIME2WIN.LIC should be copied in your WINDOWS or WIN95 directory.

## **Distribution note:**

When you create and distribute applications that use 'TIME TO WIN' dynamic link library, you should install the file 'TIME2WIN.DLL' in the customer's Microsoft Windows \SYSTEM or \SYSTEM32 subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install you applications correctly.

***You are not allowed to distribute TIME2WIN.LIC file with any application that you distribute.***

# TIME TO WIN for VB 4.0 (16-Bit) : Installation

## **Demonstration version :**

The files T2WIN-16.DLL and T2WIN-16.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT\SYSTEM32 directory.

## **Registered version :**

The files T2WIN-16.DLL, T2WIN-16.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT\SYSTEM32 directory.

The file T2WIN-16.LIC should be copied in your WINDOWS or WIN95 directory.

## **Distribution note:**

When you create and distribute applications that use 'TIME TO WIN (16-Bit)' dynamic link library, you should install the file 'T2WIN-16.DLL' in the customer's Microsoft Windows \SYSTEM or \SYSTEM32 subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install you applications correctly.

***You are not allowed to distribute T2WIN-16.LIC file with any application that you distribute.***

# TIME TO WIN for VB 4.0 (32-Bit) : Installation

## **Demonstration version :**

The files T2WIN-32.DLL and T2WIN-32.HLP should be copied in your WIN95\SYSTEM and/or WINNT\SYSTEM32 directory.

## **Registered version :**

The files T2WIN-32.DLL, T2WIN-32.HLP should be copied in your WIN95\SYSTEM and/or WINNT\SYSTEM32 directory.

The file T2WIN-32.REG should be fusionned in your REGISTRY.

## **Distribution note:**

When you create and distribute applications that use 'TIME TO WIN (32-Bit)' dynamic link library, you should install the file 'T2WIN-32.DLL' in the customer's Microsoft Windows \SYSTEM or \SYSTEM32 subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install you applications correctly.

***You are not allowed to distribute T2WIN-32.REG file with any application that you distribute.***

# TIME TO WIN for MSOffice 95 : Installation

## **Demonstration version :**

The files T2WOFFIC.DLL and T2WOFFIC.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT\SYSTEM32 directory.

## **Registered version :**

The files T2WOFFIC.DLL, T2WOFFIC.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT\SYSTEM32 directory.

The file T2WOFFIC.LIC should be copied in your WIN95 directory.

## **Distribution note:**

***You are not allowed to distribute T2WOFFIC.LIC file with any application that you distribute.***



# Technical support

**Only registered users can receive support and update.**

To receive support, you must specify your registration ID.

However, any report on any problem are the welcome.

The following information may be of help to you in streamlining your efforts to resolve any technical problems you may have with any version of TIME TO WIN Dynamic Link Library.

## **GPF?**

If you are getting a GPF (General Protection Fault), write down the information that is displayed when the error occurs. Also, make a note of what your code was doing (in general terms.)

## **ISOLATE IT**

Try to isolate the cause of the error. If at all possible, step through your code with F8 and F9. Try to find the one line of code that is causing the error.

## **SCALE IT DOWN**

If at all possible, try to reproduce the problem in a small test program that you can send in. Send your test on CompuServe.

## **Update**

You can download the update of all of my products on the following network :

On my homepage :

<http://ourworld.compuserve.com/homepages/alpouda/homepage.htm>

On CompuServe :

MSBASIC forum

On Internet :

T2WIN952.ZIP (ftp.winsite.com/pub/pc/win3/programr/vbasic OR

<http://www.geocities.com/SiliconValley/Way/7409>)

T2W16952.ZIP(ftp.winsite.com/pub/pc/win3/programr/vbasic OR

<http://www.geocities.com/SiliconValley/Way/7409>)

T2WIN-32.ZIP (ftp.winsite.com/pub/pc/win95/programr/vbasic OR

<http://www.geocities.com/SiliconValley/Way/7409>)

MCVBEHTP.ZIP (ftp.winsite.com/pub/pc/win95/programr/vbasic)

MCSECURE.ZIP (ftp.winsite.com/pub/pc/win95/programr/vbasic)

## **CompuServe Mail:**

**Name : Michael RENARD**

**CIS : 100042,3646**

**Internet : 100042.3646@compuserve.com**

I'm on CompuServe one time a day.

# License agreement

All versions of TIME TO WIN dynamic link library are not public domain software or free software.

All versions of TIME TO WIN dynamic link library are copyrighted, and all rights are reserved by its author: Michael Renard.

You are licensed to use this software on a restricted number of computers. You may copy the software to facilitate your use of it on as many computers as there are licensed users specified in the license file. Making copies for any other purpose violates international copyright laws.

***You are not allowed to distribute the [T2WIN-32.REG] file with any application that you distribute.***

## **Disclaimer:**

This software is sold AS IS without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. The authors assume no liability for any alleged or actual damages arising from the use of this software. (Some states do not allow the exclusion of implied warranties, so the exclusion may not apply to you.)

**Your use of this product indicates that you have read and agreed to these terms.**

## Distribution note

When you create and distribute applications that use a version of 'TIME TO WIN', you should install the [TIME TO WIN.DLL] in the customer's Microsoft Windows \SYSTEM or \SYSTEM32 subdirectory. The setup kit included with Visual Basic provides tools that help you write setup programs that install your applications correctly.

***You are not allowed to distribute the [T2WIN-32.REG] file with any application that you distribute.***

TIME TO WIN for VB 3.0 : time2win.dll  
TIME TO WIN for VB 4.0 (16-Bit) : t2win-16.dll  
TIME TO WIN for VB 4.0 (32-Bit) : t2win-32.dll, t2win32p.dll, t2w32pro.dll, t2w32exp.dll  
TIME TO WIN for MSOffice 95 : t2woffic.dll

TIME TO WIN for VB 3.0 : TIME2WIN.LIC  
TIME TO WIN for VB 4.0 (16-Bit) : T2WIN-16.LIC  
TIME TO WIN for VB 4.0 (32-Bit) : T2WIN-32.REG  
TIME TO WIN for MSOffice 95 : T2WOFFIC.LIC

# Acknowledgement

## For some translations :

Thanks to [Andreas Thoele](#) for German language.  
Thanks to [Silvio Sorrentino](#) for Italian language.  
Thanks to [Manuel Tobarra Narro](#) for Spanish language.  
Thanks to [Pawel Mandalian](#) for Polish language.  
Thanks to [Joan Ludevid](#) for Catalan language.  
Thanks to [Gian Kolbjornsen](#) for Norwegian language.  
Thanks to [Joakim Schramm](#) for Swedish language.

## For the following great developers :

Some routines have been written by the following great developers (I've adapted their routines for VB 3.0/4.0/5.0 under Win3.1x/Win95/WinNT) :

Special thanks to [J. Kercheval](#), [Michael M. Dodd](#), [Ray Gardner](#), [Bob Stout](#), [Thad Smith](#).  
Special thanks to [Brian Pirie](#) for [REGISTRATION KEY SYSTEM FOR C PROGRAMMERS](#).  
Special thanks to [Andy Brown](#) for [MD5 HASH ALGORITHM](#). (derived from the RSA \*\*\* Data Security, Inc. MD5 Message-Digest Algorithm).  
Special thanks to [Andy Brown](#) for adaptation of the U.S. [Data Encryption Standard \(DES\)](#) cipher in a Win3.1x DLL..  
Special thanks to [Andy Brown](#) for adaptation of the [International Data Encryption Algorithm \(IDEA\)](#) cipher in a Win3.1x DLL.  
Special thanks to [Michael Paul Johnson](#) for [Diamond Encryption Algorithm](#). [Andy Brown](#) have adapted Diamond in a Win 3.1x DLL.  
Special thanks to [Michael Paul Johnson](#) for [RUBY Mark 5 Algorithm](#).

About *DIAMOND* and *RUBY MARK 5* algorithm, you can reach Michael Paul Johnson at :

Internet e-mail : [mpj@csn.net](mailto:mpj@csn.net)  
Web site : <http://www.csn.net/~mpj> <- Get John (GLW)  
Colorado Catacombs BBS : 303-772-1062

Special thanks to [Haruhiko Okumura](#) for [Data Compression Algorithms of LARC and LHarc](#).  
Special thanks to [Jean-loup Gailly](#) and [Mark Adler](#) for [ZLIB](#) library.  
Special thanks to [Frank Pilhofer](#) and [Michael Newcomb](#) for [UUDeview Library](#). (UUDeview Library, (c) Frank Pilhofer and Michael Newcomb).

About *UUDeview Library*, you can find more informations on the UUDeview homepage at :

Web site : <http://www.uni-frankfurt.de/~fp/uudeview/>

Special thanks to [Armand Turpel](#) for [algomath-library](#).

## For registered users who ask me to add some new functions :

[Guillermo Kunst](#) for [EnumPrinterJobs](#), [Extract](#), [ExtractIsolate](#), [FStdIn](#), [FStdOut](#), [FStdErr](#).  
[Norm Zastre](#) for [3DWeightAverage](#), [FProcessAsciiFile](#), [FGotoRecord](#), [MaxNotX](#), [MinNotX](#).  
[John Sinnott](#) for [EnumOpenFiles](#).  
[Tom A. King](#) for [RUBY Mark 5 encryption](#).  
[George Srank](#) for [SplitFile](#), [CutFile](#), [FileMergeExt](#).  
[Scott Hayes](#) for [GetRegistryExt](#), [KillRegistryExt](#), [cPutRegistryExt](#).  
[Gary W. Andrews](#) for [GetFileDateTime](#), [cSetFileDateTime](#).  
[Morten Brun](#) for [FilesCopy](#).  
[Scott Hayes](#) for [FileForceCopy](#).  
[Oliver Nittel](#) for [ShortcutFileGetX](#), [cShortcutFileSetX](#).  
[Pietro B.](#) for [LockKeyboard](#), [cLockMouse](#).  
[William Joye](#) for [Mail encoder/decoder](#).  
[Mike Schuppe](#) for [SearchStr](#).  
[Doug Dimiceli](#) for [FileSearchFromLine](#), [FileSearchPatternFromLine](#), [GetAllSettings](#), [ArrayLookUp](#), [EnumPrinters](#), [Printer](#), [RunFile](#).  
[Bert Rozenberg](#) for [FileScanHeader](#), [FileScanHeaderForRecipients](#), [FilePartAppend](#), [FilePartCopy](#), [WrapLine](#), [TrashFile](#).  
[Syd S. Miller](#) for [ShortcutCreate](#), [ShortcutGetIconLocation](#), [ShortcutSetIconLocation](#).

This help has been written by using [ForeHelp](#) v1.04 from [ForeFront, Inc.](#)

# Other products

## Basis products :

### 1) [TIME TO WIN \(VB 4.0 \(32-Bit\), VB 5.0, VBA 5.0\)](#)

This product is a powerfull 32-Bit DLL with more than **913** routines for VB 4.0 (32-Bit), VB 5.0 and VBA 5.0 application.

You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select T2WIN-32.ZIP  
Internet : ftp.winsite.com/pub/pc/win95/programr/vbasic/T2W32520.ZIP  
: ftp.winsite.com/pub/pc/winNT/programr/vbasic/T2W32520.ZIP  
: http://www.geocities.com/SiliconValley/Way/7409/T2WIN-32.ZIP

You can register the full product (demo included) at the price of **\$59.95** from the following site :

CompuServe : GO SWREG : select the product item #7516  
Internet : use REGISTER.EXE in the ZIP'ed file : select [TIME TO WIN \(32-Bit\)](#)

### 2) [TIME TO WIN \(VB 3.0 or VB 4.0 \(16-Bit\)\)](#)

This product is a powerfull 16-Bit DLL with more than **720** routines for VB 3.0 and VB 4.0 (16-Bit) application.

You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select T2WIN952.ZIP and T2W16952.ZIP  
Internet : ftp.winsite.com/pub/pc/win3/programr/vbasic/T2WIN952.ZIP  
: http://www.geocities.com/SiliconValley/Way/7409/T2WIN952.ZIP  
: ftp.winsite.com/pub/pc/win3/programr/vbasic/T2W16952.ZIP  
: http://www.geocities.com/SiliconValley/Way/7409/T2W16952.ZIP

You can register the full product (demo included) at the price of **\$43.00** from the following site :

CompuServe : GO SWREG : select the product item #4045  
Internet : use REGISTER.EXE in the ZIP'ed file : select [TIME TO WIN \(16-Bit\)](#)

### 3) [TIME TO WIN for MS Office 95](#)

This product is a powerfull 32-Bit DLL with more than **380** routines for Access 95, Excel 95 and Word 95.

You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select T2WOFFIC.ZIP  
Internet : ftp.winsite.com/pub/pc/win95/programr/vbasic/T2WOFFIC.ZIP

You can register the full product (demo included) at the price of **\$25.00** from the following site :

CompuServe : GO SWREG : select the product item #10355  
Internet : use REGISTER.EXE in the ZIP'ed file : select [TIME TO WIN for MS Office 95](#)

### 4) [mcr VB/Error Handler - Tracer Profiler](#)

This product is a powerfull product for adding/removing the management of errors and tracer-profiler for project under VB 3.0, VB 4.0 (16-Bit) and VB 4.0 (32-Bit).

You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select MCVBEHTP.ZIP  
Internet : ftp.winsite.com/pub/pc/win95/programr/vbasic/MCVBEHTP.ZIP

You can register the full product (demo included) at the price of **\$25.00** from the following site :

CompuServe : GO SWREG : select the product item #4380  
Internet : use REGISTER.EXE in the ZIP'ed file : select [MCVBEHTP for UNregistered user ...](#)

### 5) [MC SECURITY for VB 4.0 \(16/32 Bit\)](#)

This product is a powerfull 16/32-Bit DLL with **28** routines for VB 4.0 (16/32 Bit), VB 5.0 and VBA 5.0 application.

You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select MCSECURE.ZIP  
Internet : ftp.winsite.com/pub/pc/win95/programr/vbasic/MCSEC113.ZIP

You can register the full product (demo included) at the price of **\$10.00** from the following site :



CompuServe : GO SWREG : select the product item #8536  
Internet : use REGISTER.EXE in the ZIP'ed file : select [MC-SECURITY \(16/32-Bit\)](#)

6) [MC STRING for VB 4.0 \(32 Bit\)](#)

This product is a powerful 32-Bit DLL with **74** routines for VB 4.0 (32 Bit), VB 5.0 and VBA 5.0 application.  
You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select MCSTR-32.ZIP  
Internet : <ftp.winsite.com/pub/pc/win95/programr/vbasic/MCSTR-32.ZIP>

You can register the full product (demo included) at the price of \$**10.00** from the following site :

CompuServe : GO SWREG : select the product item #12012  
Internet : use REGISTER.EXE in the ZIP'ed file : select [MC-STRING \(32-Bit\)](#)

7) [MC DISK VB 4.0 \(32 Bit\)](#)

This product is a powerful 32-Bit DLL with **94** routines for VB 4.0 (32 Bit), VB 5.0 and VBA 5.0 application.  
You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select MCDSK-32.ZIP  
Internet : <ftp.winsite.com/pub/pc/win95/programr/vbasic/MCDSK-32.ZIP>

You can register the full product (demo included) at the price of \$**10.00** from the following site :

CompuServe : GO SWREG : select the product item #12011  
Internet : use REGISTER.EXE in the ZIP'ed file : select [MC-DISK \(32-Bit\)](#)

Update products :

1) [Update TIME TO WIN \(VB 3.0 or VB 4.0 \(16-Bit\)\) -> TIME TO WIN 32-Bit \(VB 4.0 \(32-Bit\), VB 5.0, VBA 5.0\)](#)

This product is an update for registered user of 'TIME TO WIN' which want register the 'TIME TO WIN (32-Bit)'.  
You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select T2WIN-32.ZIP  
Internet : <ftp.winsite.com/pub/pc/win95/programr/vbasic/T2W32520.ZIP>  
: <ftp.winsite.com/pub/pc/winNT/programr/vbasic/T2W32520.ZIP>  
: <http://www.geocities.com/SiliconValley/Way/7409/T2WIN-32.ZIP>

You can register the full product (demo included) at the price of \$**34.95** from the following site :

CompuServe : GO SWREG : select the product item #7517  
Internet : use REGISTER.EXE in the ZIP'ed file : select [update TIME TO WIN \(16-Bit\) to TIME TO WIN \(32-Bit\)](#)

Special price for registered user :

1) [If you're a registered user of 'TIME TO WIN' or 'TIME TO WIN \(32-Bit\)](#)

You receive a special price for 'mcr VB/Error Handler - Tracer Profiler' under VB 3.0, VB 4.0 (16-Bit) and VB 4.0 (32-Bit).

You can download the full product (demo included) from the following site :

CompuServe : GO MSBASIC : select MCVBEHTP.ZIP  
Internet : <ftp.winsite.com/pub/pc/win95/programr/vbasic/MCVBEHTP.ZIP>

You can register the full product (demo included) at the price of \$**16.00** from the following site :

CompuServe : GO SWREG : select the product item #4379  
Internet : use REGISTER.EXE in the ZIP'ed file : select [MCVBEHTP for registered user ...](#)

# TIME TO WIN for VB 4.0 (16-Bit) : New features

See also : [Revision History](#)

Version	Comments
9.59	<i>no new features.</i>
9.52	Extracts a sub-string with a key in a string. <a href="#">Extract</a> Extracts a left/right part of a string from a key and a field separator. <a href="#">ExtractIsolate</a>
9.45	Initializes the Mail & News memory to perform encoding/decoding. <a href="#">MNInitialize</a> Shut downs the Mail & News memory (cleanup). <a href="#">MNShutdown</a> Retrieves options. <a href="#">MNGetOption</a> Sets options. <a href="#">MNSetOption</a> Encodes a file in single-part or multi-part using "UUE", "Mime-Base64", "XXE". <a href="#">MNEncode</a> Loads a file to be processed/decoded in the list of files. <a href="#">MNLoadInList</a> Counts the number of files in the list of files. <a href="#">MNCountInList</a> Retrieves a pointer to the first file in the list of files. <a href="#">MNFirstInList</a> Walks in the list of files. <a href="#">MNWalkInList</a> Retrieves the subject of a file in the list of files. <a href="#">MNGetSubjectInList</a> Retrieves the "mime identifier" (if any) of a file in the list of files. <a href="#">MNGetMimeIDInList</a> Retrieves the "mime content-type" (if any) of a file in the list of files. <a href="#">MNGetMimeContentInList</a> Renames a file in the list of files. <a href="#">MNRenameInList</a> Decodes a file from the list of files using "UUE", "Mime-Base64/PT/QP", "XXE", "BinHex". <a href="#">MNDecodeFromList</a> Deletes all occurrences of one string, in another string. <a href="#">DeleteSubString</a> Inserts a string (to the left) into a sub-string delimited by a separator in a given string. <a href="#">LSetIn</a> Inserts a string (to the right) into a sub-string delimited by a separator in a given string. <a href="#">RSetIn</a> Pads a string to the left, to a known length. <a href="#">LFill</a> Pads a string to the right, to a known length. <a href="#">RFill</a> Searches for known strings, and replaces them with another string. <a href="#">StringReplace</a> Compares a string with a set of strings delimited by a separator. <a href="#">MatchTable</a> Finds the position of the first occurrence of one string within another (like VB Instr function). <a href="#">InStr</a>
9.44	Uuencode/uudecode a file (this is can be usefull for Internet). <a href="#">FileUUCP</a>
9.43	Copy one file to an another file with the same file attribute. <a href="#">FileForceCopy</a>
9.34	Load the contents of a directory in a standard list box. <a href="#">ListFiles</a> Load the contents of a directory in a standard combo box. <a href="#">ComboFiles</a>

Search for dir(s) and save the result in a file. [SearchDir](#)  
Search for file(s) and save the result in a file. [SearchFile](#)  
Search for file(s) with attribute and save the result in a file.  
[SearchFileAttrib](#)  
Search for dir(s) using pattern matching and show the result in a standard list box.  
[ListSearchDir](#)  
Search for file(s) and show the result in a standard list box.  
[ListSearchFile](#)  
Search for file(s) with attribute and show the result in a standard list box.  
[ListSearchFileAttrib](#)  
Search for dir(s) using pattern matching and show the result in a standard combo box.  
[ComboSearchDir](#)  
Search for file(s) and show the result in a standard combo box.  
[ComboSearchFile](#)  
Search for file(s) with attribute and show the result in a standard combo box.  
[ComboSearchFileAttrib](#)

9.23 *no new features.*

9.25 *no new features.*

9.22 Disables/enables CTRL+ALT+DEL, ALT+TAB and CTRL+ESC under Win95.  
[MultitasksKeys95](#)

9.20 Convert a Single number into the Microsoft Binary Format representation.

[MKSMBF](#)

Convert a string containing the Microsoft Binary Format representation of a Single into a Single.

[CVSMBF](#)

Shift a Double array to Left and set the last element to a value.

[ShiftLeftD](#)

Shift an Integer array to Left and set the last element to a value.

[ShiftLeftI](#)

Shift a Long array to Left and set the last element to a value.

[ShiftLeftL](#)

Shift a Single array to Left and set the last element to a value.

[ShiftLeftS](#)

Shift a Double array to Right and set the first element to a value.

[ShiftRightD](#)

Shift an Integer array to Right and set the first element to a value.

[ShiftRightI](#)

Shift a Long array to Right and set the first element to a value.

[ShiftRightL](#)

Shift a Single array to Right and set the first element to a value.

[ShiftRightS](#)

Sort a Type'd array which can contain :

one or more elements of any data type, array, or a previously defined user-defined type.

[SortTypedArray](#)

9.10 Merges all files in an array in one.

[FileMergeExt](#)

9.08 Cut a file in two parts (creation of two files).

[CutFile](#)

Split a file into target files based on a part size.

[SplitFile](#)

Finding the maximum value in an array, not equal to a specified value.

[MaxNotXD](#), [MaxNotXI](#), [MaxNotXL](#), [MaxNotXS](#)

Finding the minimum value in an array, not equal to a specified value.

[MinNotXD](#), [MinNotXI](#), [MinNotXL](#), [MinNotXS](#)

9.07 *no new features.*

9.06 New command MB\_NO\_BUTTONS in [LngBoxMsg](#) and in [LngMsgBox](#).

Norwegian language supported (LNG\_NORVEGIAN) in [LngBoxMsg](#) and in [LngMsgBox](#).

Convert a Z9 string to an ascii string.

[FromZ9](#)

- Convert a ascii string to Z9 string. [ToZ9](#)
- Return if a given bit in a given BYTE is Set or Reset. [GetBitB](#)
- Return if a given bit in a given DOUBLE is Set or Reset. [GetBitD](#)
- Return if a given bit in a given INTEGER is Set or Reset. [GetBitI](#)
- Return if a given bit in a given LONG is Set or Reset. [GetBitL](#)
- Return if a given bit in a given SINGLE is Set or Reset. [GetBitS](#)
- Set a given bit in a given BYTE to Set state or Reset state. [SetBitB](#)
- Set a given bit in a given DOUBLE to Set state or Reset state. [SetBitD](#)
- Set a given bit in a given INTEGER to Set state or Reset state. [SetBitI](#)
- Set a given bit in a given LONG to Set state or Reset state. [SetBitL](#)
- Set a given bit in a given SINGLE to Set state or Reset state. [SetBitS](#)
- Convert an integer value into a binary string. [I2B](#)
- Convert a long value into a binary string. [L2B](#)
- Calculate the mod-10 of the given string. [Mod10](#)
- Calculate the reverse mod-10 of the given string. [Mod10R](#)
- Calculate the mod-11 of the given string. [Mod11](#)
- Calculate the reverse mod-11 of the given string. [Mod11R](#)
- Calculate the mod-1.3.7 of the given string. [Mod137](#)
- Calculate the reverse mod-1.3.7 of the given string. [Mod137R](#)
- 9.05 *no new features.*
- 9.04 Encode a string with a password using the RUBY algorithm (7 modes). [RUBYencrypt](#)  
 Decode a string with a password using the RUBY algorithm (7 modes). [RUBYdecrypt](#)  
 Copy one file to an another file but with RUBY algorithm (7 modes). [RUBYencryptFile](#)  
 Copy one file to an another file but with RUBY algorithm (7 modes). [RUBYdecryptFile](#)
- 9.01 Compress a string into a compressed format using GZIP compression method. [GZIPStringCompress](#)  
 Expand a compressed string into a normal format using GZIP compression method. [GZIPStringExpand](#)
- 9.00 Compress a file into a compressed format using GZIP compression method. [GZIPFileCompress](#)  
 Expand a compressed file into a normal format using GZIP compression method. [GZIPFileExpand](#)
- 8.08 *no new features.*
- 7.07 Conversion of a binary string into an integer variable. [B2I](#)  
 Conversion of a binary string into a long variable. [B2L](#)  
 Conversion of a hexa string into an integer variable. [H2I](#)  
 Conversion of a hexa string into a long variable. [H2L](#)  
 Access of method (by position) of OCX custom controls. [ObjectMethodByPos](#)  
 Access of method (by name) of OCX custom controls. [ObjectMethodByName](#)  
 Reads data in properties (by position) from OCX controls. [ObjectGetPropertyByPos](#)  
 Reads data in properties (by name) from OCX controls. [ObjectGetPropertyByName](#)  
 Writes data in properties (by position) in OCX controls. [ObjectPutPropertyByPos](#)  
 Writes data in properties (by name) from OCX controls. [ObjectPutPropertyByName](#)
- 7.00 Initial release of the 'TIME TO WIN (16-Bit)' Dynamic Link Library for Visual Basic 4.0 (16-Bit Edition).

Select the following product :

TIME TO WIN for VB 3.0

TIME TO WIN for VB 4.0 (16-Bit)

TIME TO WIN for VB 4.0 (32-Bit), VB 5.0, VBA 5.0

TIME TO WIN for MSOffice 95

# TIME TO WIN for VB 3.0 : New features

See also : [Revision History](#)

Version	Comments
9.59	<i>no new features.</i>
9.52	Extracts a sub-string with a key in a string. <a href="#">Extract</a> Extracts a left/right part of a string from a key and a field separator. <a href="#">ExtractIsolate</a>
9.45	Initializes the Mail & News memory to perform encoding/decoding. <a href="#">MNInitialize</a> Shut downs the Mail & News memory (cleanup). <a href="#">MNShutdown</a> Retrieves options. <a href="#">MNGetOption</a> Sets options. <a href="#">MNSetOption</a> Encodes a file in single-part or multi-part using "UUE", "Mime-Base64", "XXE". <a href="#">MNEncode</a> Loads a file to be processed/decoded in the list of files. <a href="#">MNLoadInList</a> Counts the number of files in the list of files. <a href="#">MNCountInList</a> Retrieves a pointer to the first file in the list of files. <a href="#">MNFirstInList</a> Walks in the list of files. <a href="#">MNWalkInList</a> Retrieves the subject of a file in the list of files. <a href="#">MNGetSubjectInList</a> Retrieves the "mime identifier" (if any) of a file in the list of files. <a href="#">MNGetMimeIDInList</a> Retrieves the "mime content-type" (if any) of a file in the list of files. <a href="#">MNGetMimeContentInList</a> Renames a file in the list of files. <a href="#">MNRenameInList</a> Decodes a file from the list of files using "UUE", "Mime-Base64/PT/QP", "XXE", "BinHex". <a href="#">MNDecodeFromList</a> Deletes all occurrences of one string, in another string. <a href="#">DeleteSubString</a> Inserts a string (to the left) into a sub-string delimited by a separator in a given string. <a href="#">LSetIn</a> Inserts a string (to the right) into a sub-string delimited by a separator in a given string. <a href="#">RSetIn</a> Pads a string to the left, to a know length. <a href="#">LFill</a> Pads a string to the left, to a know length. <a href="#">RFill</a> Searches for known strings, and replaces them with another string. <a href="#">StringReplace</a> Compares a string with a set of strings delimited by a separator. <a href="#">MatchTable</a> Finds the position of the first occurrence of one string within another (like VB Instr function). <a href="#">InStr</a>
9.44	<i>no new features.</i>
9.43	Copy one file to an another file with the same file attribute. <a href="#">FileForceCopy</a>
9.34	Load the contents of a directory in a standard list box. <a href="#">ListFiles</a> Load the contents of a directory in a standard combo box. <a href="#">ComboFiles</a> Search for dir(s) and save the result in a file. <a href="#">SearchDir</a>

- Search for file(s) and save the result in a file. [SearchFile](#)  
 Search for file(s) with attribute and save the result in a file.  
[SearchFileAttrib](#)  
 Search for dir(s) using pattern matching and show the result in a standard list box.  
[ListSearchDir](#)  
 Search for file(s) and show the result in a standard list box.  
[ListSearchFile](#)  
 Search for file(s) with attribute and show the result in a standard list box.  
[ListSearchFileAttrib](#)  
 Search for dir(s) using pattern matching and show the result in a standard combo box.  
[ComboSearchDir](#)  
 Search for file(s) and show the result in a standard combo box.  
[ComboSearchFile](#)  
 Search for file(s) with attribute and show the result in a standard combo box.  
[ComboSearchFileAttrib](#)
- 9.25 *no new features.*
- 9.23 *no new features.*
- 9.22 Disables/enables CTRL+ALT+DEL, ALT+TAB and CTRL+ESC under Win95.  
[MultitasksKeys95](#)
- 9.20 Convert a Single number into the Microsoft Binary Format representation.  
[MKSMBF](#)  
 Convert a string containing the Microsoft Binary Format representation of a Single into a Single.  
[CVSMBF](#)  
 Shift a Double/Integer/Long/Single array to Left and set the last element to a value.  
[ShiftLeftX](#)  
 Shift a Double/Integer/Long/Single array to Right and set the first element to a value.  
[ShiftRightX](#)
- 9.10 Merges all files in an array in one.  
[FileMergeExt](#)
- 9.08 Cut a file in two parts (creation of two files). [CutFile](#)  
 Split a file into target files based on a part size. [SplitFile](#)  
 Finding the maximum value in an array, not equal to a specified value.  
[MaxNotXD](#), [MaxNotXI](#), [MaxNotXL](#), [MaxNotXS](#)  
 Finding the minimum value in an array, not equal to a specified value.  
[MinNotXD](#), [MinNotXI](#), [MinNotXL](#), [MinNotXS](#)
- 9.07 *no new features.*
- 9.06 New command MB\_NO\_BUTTONS in [LngBoxMsg](#) and in [LngMsgBox](#).  
 Norwegian language supported (LNG\_NORVEGIAN) in [LngBoxMsg](#) and in [LngMsgBox](#).  
 Convert a Z9 string to an ascii string.  
 Convert a ascii string to Z9 string.  
 Return if a given bit in a given Byte/Double/Integer/Long/Single is Set or Reset.  
 Set a given bit in a given Byte/Double/Integer/Long/Single to Set state or Reset state.  
 Convert an integer value into a binary string.  
 Convert a long value into a binary string.  
 Calculate the mod-10 of the given string.  
 Calculate the reverse mod-10 of the given string.  
 Calculate the mod-11 of the given string.  
 Calculate the reverse mod-11 of the given string.  
 Calculate the mod-1.3.7 of the given string.  
 Calculate the reverse mod-1.3.7 of the given string.  
[Mod137R](#)
- 9.05 *no new features.*
- [FromZ9](#)  
[ToZ9](#)  
[GetBitX](#)  
[SetBitX](#)  
[I2B](#)  
[L2B](#)  
[Mod10](#)  
[Mod10R](#)  
[Mod11](#)  
[Mod11R](#)  
[Mod137](#)

- 9.04 Encode a string with a password using the RUBY algorithm (7 modes). [RUBYencrypt](#)  
 Decode a string with a password using the RUBY algorithm (7 modes). [RUBYdecrypt](#)  
 Copy one file to another file but with RUBY algorithm (7 modes).  
[RUBYencryptFile](#)  
 Copy one file to another file but with RUBY algorithm (7 modes).  
[RUBYdecryptFile](#)
- 9.01 Compress a string into a compressed format using GZIP compression method. [GZIPStringCompress](#)  
 Expand a compressed string into a normal format using GZIP compression method.  
[GZIPStringExpand](#)
- 9.00 Compress a file into a compressed format using GZIP compression method. [GZIPFileCompress](#)  
 Expand a compressed file into a normal format using GZIP compression method. [GZIPFileExpand](#)
- 8.08 *no new features.*
- 7.01 Adds new functionalities for language management by using only one file per language. [c.x.CtlLanguage](#)
- 7.00 *no new features.*
- 6.01 Implementation for CATALAN language : LNG\_CATALAN.  
 Constants and Types declaration  
 Implementation for POLISH language : LNG\_POLISH.  
 Constants and Types declaration  
 Counts a specific value in an Integer/Long/Single/Double array. [cCountX](#)  
 Searches a specific value in an Integer/Long/Single/Double array.  
[cSearchX](#)
- 6.00 Truncates a long path with filename. [cTruncatePath](#)  
 Searches and replace a string in a string (search can be case-sensitive or not).  
[cStringSAR](#)  
 Initializes the random generator. [cRndInit](#)  
 Returns a double random number between 0.0 and 1.0. [cRnd](#)  
 Returns an Integer/Long/Single/Double random number. [cRndX](#)
- 5.29 Returns a number in the form of a fraction. [cFraction](#)  
 Spells money value with hundredth. [cSpellMoney](#)  
 Creates or updates a file which contains the text (menu) for supporting a language.  
[cSaveMnuLanguage](#)  
 Reads a file which contains the text (menu) for supporting a language.  
[cReadMnuLanguage](#)  
 Logical size of files by file mask in a specified directory (with recursivity or not).  
[cRcsFilesSize](#)  
 Physical size of files by file mask in a specified directory (with recursivity or not).  
[cRcsFilesSizeOnDisk](#)  
 Slack percent for files by file mask in a specified directory (with recursivity or not). [cRcsFilesSlack](#)  
 Reads all files from a specified directory into an array.  
[cFilesInDirToArray](#)  
 Writes all files from a specified directory into a file on disk.  
[cFilesInDirOnDisk](#)  
 Counts the total directories or files in a specified directory (with recursivity or not). [cRcsCountFileDir](#)  
 Returns name, size, Int date, Int time, attribute of files in directory only in one call.  
[cFilesInfoInDir](#)
- 5.20 *no new features.*
- 5.10 Adds 6 Hatch Brush Pattern for 3DMeter.





- Get/Put a Huge Array from/to a file on disk.  
cHMAOnDisk
- 4.00 Adds a VB string into a Huge String.  
cHugeStrAdd  
Returns a pointer for the first char of a Huge String.  
cHugeStrAddress  
Appends a VB string into a Huge String.  
cHugeStrAppend  
Returns the number of block of 64,000 chars from a Huge String.  
cHugeStrBlocks  
Clears a Huge String.  
cHugeStrClear  
Creates a Huge String.  
cHugeStrCreate  
Free a Huge String (destroy it).  
cHugeStrFree  
Gets the Next Pointer of a Huge String.  
cHugeStrGetNP  
Gets the Write Pointer of a Huge String.  
cHugeStrGetWP  
Returns the length of data in a Huge String.  
cHugeStrLength  
Extracts a VB sub-string from a Huge String. cHugeStrMid  
Reads the next part of a Huge String.  
cHugeStrNext  
Get/Put a Huge String from/to a file on disk.  
cHugeStrOnDisk  
Read a block of 64,000 chars or minder from a Huge String.  
cHugeStrRead  
Sets the Next Pointer of a Huge String.  
cHugeStrSetNP  
Sets the Write Pointer of a Huge String.  
cHugeStrSetWP  
Returns the full size of a Huge String.  
cHugeStrSize
- 3.52 Increment the number of file handle (20 -> 80).
- 3.51 *no new features.*
- 3.50 Extracts a sub-string from the right of a given string. cGetInR  
Extracts the first/second part from the left of a given string.  
cGetInPart  
Extracts the first/second part from the right of a given string.  
cGetInPartR  
Returns the version number of 'TIME TO WIN'.  
cGetVersion
- 3.00 Calculates the day of the week (ISO and non-ISO specification).  
cDayOfWeek  
Calculates the week of the year (ISO and non-ISO specification).  
cWeekOfYear  
Calculates the day of the year.  
cDayOfYear  
Calculates a Scalar (long) from a date.  
cDateToScalar  
Decomposes a Scalar into date parts.  
cScalarToDate  
Transfers the contents of a file to a List Box.  
cFileToListBox

- Transfers the contents of a file to a Combo Box.  
cFileToComboBox
- Performs some special effects between two Picture Box.  
 cFXPicture
- Auto-increments an integer variable. cIncrI
- Auto-increments a long variable. cIncrL
- Auto-decrements an integer variable. cDecrI
- Auto-decrements a long variable. cDecrL
- Adds two time string and return a time string. cAddTwoTimes
- Create a new multiple big sized array on disk or use an existing big sized array on disk.  
cMDACreate
- Close a multiple big sized array and keep it or close a big sized array and destroy it.  
cMDAClose
- Read an element from a multiple big sized array on disk.  
cMDAGet
- Read a type'd variable from a multiple big sized array on disk.  
cMDAGetType
- Save an element to a multiple big sized array on disk. cMDAPut
- Save a type'd variable to a multiple big sized array on disk.  
cMDAPutType
- Clear a multiple big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).  
cMDAClear
- Clear a single Sheet in a multiple big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).  
cMDAClearSheet
- Clear a single Col on one Sheet or on all sheets in a multiple big sized array (see above).  
cMDAClearCol
- Clear a single Row on one Sheet or on all Sheets in a multiple big sized array (see above).  
cMDAClearRow
- Clear a single Col in a multiple big sized array with only one sheet.  
cMDAsClearCol
- Clear a single Row in a multiple big sized array with only one sheet.  
cMDAsClearRow
- Read an element from a multiple big sized array on disk with only one sheet.  
cMDAsGet
- Read a type'd variable from a multiple big sized array on disk with only one sheet.  
cMDAsGetType
- Save an element from a multiple big sized array on disk with only one sheet.  
cMDAsPut
- Save a type'd variable from a multiple big sized array on disk with only one sheet.  
cMDAsPutType
- Read an element from a multiple big sized array on disk with only one sheet and one row. cMDArGet
- Read a type'd variable from a multiple big sized array on disk with only one sheet and one row.  
cMDArGetType
- Save an element from a multiple big sized array on disk with only one sheet and one row. cMDArPut
- Save a type'd variable from a multiple big sized array on disk with only one sheet and one row.  
cMDArPutType
- 2.05 Reads the Volume Label from a disk.  
cDOSGetVolumeLabel  
 Creates/Changes/Deletes the Volume Label of a disk.  
cDOSSetVolumeLabel  
 Gets information from a floppy disk (Format, Heads, Cylinders, Sectors). cFloppyInfo
- 2.00 Converts the first letter of some words separated by a space or punctuation in upper letter case.  
cProperName2  
 Reads the media ID (serial number, volume label, ...) from a disk.  
cDOSGetMediaID  
 Changes the media ID (serial number, volume label, ...) to a disk.  
cDOSSetMediaID  
 Compress a file.  
cFileCompress

	Expands a file compressed by cFileCompress. <u>cFileExpand</u>	
	Compress a string. <u>cStringCompress</u>	
	Expands a string compressed by cStringCompress. <u>cStringExpand</u>	
	Fills an array by starting value and increment value.	<u>FillIncr</u>
	Calculates the determinant of a square matrix. <u>cMatrixDet</u>	
	Calculates the cofactor of an element in a square matrix. <u>cMatrixCoFactor</u>	
	Calculates the minor of an element in a square matrix. Fills a square matrix. <u>cMatrixFill</u>	<u>cMatrixMinor</u>
	Inverts a square matrix. <u>cMatrixInv</u>	
	Creates a symmetrical Toeplitz square matrix. <u>cMatrixSymToeplitz</u>	
1.60	2-D Geometry calculations (14 functions). <u>Geometry</u>	<u>2-D</u>
	3-D Geometry calculations (14 functions). <u>Geometry</u>	<u>3-D</u>
	Adds two square matrix. <u>cMatrixAdd</u>	
	Compares two square matrix. <u>cMatrixCompare</u>	
	Copy a square matrix. <u>cMatrixCopy</u>	
	Multiply two square matrix. Subtract two square matrix. <u>cMatrixSub</u>	<u>cMatrixMul</u>
	Transpose a square matrix. <u>cMatrixTranspose</u>	
1.52	Converts the first letter of each word separated by a space in a string to upper case. <u>cProperName</u>	
1.50	Functions for calculating interest rate (12 functions). <u>Financial</u>	
1.42	Performs the hash algorithm (MD5) to a specified string. <u>cHashMD5</u>	
	Adds registration key management.	<u>cRegistrationKey</u>
	Removes a serialization information (descriptions and number) from a serialized file. <u>cSerialRmvy</u>	
	Sorts an ASCII file or a BINARY file in ascending or descending order with case sensitive or not.	<u>cFileSort</u>
	Computes the number of combinations of n items, taken m at a time. <u>cCombination</u>	
	Converts an ASCII string into an EBCDIC string. <u>cCnvASCIItoEBCDIC</u>	
	Converts an EBCDIC string into an ASCII string. <u>cCnvEBCDICtoASCII</u>	
	Opens a file for I/O	<u>cFopen</u>
	Closes an open stream.	<u>cFclose</u>
	Reads a single character from a stream.	<u>cFgetc</u>
	Writes a single character to a stream.	<u>cFputc</u>
	Writes a line of characters to a stream.	<u>cFputs</u>
	Reads a line of characters from a stream.	<u>cFgets</u>
	Writes an arbitrary number of characters to a stream.	<u>cFwrite</u>
	Reads an arbitrary number of characters from a stream.	<u>cFread</u>

	Closes all files opened.	
	<u>cFcloseall</u>	
	Flushes buffered I/O to a particular stream to disk.	<u>cFlush</u>
	Flushes buffered I/O for all open streams to disk.	
	<u>cFlushall</u>	
	Tests for end-of-file on a stream.	<u>cFeof</u>
	Tests for an error on a stream.	<u>cFerror</u>
	Resets the error indicator for a stream.	
	<u>cFclearerr</u>	
	Moves the file pointer to a specified location.	<u>cFseek</u>
	Gets the current position of a file pointer.	<u>cFtell</u>
	Moves the file pointer to the beginning of a file.	
	<u>cFrewind</u>	
1.36	Arrange all desktop icons.	
	<u>cArrangeDesktopIcons</u>	
	Put/Get full variable string array (one dimension) on/from disk.	
	<u>cArrayStringOnDisk</u>	
	Put/Get full array (any dimension) on/from disk (very fast routine).	
	<u>cArrayOnDisk</u>	
	Extract a sub-string delimited by a separator's list in a given string.	
	<u>cTokenIn</u>	
	Align a string in left, center, right position	<u>cAlign</u>
	New timer for more accuracy (1 ms in place of 55 ms)	<u>cTimer.x</u>
	Increment the serialized number of a serialized file by a value (positive or negative).	
	<u>cSerialInc</u>	
	Check if a file is serialized.	<u>clsSerial</u>
	Put or Modify a serialization information (descriptions and number) to a file.	
	<u>cSerialPut</u>	
	Get a serialization information (descriptions and number) from a file.	
	<u>cSerialGet</u>	
	Walk thru the window's list.	
	<u>cWalkThruWindow</u>	
	UnHide all edit forms in the VB design environment.	
	<u>cUnHideAllEditForm</u>	
	Hide all edit forms in the VB design environment.	
	<u>cHideAllEditForm</u>	
	UnHide debug form in the VB design environment.	
	<u>cUnHideDebugForm</u>	
	Hide debug form in the VB design environment.	
	<u>cHideDebugForm</u>	
	Multiple AND 'InStr' in one call.	
	<u>cAndToken</u> , <u>cAndTokenIn</u>	
	Multiple OR 'InStr' in one call.	
	<u>cOrToken</u> , <u>cOrTokenIn</u>	
1.33	Close all edit forms in the VB design environment.	
	<u>cCloseAllEditForm</u>	
	Create a multiple directory in one call.	
	<u>cMakeMultipleDir</u>	
1.30	Clear a single Sheet in a big sized array (fill it with chr\$(0) or chr\$(32) (for string array))	
	<u>cDAClearSheet</u>	
	Clear a single Col on on one Sheet or on all sheets in a big sized array (see above).	
	<u>cDAClearCol</u>	
	Clear a single Row on one Sheet or on all Sheets in a big sized array (see above).	
	<u>cDAClearRow</u>	
	Clear a single Col in a big sized array with only one sheet.	
	<u>cDAsClearCol</u>	
	Clear a single Row in a big sized array with only one sheet.	
	<u>cDAsClearRow</u>	
	Read an element from a big sized array on disk with only one sheet.	<u>cDAsGet</u>

- Read a type'd variable from a big sized array on disk with only one sheet.  
cDAsGetType
- Save an element from a big sized array on disk with only one sheet. cDAsPut
- Save a type'd variable from a big sized array on disk with only one sheet.  
cDAsPutType
- Read an element from a big sized array on disk with only one sheet and one row. cDArGet
- Read a type'd variable from a big sized array on disk with only one sheet and one row.  
cDArGetType
- Save an element from a big sized array on disk with only one sheet and one row. cDArPut
- Save a type'd variable from a big sized array on disk with only one sheet and one row.  
cDArPutType
- 1.28 Merge two files in one.  
cFileMerge
- Search and replace a string in a file (search can be case-sensitive or not).  
cFileSearchAndReplace
- Search a string in a file (search is case-sensitive or not).  
cFileSearch
- Count occurrence of a string in a file (search can be case-sensitive or not).  
cFileSearchCount
- Check the specified ISBN (International Standard Book Numbers). clsISBN
- Extend the use of pattern matching with [..], [!..] constructs and hexa.  
cPatternExtMatch
- Convert a string into a morse string. cMorse
- Kill a group of files even if one or more file are read-only file in the directory and all sub-dirs.  
cKillDirFilesAll
- Kill a sub-directory and its associated directories. cKillDirs
- Base conversion between two radixs.  
cBaseConversion
- Count lines, words and chars in a file.  
cFileStatistics
- Create a new big sized array on disk or use an existing big sized array on disk.  
cDACreate
- Close an big sized array and keep it or close a big sized array and destroy it.  
cDAClose
- Read an element from a big sized array on disk. cDAGet
- Read a type'd variable from a big sized array on disk.  
cDAGetType
- Save an element to a big sized array on disk. cDAPut
- Save a type'd variable to a big sized array on disk.  
cDAPutType
- Clear a big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).  
cDAClear
- 1.22 Modification of a system menu in one call (6 different languages)  
cLngSysMenu
- 1.21 Multi-Language Message Box (fully replacement of the standard sub MsgBox)  
cLngBoxMsg
- Multi-Language Message Box (fully replacement of the standard function MsgBox)  
cLngMsgBox
- Multi-Language InputBox (fully replacement of the standard function InputBox\$)  
cLngInpBox
- Convert a partial path stored in a path to a fully qualified path.  
cFullPath
- Make a full qualified path composed of a drive letter, directory, filename, extension  
cMakePath
- Mix all chars in a given string in random position.  
cMixChars
- Kill a file even if the file is a read-only file.  
cKillFileAll

- Kill a group of file even if one or more file are read-only file.  
cKillFilesAll
- Count the total number of lines in an ASCII file.  
cFileLineCount
- Convert an ASCII file to a file with lower case char.  
cFileToLower
- Convert an ASCII file to a file with upper case char.  
cFileToUpper
- Operation on big numbers (big double) cBig.x.
- Convert a value (in the form of a string) into a big double representation (for use with cBig.x.) cMKN
- Operation on big numbers (in the form of a string)  
cBigNum
- 1.14 Compare one file to another file (attribute, contents, size, time) cCmpFile.x.  
Copy a file to an another file  
cFileCopy  
Copy a file to an another file but with filtering some chars  
cFileFilter  
Copy a file to an another file but with filtering chars not present in the filter  
cFileFilterNot  
Copy a file to an another file but with encryption  
cFileEncrypt  
Copy a file to an another file but with decryption  
cFileDecrypt  
Copy a file to an another file but with compressing spaces into tab  
cFileCompressTab  
Copy a file to an another file but with expanding tab into spaces  
cFileExpandTab  
Split a full path breaks into its four components.  
cSplitPath  
Check if the name of a file is valid  
clsFilenameValid
- 1.07 Implementation for some languages : French, Dutch, German, English, Italian, Spanish.  
Constants and Types declaration  
Full implementation for extracting the day name and the month name in different language.  
cGet.x.Day, cGet.x.Month  
Date and time in a normalized string in different language from a language number  
cGetAscTime  
Cluster size on a specified disk.  
cGetDiskClusterSize  
Physical size of files by file mask on a disk.  
cFilesSizeOnDisk  
Slack percent for files by file mask on a disk. cFilesSlack  
State (enabled or disabled) of a form.  
clsFormEnabled  
Full class name of a specified control.  
cGetClassName  
Save/Read language information from a form c.x.CtlLanguage
- 1.00 Initial release of the 'TIME TO WIN' dynamic link library.

# TIME TO WIN for VB 4.0 (32-Bit), VB 5.0 and VBA 5.0 : New features

See also : [Revision History](#)

Version	Comments
5.43	Compress a string into a compressed format using GZIP compression method. <a href="#">GZIPStringCompress2</a> Expand a compressed string into a normal format using GZIP compression method. <a href="#">GZIPStringExpand2</a>
5.34	Creates a shortcut from a specified file. <a href="#">ShortcutCreate</a> Retrieves the icon location from a shortcut file (*.lnk). <a href="#">ShortcutFileGetIconLocation</a> Modifies the icon location in a shortcut file (*.lnk). <a href="#">ShortcutFileSetIconLocation</a> Opens a specified executable or document file. <a href="#">RunFile</a> Enumerates all printers (data returned are description, printer name, comment). <a href="#">EnumPrinters1</a> Enumerates all printers (data returned are for remote printers). <a href="#">EnumPrinters2</a> Enumerates all printers (data returned are printer name, port name). <a href="#">EnumPrinters5</a> Sets the default printer. <a href="#">SetPrinterDefault</a>
5.25	Retrieves country-specific dialing information from the Windows Telephony list of countries. <a href="#">RasGetCountryInfo</a> Gets the number of copies of a printer. <a href="#">GetPrinterCopies</a> Gets the default source of a printer. <a href="#">GetPrinterDefaultSource</a> Gets the dither type of pictures of a printer. <a href="#">GetPrinterDitherType</a> Gets the paper orientation of a printer. <a href="#">GetPrinterOrientation</a> Gets the paper size of a printer. <a href="#">GetPrinterPaper</a> Gets the quality for a printer. <a href="#">GetPrinterQuality</a> Sets the number of copies for a printer. <a href="#">SetPrinterCopies</a> Sets the default source for a printer. <a href="#">SetPrinterDefaultSource</a> Sets the dither type of pictures for a printer. <a href="#">SetPrinterDitherType</a> Sets the paper orientation for a printer. <a href="#">SetPrinterOrientation</a> Sets the quality for a printer. <a href="#">SetPrinterQuality</a> Sends a file to the trash with persistence and confirmation. <a href="#">TrashFile</a> Retrieves the short path form of a specified input path. <a href="#">GetShortPathName</a> Wraps a file in multiple lines with a maximum length by line. <a href="#">FileWrapLine</a>



- 
- Increasing from 7 keys to 10 keys to Sort a Type'd array.  
[SortTypedArray](#)
- 
- Converts an integer value into a binary string (custom chars).  
[I2Bext](#)
- Converts a long value into a binary string (custom chars).  
[L2Bext](#)
- 5.20 Searches an element in a two-dimensionnal string array and returns value in the second dimension.  
[ArrayLookUp](#)
- Returns a list of key settings and their respective values from an application's entry in the registry.  
[GetAllSettings](#)
- Wraps a line in multiple lines with a maximum length by line.  
[WrapLine](#)
- Appends part of one file to another file starting at or until Offset.  
[FilePartAppend](#)
- Copies part of one file to another file starting at or until Offset.  
[FilePartCopy](#)
- Checks if the system uses Win95 OSR2 or not.  
[IsWin95OSR2](#)
- Rounds a number with a precision. [Round](#)
- Rounds a number to the nearest value.  
[RoundNearest](#)
- Retrieves informations about the specified key in the Windows registry.  
[RegistryKeyInfo](#)
- Searchs a string in a HEADER file starting at a certain line number  
[FileScanHeader](#)
- Searchs a string (recipients like "to: ", "cc: ", ...) in a given HEADER file starting at a certain line number.  
[FileScanHeaderForRecipients](#)
- Searchs a pattern string in a given TEXT file starting at a certain line number.  
[FileSearchPatternFromLine](#)
- Verifies if the CAPS LOCK key is On.  
[IsCapsLockOn](#)
- Verifies if the INSERT key is On.  
[IsInsertOn](#)
- Verifies if the NUM LOCK key is On.  
[IsNumLockOn](#)
- Verifies if the SCROLL LOCK key is On.  
[IsScrollLockOn](#)
- Sets the CAPS LOCK key On or Off.  
[SetCapsLock](#)
- Sets the INSERT key On or Off.  
[SetInsert](#)
- Sets the NUM LOCK key On or Off.  
[SetNumLock](#)
- Sets the SCROLL LOCK key On or Off.  
[SetScrollLock](#)
- Swedish language supported (LNG\_SWEDISH).
- 5.11 Searches a string in a given TEXT file starting at a certain line number.  
[FileSearchFromLine](#)
- Counts the total number of lines (maximum 32768 chars per line) in an ASCII file.  
[FileLineCount2](#)
- Returns the Greatest Common Divisor of two numbers.  
[GreatCommonDivisor](#)
- Rotates the digits of a number, x times, in base 10.  
[Rotate](#)
- Sorts the digits of a number.  
[SortDigits](#)
- Returns the sum of the digits of a number.  
[SumDigits](#)
- Returns the alternating sum of the digits of a number.  
[SumDigitsAlt](#)

- Returns the sum of all possible divisors of a number, number not included.  
[SumDivisors](#)
- Checks if a number is a prime number.  
[IsPrime](#)
- 5.09 Searches a string array for a given string.  
[SearchStr](#)  
 Returns the height of the printer in inch.  
[PrinterHeight](#)  
 Returns the left offset of the printer in inch (begin of the printable area).  
[PrinterOffsetLeft](#)  
 Returns the top offset of the printer in inch (begin of the printable area).  
[PrinterOffsetTop](#)  
 Returns the height of the printer in inch.  
[PrinterWidth](#)
- 5.02 Creates a gradient (8 effects) on a control/form which can accept .hDC  
[Gradient](#)
- 5.01 Replacement of cCVx(MID\$(string, offset, length)).  
[GetCVB](#) [GetCVC](#) [GetCVD](#) [GetCVI](#) [GetCVL](#) [GetCVS](#)  
 Replacement of Mid\$(string, offset, length) = Value.  
[PutMKB](#) [PutMKC](#) [PutMKD](#) [PutMKI](#) [PutMKL](#) [PutMKS](#)  
 Replacement of Mid\$(string, offset, length) = Value for a field in a recordset.  
[PutMKBs](#) [PutMKCs](#) [PutMKDs](#) [PutMKIs](#) [PutMKLs](#) [PutMKsS](#)
- 5.00 Retrieves the encoded address of the network adapter (MAC address).  
[GetNetAdapterMacAddress](#)  
 Retrieves the logical address of a physical LAN adapter.  
[GetNetAdapterNumber](#)  
 Retrieves the number of LAN adapter (number of network card).  
[GetNetNumberOfAdapter](#)  
 Returns the seconds elapsed since midnight.  
[ExtCurrentTime](#)
- 4.52 Identifies the build number of the operating system.  
[GetOSBuildNumber](#)  
 Provides arbitrary additional information about the operating system.  
[GetOSCDVersion](#)  
 Identifies the major version number of the operating system.  
[GetOSMajorVersion](#)  
 Identifies the minor version number of the operating system.  
[GetOSMinorVersion](#)  
 Identifies the version number of the operating system (major.minor version).  
[GetOSVersion](#)  
 Identifies the operating system platform in a numerical format.  
[GetOSPlatformId](#)  
 Identifies the operating system platform in a readable format (string).  
[GetOsPlatformName](#)  
 Retrieves the current default user name or the user name used to establish a network connection.  
[GetNetUser](#)  
 Set a given bit (real bit position) in a given BYTE to Set state or Reset state.  
[SetBitB2](#)  
 Set a given bit (real bit position) in a given INTEGER to Set state or Reset state.  
[SetBitI2](#)  
 Set a given bit (real bit position) in a given LONG to Set state or Reset state.  
[SetBitL2](#)  
 Returns if a given bit (real bit position) in a given BYTE is Set or Reset.  
[GetBitB2](#)  
 Returns if a given bit (real bit position) in a given INTEGER is Set or Reset.  
[GetBitI2](#)

Returns if a given bit (real bit position) in a given LONG is Set or Reset.

[GetBitL2](#)

- 4.43 [RmvDupD](#)  
Removes duplicate values in a Double array (one and two dimensions).
- [RmvDupI](#)  
Removes duplicate values in an Integer array (one and two dimensions).
- [RmvDupL](#)  
Removes duplicate values in a Long array (one and two dimensions).
- [RmvDupS](#)  
Removes duplicate values in a Single array (one and two dimensions).
- [TrayBar](#)  
Hides or shows the tray bar.
- [Extract](#)  
Extracts a sub-string with a key in a string.
- [ExtractIsolate](#)  
Extracts a left/right part of a string from a key and a field separator.
- [FStdIn](#)  
Returns the stream of the standard input.
- [FStdOut](#)  
Returns the stream of the standard output.
- [FStdErr](#)  
Returns the stream of the standard error.
- 4.34 [MNInitialize](#)  
Initializes the Mail & News memory to perform encoding/decoding.
- [MNShutdown](#)  
Shut downs the Mail & News memory (cleanup).
- [MNGetOption](#)  
Retrieves options.
- [MNSetOption](#)  
Sets options.
- [MNEncode](#)  
Encodes a file in single-part or multi-part using "UUE", "Mime-Base64", "XXE".
- [MNLdInList](#)  
Loads a file to be processed/decoded in the list of files.
- [MNCountInList](#)  
Counts the number of files in the list of files.
- [MNFIRSTInList](#)  
Retrieves a pointer to the first file in the list of files.
- [MNWalkInList](#)  
Walks in the list of files.
- [MNGetSubjectInList](#)  
Retrieves the subject of a file in the list of files.
- [MNGetMimeIDInList](#)  
Retrieves the "mime identifier" (if any) of a file in the list of files.
- [MNGetMimeContentInList](#)  
Retrieves the "mime content-type" (if any) of a file in the list of files.
- [MNRenameInList](#)  
Renames a file in the list of files.
- [MNDecodeFromList](#)  
Decodes a file from the list of files using "UUE", "Mime-Base64/PT/QP", "XXE", "BinHex".
- [DeleteSubString](#)  
Deletes all occurrences of one string, in another string.
- [LSetIn](#)  
Inserts a string (to the left) into a sub-string delimited by a separator in a given string.
- [RSetIn](#)  
Inserts a string (to the right) into a sub-string delimited by a separator in a given string.
- [LFill](#)  
Pads a string to the left, to a know length.
- [RFill](#)  
Pads a string to the right, to a know length.

Searches for known strings, and replaces them with another string.

[StringReplace](#)

Compares a string with a set of strings delimited by a separator.

[MatchTable](#)

Finds the position of the first occurrence of one string within another (like VB Instr function).

[InStr](#)

Adds two complex.

[CpxAdd](#)

Subtracts two complex.

[CpxSub](#)

Multiplies two complex.

[CpxMul](#)

Divides two complex.

[CpxDiv](#)

Computes the conjugue of a complex.

[CpxConjugue](#)

Computes the modulus of a complex.

[CpxModulus](#)

Computes the argument of a complex.

[CpxArgument](#)

Computes the power N of a complex.

[CpxPowerN](#)

Computes the square root of a complex.

[CpxSqrt](#)

Computes the root N of a double number.

[RootN](#)

4.28 *no new features.*

4.25 Now, there is a new little dll, called T2W32MPT.DLL, which WORK ONLY under Windows 95 and handle :  
[cModule](#), [cProcess](#) and [cThread](#).

Locks/unlocks the keyboard for any applications.

[LockKeyboard](#)

Locks/unlocks the mouse (any events : left, middle and right click) for any applications.

[LockMouse](#)

4.21 Now, there is a version of TIME TO WIN 32-Bit for Pentium Pro : T2W32PRO.DLL

Retrieves the argument list from a shortcut file. \_\_\_\_\_

[ShortcutFileGetArguments](#)

Retrieves the description from a shortcut file. \_\_\_\_\_

[ShortcutFileGetDescription](#)

Retrieves the path from a shortcut file.

[ShortcutFileGetPath](#)

Retrieves the path (in format 8.3) from a shortcut file.

[ShortcutFileGetPath83](#)

Retrieves the working directory from a shortcut file.

[ShortcutFileGetWorkingDir](#)

Generic function to retrieve the above parameter from a shortcut file.

[ShortcutFileGetInfo](#)

Modifies the argument list in a shortcut file.

[ShortcutFileSetArguments](#)

Modifies the description in a shortcut file.

[ShortcutFileSetDescription](#)

Modifies the path in a shortcut file.

[ShortcutFileSetPath](#)

Modifies the working directory in a shortcut file. \_\_\_\_\_

[ShortcutFileSetWorkingDir](#)

Generic function to modify the above parameter in a shortcut file. \_\_\_\_\_

[ShortcutFileSetInfo](#)

Copy one file to an another file with the same file attribute.

[FileForceCopy](#)

- 4.16 Search for file(s) with attribute and save the result in a file.  
[SearchFileAttrib](#)  
 Search for file(s) with attribute and show the result in a standard list box.  
[ListSearchFileAttrib](#)  
 Search for file(s) with attribute and show the result in a standard combo box.  
[ComboSearchFileAttrib](#)
- 4.12 *no new features.*
- 4.10 *no new features.*
- 4.04 Disables/enables CTRL+ALT+DEL, ALT+TAB and CTRL+ESC under Win95.  
[MultitasksKeys](#)  
 Compute a Long from all date-hour parts.  
[DateHourToLong](#)  
 Decompose a Long date-hour into these components.  
[LongToDateHour](#)  
 Convert a Single number into the Microsoft Binary Format representation.  
[MKSMBF](#)  
 Convert a string containing the Microsoft Binary Format representation of a Single into a Single.  
[CVSMBF](#)
- 4.03 Search for dir(s) and save the result in a file.  
[SearchDir](#)  
 Search for dir(s) using pattern matching and show the result in a standard list box.  
[ListSearchDir](#)  
 Search for dir(s) using pattern matching and show the result in a standard combo box.  
[ComboSearchDir](#)  
 Perform conversion between date, time, hour, minute, hundred. [Convert](#)  
 Calculates an Integer from a date.  
[DateToInt](#)  
 Decomposes an Integer into date parts.  
[IntToDate](#)  
 Move files from one directory to an another directory/disk.  
[FilesMove](#)  
 Shift a Byte array to Left and set the last element to a value.  
[ShiftLeftB](#)  
 Shift a Double array to Left and set the last element to a value.  
[ShiftLeftD](#)  
 Shift an Integer array to Left and set the last element to a value.  
[ShiftLeftI](#)  
 Shift a Long array to Left and set the last element to a value.  
[ShiftLeftL](#)  
 Shift a Single array to Left and set the last element to a value.  
[ShiftLeftS](#)  
 Shift a Byte array to Right and set the first element to a value.  
[ShiftRightB](#)  
 Shift a Double array to Right and set the first element to a value.  
[ShiftRightD](#)  
 Shift an Integer array to Right and set the first element to a value.  
[ShiftRightI](#)  
 Shift a Long array to Right and set the first element to a value.  
[ShiftRightL](#)  
 Shift a Single array to Right and set the first element to a value.  
[ShiftRightS](#)
- 4.00 Sort a Type'd array which can contain :  
 one or more elements of any data type, array, or a previously defined user-defined type.\_\_\_\_  


---

[SortTypedArray](#)
- 3.50 Copy files from one directory to an another directory/disk.  
[FilesCopy](#)

- Search if a given pattern can be found in a given string (case-sensitive or not).  
[PatternMatchS](#)
- Search if a given pattern can be found in a given string (case-sensitive or not).  
[PatternExtMatchS](#)
- 3.40 Merges all files in an array in one.  
[FileMergeExt](#)  
Retrieve the EXPLORER (95/NT4.0x) display name of a file.  
[GetFileDisplayName](#)  
Retrieve the EXPLORER (95/NT4.0x) type name of a file.  
[GetFileType](#)  
Adds a document to the shell's list of recently used documents.  
[ExplorerAddToRecentDocs](#)  
Clears all documents from the shell's list of recently used documents.  
[ExplorerClearAllRecentDocs](#)
- 3.30 Returns a key setting value from an application's Windows registry entry for any key handle.  
[GetRegistryExt](#)  
Saves or creates an application entry in the Windows registry entry for any key handle.  
[PutRegistryExt](#)  
Deletes a section or key setting from the Windows registry entry for any key handle.  
[KillRegistryExt](#)  
Retrieve in one routine all date & time informations (creation, last access, last write) for a file.  
[GetFileDateTime](#)  
Set in one routine all date & time informations (creation, last access, last write) for a file.  
[SetFileDateTime](#)
- 3.22 *no new features.*
- 3.20 Finding the maximum value in an array, not equal to a specified value.  
[MaxNotXD](#), [MaxNotXI](#), [MaxNotXL](#), [MaxNotXS](#)  
Finding the minimum value in an array, not equal to a specified value.  
[MinNotXD](#), [MinNotXI](#), [MinNotXL](#), [MinNotXS](#)  
Cut a file in two parts (creation of two files).  
[CutFile](#)  
Split a file into target files based on a part size.  
[SplitFile](#)  
Insert different block of char in a given string separated by '~' (It can handle empty insert string).  
[InsertBlocksExt](#)
- 3.10 Display a bitmap as splash screen.  
[DisplaySplash](#)  
Destroy a bitmap (splash screen) displayed by DisplaySplash.  
[DestroySplash](#)
- 3.07 *no new features.*
- 3.06 New command MB\_NO\_BUTTONS in [LngBoxMsg](#) and in [LngMsgBox](#).  
Norwegian language supported (LNG\_NORVEGIAN) in [LngBoxMsg](#) and in [LngMsgBox](#).  
Convert a Z9 string to an ascii string.  
[FromZ9](#)  
Convert a ascii string to Z9 string.  
[ToZ9](#)  
Return if a given bit in a given BYTE is Set or Reset.  
[GetBitB](#)  
Return if a given bit in a given DOUBLE is Set or Reset.  
[GetBitD](#)  
Return if a given bit in a given INTEGER is Set or Reset.  
[GetBitI](#)  
Return if a given bit in a given LONG is Set or Reset.  
[GetBitL](#)

Return if a given bit in a given SINGLE is Set or Reset.

[GetBitS](#)

Set a given bit in a given BYTE to Set state or Reset state.

[SetBitB](#)

Set a given bit in a given DOUBLE to Set state or Reset state.

[SetBitD](#)

Set a given bit in a given INTEGER to Set state or Reset state.

[SetBitI](#)

Set a given bit in a given LONG to Set state or Reset state.

[SetBitL](#)

Set a given bit in a given SINGLE to Set state or Reset state.

[SetBitS](#)

Convert an integer value into a binary string.

[I2B](#)

Convert a long value into a binary string.

[L2B](#)

Calculate the mod-10 of the given string.

[Mod10](#)

Calculate the reverse mod-10 of the given string.

[Mod10R](#)

Calculate the mod-11 of the given string.

[Mod11](#)

Calculate the reverse mod-11 of the given string.

[Mod11R](#)

Calculate the mod-1.3.7 of the given string.

[Mod137](#)

Calculate the reverse mod-1.3.7 of the given string.

[Mod137R](#)

3.05 *no new features.*

3.04 Encode a string with a password using the RUBY algorithm (7 modes).

[RUBYencrypt](#)

Decode a string with a password using the RUBY algorithm (7 modes).

[RUBYdecrypt](#)

Copy one file to an another file but with RUBY algorithm (7 modes).

[RUBYencryptFile](#)

Copy one file to an another file but with RUBY algorithm (7 modes).

[RUBYdecryptFile](#)

Encode a string with a password using the Diamond Encryption Algorithm (4 modes).

[DIAMONDencrypt](#)

Decode a string with a password using the Diamond Encryption Algorithm (4 modes).

[DIAMONDdecrypt](#)

Copy one file to an another file but with Diamond Encryption Algorithm (4 modes).

[DIAMONDencryptFile](#)

Copy one file to an another file but with Diamond Encryption Algorithm (4 modes).

[DIAMONDdecryptFile](#)

Copy one file to an another file but with U.S. Data Encryption Standard encryption.

[DESencryptFile](#)

Copy one file to an another file but with U.S. Data Encryption Standard decryption.

[DESdecryptFile](#)

Copy one file to an another file but with the International Data Encryption Algorithm encryption.

[IDEAencryptFile](#)

Copy one file to an another file but with the International Data Encryption Algorithm decryption.

[IDEAdencryptFile](#)

3.03 Compress a file into a compressed format using ASH arithmetic compression.

[ASHFileCompress](#)

Expand a compressed file into a normal format using ASH arithmetic compression.

[ASHFileExpand](#)

- 3.02 Compress a string into a compressed format using GZIP compression method.  
[GZIPStringCompress](#)  
Expand a compressed string into a normal format using GZIP compression method.  
[GZIPStringExpand](#)
- 3.01 Compress a file into a compressed format using GZIP compression method.  
[GZIPFileCompress](#)  
Expand a compressed file into a normal format using GZIP compression method.  
[GZIPFileExpand](#)
- 3.00 Encode a string with a password using the U.S. Data Encryption Standard cipher.  
[DESEncrypt](#)  
Decode a string with a password using the U.S. Data Encryption Standard cipher.  
[DESDecrypt](#)  
Encode a string with a password using the International Data Encryption Algorithm cipher.  
[IDEAEncrypt](#)  
Decode a string with a password using the International Data Encryption Algorithm cipher.  
[IDEADecrypt](#)  
Compress a file into a compressed format using arithmetic compression.  
[LZARCompress](#)  
Expand a compressed file into a normal format using arithmetic compression.  
[LZARExpand](#)
- 2.52 Enumerate all open files and/or all unmovable open files.  
[EnumOpenFiles](#)
- 2.51 Now, T2WIN-32.DLL can be registered directly by using the Register button. This method is usefully for Internet user.
- 2.50 Now, T2WIN-32.DLL is compatible with Windows NT 3.51.  
New help file T2WINALL.HLP (this file).
- 2.10 Reads the offset of each line from an ASCII file (CR/LF line terminated) in an array.  
[FProcessAsciiFile](#)  
Moves the file pointer to the beginning of the specified line in an ASCII file (CR/LF line terminated).  
[FGotoRecord](#)  
Calculate the z value of an additional point from four points.  
[3DWeightAverage](#)
- 2.00 Enumerate all pendings jobs on a printer.  
[EnumPrinterJobs](#)
- 1.60 TileBitmapOnWindow tile a bitmap (DDB or DIB format) on a window.  
[TileBitmapOnWindow](#)
- 1.42 Save the screen (entire desktop) in a file (DIB format).  
[DIBSaveScreen](#)  
Save a window in a file (DIB format).  
[DIBSaveWindow](#)  
Install a hook keyboard to save the screen or the active window in a file (DIB format).  
[InstallHookKeyboard](#)
- 1.33 Display an icon for an application in the tray of the task bar.  
[TaskBarAddIcon](#)  
Delete the tray icon from an application in the task bar.  
[TaskBarDeleteIcon](#)  
Modify an icon for an application in the tray of the task bar.  
[TaskBarModifyIcon](#)
- 1.24 Reads the media ID (serial number, volume label, ...) from a disk.  
[DOSGetMediaID](#)



- Changes the media ID (serial number, volume label, ...) to a disk.  
[DOSSetMediaID](#)
- 1.20 Returns a key setting value from an application's Windows registry entry.  
[GetRegistry](#)  
Saves or creates an application entry in the Windows registry entry.  
[PutRegistry](#)  
Deletes a section or key setting from the Windows registry entry.  
[KillRegistry](#)
- 1.11 *no new features.*
- 1.10 *no new features.*
- 1.06 Search for file(s) and save the result in a file.  
[SearchFile](#)  
Search for file(s) and show the result in a standard list box.  
[ListSearchFile](#)  
Search for file(s) and show the result in a standard combo box.  
[ComboSearchFile](#)  
Crypt a file with password.  
[FileCrypt](#)  
Crypt a string with password.  
[Crypt](#)  
Calculate a registration key (method 1).  
[RegistrationKey](#)  
Calculate a registration key (method 2).  
[RegistrationKey2](#)  
Calculate a registration key (method 3).  
[RegistrationKey3](#)  
Perform a file copy and show a progress bar in a standard control or form.  
[PBFileCopy](#)  
Perform a file copy and show a dialog box with progress bar on desktop.  
[DBFileCopy](#)  
UUencode/UUdecode a file.  
[FileUUCP](#)
- 1.02 Set tab spacing in a standard list box.  
[ListSetTabs](#)  
Load the contents of a directory in a standard list box. [ListFiles](#)  
Load the contents of a directory in a standard combo box.  
[ComboFiles](#)
- 1.00 Initial release of the 'TIME TO WIN (32-Bit)' Dynamic Link Library for Visual Basic 4.0 (32-Bit Edition under Windows 95/NT).

# TIME TO WIN for MSOffice 95 : New features

See also : [Revision History](#)

Version	Comments
2.02	<p>New command MB_NO_BUTTONS in <a href="#">LngBoxMsg</a> and in <a href="#">LngMsgBox</a>. Norwegian language supported (LNG_NORVEGIAN) in <a href="#">LngBoxMsg</a> and in <a href="#">LngMsgBox</a>. Convert a Z9 string to an ascii string. <a href="#">FromZ9</a> Convert a ascii string to Z9 string. <a href="#">ToZ9</a> Return if a given bit in a given BYTE is Set or Reset. <a href="#">GetBitB</a> Return if a given bit in a given DOUBLE is Set or Reset. <a href="#">GetBitD</a> Return if a given bit in a given INTEGER is Set or Reset. <a href="#">GetBitI</a> Return if a given bit in a given LONG is Set or Reset. <a href="#">GetBitL</a> Return if a given bit in a given SINGLE is Set or Reset. <a href="#">GetBitS</a> Set a given bit in a given BYTE to Set state or Reset state. <a href="#">SetBitB</a> Set a given bit in a given DOUBLE to Set state or Reset state. <a href="#">SetBitD</a> Set a given bit in a given INTEGER to Set state or Reset state. <a href="#">SetBitI</a> Set a given bit in a given LONG to Set state or Reset state. <a href="#">SetBitL</a> Set a given bit in a given SINGLE to Set state or Reset state. <a href="#">SetBitS</a> Convert an integer value into a binary string. <a href="#">I2B</a> Convert a long value into a binary string. <a href="#">L2B</a> Calculate the mod-10 of the given string. <a href="#">Mod10</a> Calculate the reverse mod-10 of the given string. <a href="#">Mod10R</a> Calculate the mod-11 of the given string. <a href="#">Mod11</a> Calculate the reverse mod-11 of the given string. <a href="#">Mod11R</a> Calculate the mod-1.3.7 of the given string. <a href="#">Mod137</a> Calculate the reverse mod-1.3.7 of the given string. <a href="#">Mod137R</a></p>
2.00	<p>Encode a string with a password using the RUBY algorithm (7 modes). <a href="#">RUBYencrypt</a> Decode a string with a password using the RUBY algorithm (7 modes). <a href="#">RUBYdecrypt</a> Copy one file to an another file but with RUBY algorithm (7 modes). <a href="#">RUBYencryptFile</a> Copy one file to an another file but with RUBY algorithm (7 modes). <a href="#">RUBYdecryptFile</a> Compress a string into a compressed format using GZIP compression method. <a href="#">GZIPStringCompress</a></p>

Expand a compressed string into a normal format using GZIP compression method.

[GZIPStringExpand](#)

Compress a file into a compressed format using GZIP compression method.

[GZIPFileCompress](#)

Expand a compressed file into a normal format using GZIP compression method.

[GZIPFileExpand](#)

1.00 Initial release of the 'TIME TO WIN for MSOffice 95' Dynamic Link Library.

# TIME TO WIN for VB 3.0 : Revision history

See also : [New Features](#)

Version	Comments
9.59	Includes new version of ZLIB (v1.12). Correction in <a href="#">PutIni</a> to handle the deleting of a section (szItem = "") or an item (szDefault = ""). Corrects a problem (buffer initialization, causes a GPF in compiled program) with <a href="#">StringSAR</a> . Modifies the buffer for uncompressed string in <a href="#">GZIPStringExpand</a> .
9.52	<i>no revision.</i>
9.45	Correction of a GPF in <a href="#">FileSearchAndReplace</a> (function has been fully rewritten).
9.44	<a href="#">FileForceCopy</a> now accepts a new parameter 'ForceOrNot' to perform the copy. <a href="#">FileUUCP</a> now accepts a new parameter 'HeaderOrNot' to perform the uuencode/uudecode with header/footer or not.
9.43	<i>no revision.</i>
9.34	<i>no revision.</i>
9.25	The following routine returns a Long data type instead of a Integer data type (especially for Network structure) : <a href="#">RcsCountFileDir</a> Correction of matching file with "*" in the following routines (all files without extension were not treated) : <a href="#">RcsCountFileDir</a> , <a href="#">RcsFilesSize</a> , <a href="#">RcsFilesSizeOnDisk</a> , <a href="#">RcsFilesSlack</a> , Correction of bad calculation for <a href="#">RcsFilesSizeOnDisk</a> , <a href="#">RcsFilesSlack</a> ,
9.23	Correct a problem with <a href="#">cGetFileVersion</a> and <a href="#">cGetFileVersionInfo</a> (for some files, not enough memory to handle information).
9.22	<i>no revision.</i>
9.21	Correct a problem in <a href="#">cFileSearchAndReplace</a> . The function worked not good in some cases.
9.20	<i>no revision.</i>
9.10	<i>no revision.</i>
9.08	Increase the length of the returned string in <a href="#">StringSAR</a> to 25 times the length of the search string in place of 4 times.
9.07	Now, <a href="#">FilesSize</a> / <a href="#">FilesSizeOnDisk</a> / <a href="#">FilesSlack</a> can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG). Now, <a href="#">RcsFilesSize</a> / <a href="#">RcsFilesSizeOnDisk</a> / <a href="#">RcsFilesSlack</a> can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG). Now, <a href="#">GetDiskFree</a> / <a href="#">GetDiskSpace</a> / <a href="#">GetDiskUsed</a> / <a href="#">GetDiskClusterSize</a> can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG).
9.06	Correct a problem in <a href="#">LngMsgBox</a> and <a href="#">LngBoxMsg</a> : default button 1 is not selected when 3 buttons are displayed.
9.05	Add the registration of my products on Internet with KAGI SHAREWARE, use REGISTER.EXE for registration.
9.04	Modification of the memory allocation for some routines.

- 9.01 *no revision.*
- 9.00 *no revision.*
- 8.08 Correct a problem with cDOSSetVolumeLabel. The function can't delete the volume label.
- 7.01 *no revision.*
- 7.00 Correct a problem when accessing a Sheet other than the first in ClearSheet, ClearRow, ClearCol in Disk Array routines  
Multiple Disk Array routines, Huge Memory Arrays.
- 6.01 *no revision.*
- 6.00 Increase of line length from 2304 to 4096 and changes some internal functionalities in cFileSearchAndReplace.
- 5.29 Adds RS\_MENU for language's management.  
Adds A\_NORMAL\_ARCHIVE and A\_ALL attributes.
- 5.20 Correct a GPF problem with cGetCurrentDrive.
- 5.10 *no revision.*
- 5.02 Correct a problem with cGetVersion. The version returned don't take care of minor version.
- 5.00 *no revision.*
- 4.57 *no revision.*
- 4.50 *no revision.*
- 4.00 *no revision.*
- 3.52 Some little internal change.
- 3.51 Correct a problem with cIsFilenameValid if the filename begins with '..\'
- 3.50 *no revision.*
- 3.00 Changes the functionality of cProperName2.
- 2.05 *no revision.*
- 2.00 *no revision.*
- 1.60 *no revision.*
- 1.52 *no revision.*
- 1.50 Correct a problem with cGetSectionItems (other .INI file than WIN.INI are not processed).
- 1.42 Adds a new value for Encrypt/Decrypt (ENCRYPT\_LEVEL\_4) (see cEncrypt, cDecrypt, cFileEncrypt, cFileDecrypt).
- 1.36 *no revision.*
- 1.33 Corrects a problem if you pass a bad open mode (not OPEN\_MODE\_BINARY or OPEN\_MODE\_TEXT) in cFileCRC32.  
Corrects a problem in cLngMsgbox, cLngBoxMsg when using MB\_MESSAGE\_LEFT (default).  
Corrects a problem in the UNREGISTERED version when the 'Shareware License Agreement' box is displayed (VB causes a GPF).

- 1.30 Adds a new item (.nlstyped) in the description of a big sized array to specify the init of a type'd variable, see cDACreate.  
Adds a new item (RS\_TAG) to handle .Tag property in cSaveCtlLanguage, cReadCtlLanguage.  
Adds missing help topic for cScrollL and cScrollR.  
Speed up the encrypt/decrypt algorithm by 20% (cEncrypt, cDecrypt, cFileEncrypt, cFileDecrypt).
- Corrects a problem when accessing a sheet in a big sized array. This problem has no effect on a single sheet array.  
Changes allocation of temporary memory to avoid/correct some problems in some strings routines (see Affected routines).  
Corrects a problem when creating a big sized array of type'd variable in disk. This problem has not occurred all times.
- 1.28 Adds TimeOut functionality (from 2 to 30 seconds by step of 2 seconds) and display TimeOut to cLngMsgBox, cLngBoxMsg.  
Adds the detection of CD-ROM drive (with MSCDEX driver) in cGetDriveType.  
Adds some errors code and network drive validation for clsFilenameValid.  
cKillFile, cKillFileAll, now, returns TRUE if the filename not exists.  
Now, all files, from the executable demo, are included. (Be indulgent, no comments are in the demo).
- 1.22 *no revision.*
- 1.21 Removes the need of passing the letter drive in cFilesSizeOnDisk and cFilesSlack by using cSplitPath.  
Now, cFilesSize, cFilesSizeOnDisk, cFilesSlack and cFilesInDirectory take care of the file attribute (Read-Only, System, Hidden).  
Now, cAllSubDirectories can handle 700 directories (in place of 300) of maximum 70 chars long each.  
Changes cSplitPath from sub to function to check if the filename is valid.  
Improves cFileCopy, cFileFilter, cFileFilterNot, cCmpFileContents speed performance.  
Improves cFileEncrypt, cFileDecrypt, cFileCompressTab, cFileExpandTab speed performance.  
Improves cFileCRC32 speed performance.  
Changes some errors number returned for standardization (see Returned Errors).
- Corrects a problem with clsFilenameValid (some valid filename was not check als valid).  
Corrects a problem with cGetFileVersion (sometimes GPF when accessing '\StringFileInfo\04090000').  
Corrects a problem with cGetFileVersionInfo (sometimes returns a chr\$(0)).
- 1.14 Modify the encrypt/decrypt algorithm. (cEncrypt, cDecrypt, cFileEncrypt, cFileDecrypt).
- 1.07 Add a new protection algorithm.  
Add modal dialog box for unregistered version in place of message box.
- 1.00 Initial release of the 'TIME TO WIN' dynamic link library for VB 3.0.

Select the following product :

TIME TO WIN for VB 3.0

TIME TO WIN for VB 4.0 (16-Bit)

TIME TO WIN for VB 4.0 (32-Bit), VB 5.0 and VBA 5.0

TIME TO WIN for MSOffice 95

# TIME TO WIN for VB 4.0 (16-Bit) : Revision history

See also : [New Features](#)

Version	Comments
9.59	Includes new version of ZLIB (v1.12). Correction in <a href="#">PutIni</a> to handle the deleting of a section (szItem = "") or an item (szDefault = ""). Corrects a problem (buffer initialization, causes a GPF in compiled program) with <a href="#">StringSAR</a> . Modifies the buffer for uncompressed string in <a href="#">GZIPStringExpand</a> .
9.52	<i>no revision.</i>
9.45	Correction of a GPF in <a href="#">FileSearchAndReplace</a> (function has been fully rewritten).
9.44	<a href="#">FileForceCopy</a> now accepts a new parameter 'ForceOrNot' to perform the copy. <a href="#">FileUUCP</a> now accepts a new parameter 'HeaderOrNot' to perform the uuencode/uudecode with header/footer or not.
9.43	<i>no revision.</i>
9.34	<i>no revision.</i>
9.25	The following routine returns a Long data type instead of a Integer data type (especially for Network structure) : <a href="#">RcsCountFileDir</a> Correction of matching file with "*" in the following routines (all files without extension were not treated) : <a href="#">RcsCountFileDir</a> , <a href="#">RcsFilesSize</a> , <a href="#">RcsFilesSizeOnDisk</a> , <a href="#">RcsFilesSlack</a> , Correction of bad calculation for <a href="#">RcsFilesSizeOnDisk</a> , <a href="#">RcsFilesSlack</a> ,
9.23	Correct a problem with <a href="#">cGetFileVersion</a> and <a href="#">cGetFileVersionInfo</a> (for some files, not enough memory to handle information).
9.22	<i>no revision.</i>
9.21	Correct a problem in <a href="#">cFileSearchAndReplace</a> . The function worked not good in some cases.
9.20	<i>no revision.</i>
9.10	<i>no revision.</i>
9.08	Increase the length of the returned string in <a href="#">StringSAR</a> to 25 times the length of the search string in place of 6 times.
9.07	Now, <a href="#">FilesSize</a> / <a href="#">FilesSizeOnDisk</a> / <a href="#">FilesSlack</a> can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG). Now, <a href="#">RcsFilesSize</a> / <a href="#">RcsFilesSizeOnDisk</a> / <a href="#">RcsFilesSlack</a> can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG). Now, <a href="#">GetDiskFree</a> / <a href="#">GetDiskSpace</a> / <a href="#">GetDiskUsed</a> / <a href="#">GetDiskClusterSize</a> can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG).
9.06	Correct a problem in <a href="#">LngMsgBox</a> and <a href="#">LngBoxMsg</a> : default button 1 is not selected when 3 buttons are displayed.
9.05	Add the registration of my products on Internet with KAGI SHAREWARE, use REGISTER.EXE for registration.
9.01	<i>no revision.</i>



9.00 *no revision.*

8.08 Correct a problem with cDOSSetVolumeLabel. The function can't delete the volume label.

7.07 The following functions has been removed :

cReadMnuLanguage has been included in the functions cReadCtlLanguage,  
cReadCtlLanguageExt

cSaveMnuLanguage has been included in the functions cSaveCtlLanguage, cSaveCtlLanguageExt

7.00 Initial release of the 'TIME TO WIN (16-Bit)' Dynamic Link Library for Visual Basic 4.0 (16-Bit Edition).

## Compression : Overview

[ASHFileCompress](#)

compress a file into a compressed format using ASH arithmetic compression.

[ASHFileExpand](#)

expand a compressed file into a normal format using ASH arithmetic compression.

[FileCompress](#)

compress a file into a compressed format.

[FileExpand](#)

expand a compressed file into a normal format.

[GZIPFileCompress](#)

compress a file into a compressed format using GZIP compression method.

[GZIPFileExpand](#)

expand a compressed file into a normal format using GZIP compression method.

[GZIPStringCompress](#)

compress a string into a compressed format using GZIP compression method.

[GZIPStringExpand](#)

expand a compressed string into a normal format using GZIP compression method.

[LZARlcompress](#)

compress a file into a compressed format using arithmetic compression.

[LZARlexpand](#)

expand a compressed file into a normal format using arithmetic compression.

[StringCompress](#)

compress a string into a compressed format.

[StringExpand](#)

expand a compressed string into a normal format.

# TIME TO WIN for VB 4.0 (32-Bit), VB 5.0 and VBA 5.0 :

## Revision history

See also : [New Features](#)

Version	Comments
5.43	The demo doesn't use THREEED32.OCX
5.34	Corrects a problem (buffer initialization, causes a GPF in compiled program) with <a href="#">StringSAR</a> . Modifies the buffer for uncompressed string in <a href="#">GZIPStringExpand</a> .
5.25	Add functionalities to <a href="#">WrapLine</a> to handle CR+LF.
5.20	Price of "TIME TO WIN 32-Bit" and "update to TIME TO WIN 32-Bit" has been updated (see <a href="#">Registration</a> ) Includes new registration method (usage of the registry in place of a license file). Includes new version of ZLIB (v1.12). <a href="#">GetDiskX</a> has been adapted to take care of Win95 OSR2.
5.11	Correction in <a href="#">FileStatistics</a> for file size > 32768 bytes. Includes new version of ZLIB (v1.11).
5.09	Includes new version of ZLIB (v1.10).
5.02	Correction of a GPF in <a href="#">GZIPFileCompress</a> under Windows NT.
5.01	<i>no revision.</i>
5.00	Correction in <a href="#">PutIni</a> to handle the deleting of a section (szItem = "") or an item (szDefault = "").
4.52	Adds 2 new parameters in <a href="#">RmvDupX</a> to remove duplicate values in an array (Integer, Long, Single, Double)
4.43	Now, <a href="#">cSortX</a> can handle array with two dimensions. Now, <a href="#">cReverseSortX</a> can handle array with two dimensions.
4.34	Correction of a GPF in <a href="#">FileSearchAndReplace</a> (function has been fully rewritten).
4.28	<a href="#">LockMouse</a> now accepts a new parameter called to perform the choice of the click to hook (left and/or middle and/or right). <a href="#">FileForceCopy</a> now accepts a new parameter called 'ForceOrNot' to perform the copy. <a href="#">FileUUCP</a> now accepts a new parameter called 'HeaderOrNot' to perform the uuencode/uudecode with header/footer or not.
4.25	<i>no revision.</i>
4.21	<i>no revision.</i>
4.16	<i>no revision.</i>
4.12	Correction of bad calculation for <a href="#">RcsFilesSizeOnDisk</a> , <a href="#">RcsFilesSlack</a> ,
4.10	The following routine returns a Long data type instead of a Integer data type (especially for Network structure) : <a href="#">RcsCountFileDir</a> Correction of matching file with "*" in the following routines (all files without extension were not treated) : <a href="#">RcsCountFileDir</a> , <a href="#">RcsFilesSize</a> , <a href="#">RcsFilesSizeOnDisk</a> , <a href="#">RcsFilesSlack</a> , <a href="#">SearchFile</a> , <a href="#">ComboSearchFile</a> , <a href="#">ListSearchFile</a> , <a href="#">SearchDir</a> , <a href="#">ComboSearchDir</a> , <a href="#">ListSearchDir</a> <a href="#">KillDirs</a> , <a href="#">KillDirFilesAll</a>

- 4.05 Correct a problem in cFileSearchAndReplace. The function worked not good in some cases.
- 4.04 *no revision.*
- 4.03 *no revision.*
- 4.00 *no revision.*
- 3.50 cKillDirs has been fully rewritten. Now, There is no limitation.  
Correction of matching file (now internal searching is case insensitive) in the following routines :  
RcsCountFileDir, RcsFilesSize, RcsFilesSizeOnDisk, RcsFilesSlack,  
SearchFile, ComboSearchFile, ListSearchFile, SearchDir, ComboSearchDir, ListSearchDir  
KillDirs, KillDirFilesAll
- 3.40 *no revision.*
- 3.30 *no revision.*
- 3.22 Correct a problem in DESencryptFile when the size of the source file is a multiple of 8.  
Correct a problem in DIAMONDencryptFile when the size of the source file is a multiple of 16 (full mode) or 8 (lite mode).  
Correct a problem in IDEAencryptFile when the size of the source file is a multiple of 8.  
Correct a problem in DIAMONDencrypt, DIAMONDencryptFile (internal error especially under WinNT 4.0).  
Now, FileDateCreated / FileLastDateAccess / FileLastDateModified take care of the format of the date in regional parameters.  
Now, FileTimeCreated / FileLastTimeAccess / FileLastTimeModified take care of the format of the hour in regional parameters.
- 3.20 Increase the length of the returned string in StringSAR to 25 times the length of the search string in place of 6 times.  
Correct some problems with InsertByMask, InsertChars, FilterBlocks, FilterChars, FilterFirstChars, FilterNotChars,  
GetInPart, GetInPartR, RemoveOneChar, RemoveBlockChar, StringSAR, especially when out of limits or some empty strings.
- 3.10 *no revision.*
- 3.07 Include a Pentium Processor (Compatible) version of T2WIN-32.DLL.
- 3.06 Correct a problem in LngMsgBox and LngBoxMsg : default button 1 is not selected when 3 buttons are displayed.
- 3.05 Add the registration of my products on Internet with KAGI SHAREWARE, use REGISTER.EXE for registration.  
Change of the maximum compress ratio from 1:3 to 1:10 in StringCompression / StringExpand  
Now, FilesSize / FilesSizeOnDisk / FilesSlack can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG).  
Now, RcsFilesSize / RcsFilesSizeOnDisk / RcsFilesSlack can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG).  
Now, GetDiskFree / GetDiskSpace / GetDiskUsed / GetDiskClusterSize can handle hard disk greater than 2Gb (return value is a DOUBLE in place of LONG).
- 3.04 *no revision.*
- 3.03 Now, DESencrypt / DESdecrypt can handle a string with any size (not only a multiple of 8 chars).  
Now, IDEAencrypt / IDEAdencrypt can handle a string with any size (not only a multiple of 8 chars).
- 3.02 *no revision.*
- 3.01 *no revision.*

- 3.00 *no revision.*
- 2.52 *no revision.*
- 2.51 Now, T2WIN-32.DLL can be registered directly by using the Register button. This method is usefully for Internet user.
- 2.50 Now, T2WIN-32.DLL is compatible with Windows NT 3.51. To do this, I've removed the cModule, cProcess, cThread functions.  
New help file T2WINALL.HLP (this file).
- 2.10 *no revision.*
- 2.00 *no revision.*
- 1.60 *no revision.*
- 1.42 *no revision.*
- 1.33 Display some TimeOuts when 'TIME TO WIN (32-Bit) is not registered.  
Display an icon (and a tooltip) in the tray on the task bar when 'TIME TO WIN (32-Bit) is used in design mode.
- 1.24 The icons used in the International Message Box and International Input Box are now the icons used by Windows 95.
- 1.20 *no revision.*
- 1.11 Correct a problem with cChDir and cChDrive when the parameter is a zero-length string.
- 1.10 Suppression of the expiration date.  
Add a logo in the UNregistered version.  
Add a module (\_T2WREG.EXE) for registering thru Internet.  
Some improvements.
- 1.06 Correct a problem in cFileCRC32.
- 1.02 *no revision.*
- 1.00 Initial release of the 'TIME TO WIN (32-Bit)' Dynamic Link Library for Visual Basic 4.0 (32-Bit Edition under Windows 95/NT).

# ArrayStringOnDisk

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ArrayStringOnDisk](#) put/get full variable string array (one dimension) on/from disk ascii file.

## Declare Syntax :

Declare Function cArrayStringOnDisk Lib "[time2win.dll](#)" (ByVal File As String, Array() As Any, ByVal GetPut As Integer, rRecords As Long) As Long

## Call Syntax :

test& = cArrayStringOnDisk(File\$, Array(), GetPut%, rRecords&)

## Where :

File\$	is the file to use.
Array()	is the variable array string with one dimension.
GetPut%	PUT_ARRAY_ON_DISK to put the array on disk, GET_ARRAY_ON_DISK to get the array from disk.
rRecords&	the returned number of records.
test&	>=0 is the returned length of the file, < 0 is an error occurs (error n° is the negative value of all DA_x values, see <a href="#">Constants and</a>

[Types declaration](#)).

## Comments :

This function can handle only a variable type'd string derived from tagVARSTRING (see below).

Don't forget that if you use the 'ReDim' statement at the procedure level without have declared the array als Global, you must initialize the array before using this function (see below). You must initialize the array with enough space to handle the size of the file This is due to a VB limitation.

When reading, if the number of lines in the file is below the size of the array, the remain items in the array are set to EMPTY string. The CR + LF are not included in the array.

When writing, all lines are appended with CR + LF.

This function can handle huge array (greater than 65535 bytes) (see the example below).

```
Type tagVARSTRING
    Contents           As String
End Type
```

## Examples :

```
ReDim AD(-999 To 1000)           As tagVARSTRING
Dim i                             As Long
Dim r                             As Long

For i = -999 To 1000
    AD(i).Contents = Space$(256)
Next i

Debug.Print cArrayStringOnDisk("c:\autoexec.bat", AD(), GET_ARRAY_ON_DISK, r)

Debug.Print cArrayStringOnDisk("c:\autoexec.tab", AD(), PUT_ARRAY_ON_DISK, r)
```

```
For i = -999 To 1000
    AD(i).Contents = Space$(256)
Next i

Debug.Print cArrayStringOnDisk("c:\autoexec.tab", AD(), GET_ARRAY_ON_DISK, r)

Debug.Print AD(-999).Contents
Debug.Print AD(-998).Contents
```

**See also :** Disk Array routines, [cArrayOnDisk](#)

# TIME TO WIN for MSOffice 95 : Revision history

**See also :** [New Features](#)

<b>Version</b>	<b>Comments</b>
2.02	Correct a problem in <a href="#">LngMsgBox</a> and <a href="#">LngBoxMsg</a> : default button 1 is not selected when 3 buttons are displayed.
2.01	Add the registration of my products on Internet with KAGI SHAREWARE, use REGISTER.EXE for registration.
2.00	<i>no revision.</i>
1.00	Initial release of the 'TIME TO WIN for MSOffice 95' Dynamic Link Library.



# EnumPrinterJobs

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[EnumPrinterJobs](#) enumerates all pending jobs on a printer.

## Declare Syntax :

Declare Function cEnumPrinterJobs Lib "[time2win.dll](#)" (ByVal PrinterName As String, JOBINFO As tagJOBINFO, ByVal FirstNext As Integer) As Integer

## Call Syntax :

```
intResult% = cEnumPrinterJobs(PrinterName$, JOBINFO, FirstNext%)
```

## Where :

PrinterName\$ is the name of the printer for which the job is spooled;  
JOBINFO is the type'd tagJOBINFO;  
FirstNext% TRUE : begin the enumeration and return the first job;  
FALSE : continue the enumeration and return the next job;  
intResult% EPJ\_SUCCESS : all is ok  
EPJ\_PRINTER\_NAME\_EMPTY: PrinterName\$ is empty  
EPJ\_CANT\_OPEN\_PRINTER : can't open the specified PrinterName\$  
EPJ\_STRANGE\_ERROR : unknow error when accessing the enumeration  
EPJ\_CANT\_ENUMERATE\_MORE\_JOBS : no more jobs

## Comments :

The returned 'IStatus' can be one or more of the following constant value :

```
JOB_STATUS_PAUSED  
JOB_STATUS_ERROR  
JOB_STATUS_DELETING  
JOB_STATUS_SPOOLING  
JOB_STATUS_PRINTING  
JOB_STATUS_OFFLINE  
JOB_STATUS_PAPEROUT  
JOB_STATUS_PRINTED  
JOB_STATUS_DELETED  
JOB_STATUS_BLOCKED_DEVQ  
JOB_STATUS_USER_INTERVENTION
```

## Examples :

```
Dim intResult          As Integer  
Dim strDisplay         As String  
Dim JI                 As tagJOBINFO  
  
strDisplay = ""  
  
intResult = cEnumPrinterJobs("LPT1:", JI, True) 'first job  
  
Do While intResult = EPJ_SUCCESS  
  
    strDisplay = strDisplay + "sPrinterName : " & JI.sPrinterName & "" & vbCrLf  
    strDisplay = strDisplay + "sMachineName : " & JI.sMachineName & "" & vbCrLf  
    strDisplay = strDisplay + "sUserName : " & JI.sUserName & "" & vbCrLf  
    strDisplay = strDisplay + "sDocument : " & JI.sDocument & "" & vbCrLf
```

```
strDisplay = strDisplay + "JobId : " & JI.JobId & vbCrLf
strDisplay = strDisplay + "Status : " & JI.IStatus & vbCrLf
strDisplay = strDisplay + "Priority : " & JI.IPriority & vbCrLf
strDisplay = strDisplay + "Position : " & JI.IPosition & vbCrLf
strDisplay = strDisplay + "StartTime : " & JI.IStartTime & vbCrLf
strDisplay = strDisplay + "UntilTime : " & JI.IUntilTime & vbCrLf
strDisplay = strDisplay + "TotalPages : " & JI.ITotalPages & vbCrLf
strDisplay = strDisplay + "PagesPrinted : " & JI.IPagesPrinted & vbCrLf
strDisplay = strDisplay + "Size : " & JI.ISize & vbCrLf
strDisplay = strDisplay + "Time : " & JI.ITime & vbCrLf
strDisplay = strDisplay + "Submitted : " & JI.wMonth & "/" & JI.wDay & "/" & JI.wYear & " " & JI.wHour & ":" &
JI.wMinute & ":" & JI.wSecond & vbCrLf & vbCrLf
```

```
intResult = cEnumPrinterJobs("LPT1:", JI, False) 'next job
```

```
Loop
```

```
debug.print strDisplay
```

**See also :**

Select the following product :

TIME TO WIN for VB 3.0

TIME TO WIN for VB 4.0 (16-Bit)

TIME TO WIN for VB 4.0 (32-Bit)

TIME TO WIN for MSOffice 95

# TIME TO WIN for VB 3.0 : Registration

'TIME TO WIN' Library Registration Benefits :

- Create your application easier and faster
- Create a smaller application
- Accelerate the speed of your application
- Full support for one year

## [Registering the 'TIME TO WIN' Library \(DLL\) on CompuServe](#)

- 1) On CompuServe [GO SWREG](#)
- 2) Choose Register Shareware.
- 3) 'TIME TO WIN' SWREG ID is : #4045. (price is \$43.00)

## [Registering the 'TIME TO WIN' Library \(DLL\) on Internet](#)

- 1) Use the program REGISTER.EXE or download REGISTER.ZIP from my Internet Homepage.
- 2) Select the product TIME TO WIN (16-Bit).
- 3) Send by e-mail or fax or postal.

As soon as I receive notification of your registration (usually 1 - 3 days) I will send you out via e-Mail a license file for each single user license that you've asked, or site license or world license.

You also qualify to receive new versions of 'TIME TO WIN' during one year.

*This price is much a contribution to my works that a payment.*

*When you register 'TIME TO WIN', you help me to develop better products and others products.*

'TIME TO WIN' is written in C and has been compiled using Visual C++ 1.52c.  
The code has been optimized for 80386 use with the 'maximize speed' option.

'TIME TO WIN' can only be used with Visual Basic 3.0 under Windows 3.1x, Windows 95 and Windows NT.

# TIME TO WIN for VB 4.0 (16-Bit) : Registration

'TIME TO WIN (16-Bit)' Library Registration Benefits :

- Create your application easier and faster
- Create a smaller application
- Accelerate the speed of your application
- Full support for one year

## [Registering the 'TIME TO WIN \(16-Bit\)' Library \(DLL\) on CompuServe](#)

- 1) On CompuServe [GO SWREG](#)
- 2) Choose Register Shareware.
- 3) 'TIME TO WIN (16-Bit)' SWREG ID is : #4045. (price is \$43.00)

## [Registering the 'TIME TO WIN \(16-Bit\)' Library \(DLL\) on Internet](#)

- 1) Use the program REGISTER.EXE or download REGISTER.ZIP from my Internet Homepage.
- 2) Select the product TIME TO WIN (16-Bit).
- 3) Send by e-mail or fax or postal

As soon as I receive notification of your registration (usually 1 - 3 days) I will send you out via e-Mail a license file for each single user license that you've asked, or site license or world license.

You also qualify to receive new versions of 'TIME TO WIN' during one year.

*This price is much a contribution to my works that a payment.*

*When you register 'TIME TO WIN (16-Bit)', you help me to develop better products and others products.*

'TIME TO WIN (16-Bit)' is written in C and has been compiled using Visual C++ 1.52c.

The code has been optimized for 80386 use with the 'maximize speed' option.

'TIME TO WIN (16-Bit)' can only be used with Visual Basic 4.0 (16-Bit) under Windows 3.1x, Windows 95 and Windows NT.

# TIME TO WIN for VB 4.0 (32-Bit) : Registration

'TIME TO WIN (32-Bit)' Library Registration Benefits :

- Create your application easier and faster
- Create a smaller application
- Accelerate the speed of your application
- Full support for one year

## Registering the 'TIME TO WIN (32-Bit)' Library (DLL) on CompuServe

- 1) On CompuServe [GO SWREG](#)
- 2) Choose Register Shareware.
- 3) 'TIME TO WIN (32-Bit)' SWREG ID is : #7516. (price is \$59.95)

## Registering the 'TIME TO WIN (32-Bit)' Library (DLL) on Internet

- 1) Use the program REGISTER.EXE or download REGISTER.ZIP from my Internet Homepage.
- 2) Select the product TIME TO WIN (32-Bit).
- 3) Send by e-mail or fax or postal.

## Upgrading to 'TIME TO WIN (32-Bit)' Library from 'TIME TO WIN' or 'TIME TO WIN (16-Bit)' on CompuServe

- 1) On CompuServe [GO SWREG](#)
- 2) Choose Register Shareware.
- 3) 'UPDATE T2WIN -> T2WIN (32-Bit)' SWREG ID is : #7517. (price is \$34.95)

## Upgrading to 'TIME TO WIN (32-Bit)' Library from 'TIME TO WIN' or 'TIME TO WIN (16-Bit)' on Internet

- 1) Use the program REGISTER.EXE or download REGISTER.ZIP from my Internet Homepage.
- 2) Select the product update TIME TO WIN (16-Bit) to TIME TO WIN (32-Bit).
- 3) Send by e-mail or fax or postal.

As soon as I receive notification of your registration (usually 1 - 3 days) I will send you out via e-Mail a license file for each single user license that you've asked, or site license or world license.

You also qualify to receive new versions of 'TIME TO WIN (32-Bit)' during one year.

*This price is much a contribution to my works that a payment.*

*When you register 'TIME TO WIN (32-Bit)', you help me to develop better products and others products.*

'TIME TO WIN (32-Bit)' is written in C and has been compiled using Visual C++ 5.00.

The code for T2WIN-32.DLL has been optimized for 80486 use with the 'maximize speed' option.

The code for T2WIN32P.DLL has been optimized for Pentium Processor use with the 'maximize speed' option.

The code for T2W32PRO.DLL has been optimized for Pentium Pro Processor use with the 'maximize speed' option.

'TIME TO WIN (32-Bit)' can only be used with Visual Basic 4.0 (32-Bit Edition) or higher under Windows 95 and Windows NT.

# TIME TO WIN for MSOffice 95 : Registration

'TIME TO WIN for MSOffice 95' Library Registration Benefits :

- Create your application easier and faster
- Create a smaller application
- Accelerate the speed of your application
- Full support for one year

## [Registering the 'TIME TO WIN for MSOffice 95' Library \(DLL\) on CompuServe](#)

- 1) On CompuServe [GO SWREG](#)
- 2) Choose Register Shareware.
- 3) 'TIME TO WIN for MSOffice 95' SWREG ID is : #10355. (price is **\$25.00**)

## [Registering the 'TIME TO WIN for MSOffice 95' Library \(DLL\) on Internet](#)

- 1) Use the program REGISTER.EXE or download REGISTER.ZIP from my Internet Homepage.
- 2) Select the product TIME TO WIN for MSOffice 95.
- 3) Send by e-mail or fax or postal.

As soon as I receive notification of your registration (usually 1 - 3 days) I will send you out via e-Mail a license file for each single user license that you've asked, or site license or world license.

You also qualify to receive new versions of 'TIME TO WIN' during one year.

*This price is much a contribution to my works that a payment.*

*When you register 'TIME TO WIN for MSOffice 95', you help me to develop better products and others products.*

'TIME TO WIN for MSOffice 95' is written in C and has been compiled using Visual C++ 4.00.

The code has been optimized for 80486 use with the 'maximize speed' option.

'TIME TO WIN for MSOffice 95' can only be used with MSOffice 95.

# AddD, AddI, AddL, AddS

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[AddD](#) add a constant value to all of the elements of a [Double](#) array.

[AddI](#) add a constant value to all of the elements of an [Integer](#) array.

[AddL](#) add a constant value to all of the elements of a [Long](#) array.

[AddS](#) add a constant value to all of the elements of a [Single](#) array.

## Declare Syntax :

Declare Function cAddD Lib "[time2win.dll](#)" (array() As Double, ByVal nValue As Double) As Integer

Declare Function cAddI Lib "[time2win.dll](#)" (array() As Integer, ByVal nValue As Integer) As Integer

Declare Function cAddL Lib "[time2win.dll](#)" (array() As Long, ByVal nValue As Long) As Integer

Declare Function cAddS Lib "[time2win.dll](#)" (array() As Single, ByVal nValue As Single) As Integer

## Call Syntax :

status% = cAddD(array(), nValue)

status% = cAddI(array(), nValue)

status% = cAddL(array(), nValue)

status% = cAddS(array(), nValue)

## Where :

array() is the array (Double, Integer, Long, Single).

nValue is the value (Double, Integer, Long, Single) to add (if positive) or to subtract (if negative) to all of the elements of the array (Double, Integer, Long, Single).

status% always TRUE

## Comments :

**See Also :** [Array](#)



# Overview

'TIME TO WIN' is a DLL (Dynamic Link Library) for Visual Basic 3.0, Visual Basic 4.0 (16/32-Bit), Visual Basic 5.0, Visual Basic 5.0 for Application and MSOffice 95.

I'm an Engineer in Electricity and Electronic and I've written 'TIME TO WIN' to help any users of VB to find a solution at some missing functions in VB.

VB is a powerfull product but by some aspects it is limited.

I hope that 'TIME TO WIN' will be a great advantage for you and for your application.

'TIME TO WIN' contains more over **913** functions or subroutines (following product).

You can find functions or routines over the following sections :

[2-D Geometry](#)

[3-D Geometry](#)

[Array](#)

[Binary](#)

[Bitmap](#)

[Compression](#)

[Crc32](#)

[Date and time](#)

[Days and months in different language](#)

[Disk array](#)

[Encryption](#)

[File](#)

[File I/O from C](#)

[Hi-Crypt](#)

[Huge memory array](#)

[Huge string](#)

[IEEEEnum](#)

[Interest rate](#)

[Is](#)

[Language control](#)

[List box - combo box](#)

[Mail & News](#)

[Math](#)

[Matrix](#)

[Media ID - Volume](#)

[Miscellaneous](#)

[Multiple disk array](#)

[Multi language message box - input box](#)

[Network](#)

[Object](#)

[Printer](#)

[Process ID](#)

[Protection](#)

[Random](#)

[Registry key](#)

[Remote Access Service](#)

[Serialization](#)

[String](#)

[Swap](#)

[Task - File version](#)

[TIME2WIN](#)

[Timer](#)

[Type](#)

[UUCP](#)

[Windows](#)

[Windows 95/NT4.0x](#)

# ArrayOnDisk

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

ArrayOnDisk put/get full array on/from disk

## Declare Syntax :

Declare Function cArrayOnDisk Lib "time2win.dll" (ByVal File As String, Array() As Any, ByVal GetPut As Integer) As Long

## Call Syntax :

test& = cArrayOnDisk(File\$, Array(), GetPut%)

## Where :

File\$ is the file to use.  
Array() is the array with any dimension.  
GetPut% PUT\_ARRAY\_ON\_DISK to put the array on disk,  
GET\_ARRAY\_ON\_DISK to get the array from disk.  
test& >=0 is the returned length of the file,  
< 0 is an error occurs (error n° is the negative value of all DA\_x values, see [Constants and Types declaration](#)).

## Comments :

This function can handle any type'd variable (if strings are used, you must use only fixed string).

Don't forget that if you use the 'ReDim' statement at the procedure level without have declared the array als Global, you must initialize the array before using this function (see below). You must initialize the array with enough space to handle the size of the file This is due to a VB limitation.

This function can handle huge array (greater than 65535 bytes) (see the example below).

Beware, the ANY parameter in the defintion of this function doesn't support string array (why ? ask to VB creator). To handle string (only fixed string), create a type'd variable with only an item, see below :

```
Type tagStringType
    newString As String * 80
End Type
```

'This type replaces

```
Dim newString As String * 80
```

## Examples :

```
ReDim AD(-999 To 9000, 0 To 1) As Long 'size is ((1+(9000 - -999)) * (1+(1 - 0)) * 4) = 80.000 bytes
Dim i As Long
```

```
For i = -999 To 9000
    AD(i, 0) = 1
    AD(i, 1) = 2
Next i
```

```
Debug.Print cArrayOnDisk("c:\tmp\test.dat", AD(), PUT_ARRAY_ON_DISK) '-> 80.000
```

```
For i = -999 To 9000
```

```
        AD(i, 0) = 0
        AD(i, 1) = 0
Next i

Debug.Print cArrayOnDisk("c:\tmp\test.dat", AD(), GET_ARRAY_ON_DISK)    '-> 80.000

Debug.Print AD(-999, 0), AD(9000, 0)
Debug.Print AD(-999, 1), AD(9000, 1)
```

**See also :** [Array](#)

# Array : Overview

[ArrayLookUp](#)

array and returns value in the second dimension.

[ArrayOnDisk](#), [ArrayStringOnDisk](#)

[AddD](#), [AddI](#), [AddL](#), [AddS](#)

[ArrayPrm](#)

[CountD](#), [CountI](#), [CountL](#), [CountS](#)

[DeviationD](#), [DeviationI](#), [DeviationL](#), [DeviationS](#)

in an array

[FillD](#), [FillI](#), [FillL](#), [FillS](#)

any element.

[FillIncrD](#), [FillIncrI](#), [FillIncrL](#), [FillIncrS](#)

increment for any element.

[MaxD](#), [MaxI](#), [MaxL](#), [MaxS](#)

[MaxNotXD](#), [MaxNotXI](#), [MaxNotXL](#), [MaxNotXS](#)

a specified value.

[MeanD](#), [MeanI](#), [MeanL](#), [MeanS](#)

[MinD](#), [MinI](#), [MinL](#), [MinS](#)

[MinNotXD](#), [MinNotXI](#), [MinNotXL](#), [MinNotXS](#)

specified value.

[ReverseSortD](#), [ReverseSortI](#), [ReverseSortL](#), [ReverseSortS](#)

[ReverseSortStr](#)

[RmvDupD](#), [RmvDupI](#), [RmvDupL](#), [RmvDupS](#)

dimensions).

[SearchD](#), [SearchI](#), [SearchL](#), [SearchS](#)

[SetD](#), [SetI](#), [SetL](#), [SetS](#)

value.

[ShiftLeftB](#), [ShiftLeftD](#), [ShiftLeftI](#), [ShiftLeftL](#), [ShiftLeftS](#)

value.

[ShiftRightB](#), [ShiftRightD](#), [ShiftRightI](#), [ShiftRightL](#), [ShiftRightS](#)

value.

[SortD](#), [SortI](#), [SortL](#), [SortS](#)

[SortStr](#)

[SortTypedArray](#)

elements of any data type, array, or a previously defined user-defined type.

[SumD](#), [SumI](#), [SumL](#), [SumS](#)

[SearchStr](#)

Searchs an element in a two-dimensionnal string

Put/Get full array on/from disk.

Adding a value to all elements in an array

Read the configuration of an array.

Count a specific value in an array.

Calculating the standard deviation from all elements

Filling an array with a value incremented by one for

Filling an array with a value incremented by an

Finding the maximum value in an array.

Finding the maximum value in an array, not equal to

Calculating the mean from all elements in an array.

Finding the minimum value in an array.

Finding the minimum value in an array, not equal to a

Sort an array in descending order.

Sort, in descending order, a string.

Remove duplicate values in an array (one and two

Search a specific value in an array.

Setting all elements in an array with the same

Shift an array to Left and set the last element to a

Shift an array to Right and set the first element to a

Sort an array in ascending order.

Sort, in ascending order, a string.

Sort a Type'd array which can contain one or more

Sum all elements from an array.

searchs a string array for a given string

# DeviationD, DeviationI, DeviationL, DeviationS

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[DeviationD](#) will calculate the standard deviation from all elements in a [Double](#) array.

[DeviationI](#) will calculate the standard deviation from all elements in an [Integer](#) array.

[DeviationL](#) will calculate the standard deviation from all elements in a [Long](#) array.

[DeviationS](#) will calculate the standard deviation from all elements in a [Single](#) array.

## Declare Syntax :

Declare Function cDeviationD Lib "[time2win.dll](#)" (array() As Double) As Double

Declare Function cDeviationI Lib "[time2win.dll](#)" (array() As Integer) As Double

Declare Function cDeviationL Lib "[time2win.dll](#)" (array() As Long) As Double

Declare Function cDeviationS Lib "[time2win.dll](#)" (array() As Single) As Double

## Call Syntax :

deviation# = cDeviationD(array())

deviation# = cDeviationI(array())

deviation# = cDeviationL(array())

deviation# = cDeviationS(array())

## Where :

array() is the array (Double, Integer, Long, Single).

deviation# is the standard deviation calculated. This value is always a Double value.

## Comments :

**See Also :** [Array](#)

# ArrayPrm

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ArrayPrm](#) retrieve the definition of a given array (only one dimension and for numeric array)

## Declare Syntax :

Declare Function cArrayPrm Lib "[time2win.dll](#)" (array() As Any, nArray As Any) As Integer

## Call Syntax :

status% = cArrayPrm(array(), nArray)

## Where :

array()	the array to proceed
nArray	a type variable 'ArrayType' for receiving the definition
status%	always TRUE

## Comments :

The definition of an array is given by the following parameters :

Bounds	is the far address of the array in memory.
LBound	is the smallest available subscript for the first dimension of the array.
UBound	is the highest available subscript for the first dimension of the array.
ElemSize	is the size of the element of the array
IndexCount	is the number of dimension of the array.
TotalElem	is the number of element in the array (UBound - LBound + 1) in the first dimension.

## Examples :

Dim array(1 To 16)	As Integer
Dim arrayDef	As ArrayType
Dim status	As Integer

status = cArrayPrm(array(), arrayDef)

arrayDef.Bounds	is 1048577
arrayDef.LBound	is 1
arrayDef.UBound	is 16
arrayDef.ElemSize	is 2 (INTEGER)
arrayDef.IndexCount	is 1
arrayDef.TotalElem	is 16

Dim array(-7 To 25)	As Double
Dim arrayDef	As ArrayType
Dim status	As Integer

status = cArrayPrm(array(), arrayDef)

arrayDef.Bounds	is 1703929
arrayDef.LBound	is -7
arrayDef.UBound	is 25
arrayDef.ElemSize	is 8 (DOUBLE)
arrayDef.IndexCount	is 1
arrayDef.TotalElem	is 33

Dim array(-10 To 10, 1 TO 7)	As Long
Dim arrayDef	As ArrayType



# FillD, FillI, FillL, FillS

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

**FillD** fill, with an automatic incremented value, all of the elements of a [Double](#) array.

**FillI** fill, with an automatic incremented value, all of the elements of an [Integer](#) array.

**FillL** fill, with an automatic incremented value, all of the elements of a [Long](#) array.

**FillS** fill, with an automatic incremented value, all of the elements of a [Single](#) array.

## Declare Syntax :

Declare Function cFillD Lib "[time2win.dll](#)" (array() As Double, ByVal nValue As Double) As Integer

Declare Function cFillI Lib "[time2win.dll](#)" (array() As Integer, ByVal nValue As Integer) As Integer

Declare Function cFillL Lib "[time2win.dll](#)" (array() As Long, ByVal nValue As Long) As Integer

Declare Function cFillS Lib "[time2win.dll](#)" (array() As Single, ByVal nValue As Single) As Integer

## Call Syntax :

status% = cFillD(array(), nValue#)

status% = cFillI(array(), nValue%)

status% = cFillL(array(), nValue&)

status% = cFillS(array(), nValue!)

## Where :

array() is the Double array.

nValue is the Double value automatically incremented by one.

status is always TRUE.

## Comments :

**See Also :** [Array](#)



# FillIncrD, FillIncrI, FillIncrL, FillIncrS

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FillIncrD](#) fill, with an automatic incremented value, all of the elements of a [Double](#) array.

[FillIncrI](#) fill, with an automatic incremented value, all of the elements of an [Integer](#) array.

[FillIncrL](#) fill, with an automatic incremented value, all of the elements of a [Long](#) array.

[FillIncrS](#) fill, with an automatic incremented value, all of the elements of a [Single](#) array.

## Declare Syntax :

```
Declare Function cFillIncrD Lib "time2win.dll" (Array() As Double, ByVal nValue As Double, ByVal Increment As Double) As Integer
```

```
Declare Function cFillIncrI Lib "time2win.dll" (Array() As Integer, ByVal nValue As Integer, ByVal Increment As Integer) As Integer
```

```
Declare Function cFillIncrL Lib "time2win.dll" (Array() As Long, ByVal nValue As Long, ByVal Increment As Long) As Integer
```

```
Declare Function cFillIncrS Lib "time2win.dll" (Array() As Single, ByVal nValue As Single, ByVal Increment As Single) As Integer
```

## Call Syntax :

```
status% = cFillIncrD(array(), nValue#, Increment#)
```

```
status% = cFillIncrI(array(), nValue%, Increment%)
```

```
status% = cFillIncrL(array(), nValue&, Increment&)
```

```
status% = cFillIncrS(array(), nValue!, Increment!)
```

## Where :

array()	is the array (Double, Integer, Long, Single).
nValue	is the starting value (Double, Integer, Long, Single).
Increment	is the increment (Double, Integer, Long, Single).
status	is always TRUE.

## Comments :

**See Also** : [Array](#)

# MaxD, MaxI, MaxL, MaxS

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[MaxD](#) will return the largest value in a [Double](#) array.

[MaxI](#) will return the largest value in an [Integer](#) array.

[MaxL](#) will return the largest value in a [Long](#) array.

[MaxS](#) will return the largest value in a [Single](#) array.

## Declare Syntax :

```
Declare Function cMaxD Lib "time2win.dll" (array() As Double) As Double
```

```
Declare Function cMaxI Lib "time2win.dll" (array() As Integer) As Integer
```

```
Declare Function cMaxL Lib "time2win.dll" (array() As Long) As Long
```

```
Declare Function cMaxS Lib "time2win.dll" (array() As Single) As Single
```

## Call Syntax :

```
largest# = cMaxD(array())
```

```
largest% = cMaxI(array())
```

```
largest& = cMaxL(array())
```

```
largest! = cMaxS(array())
```

## Where :

array() is the array (Double, Integer, Long, Single).

largest is the largest value (Double, Integer, Long, Single) from all of the elements of the array (Double, Integer, Long, Single).

## Comments :

**See Also :** [Array](#)

# MeanD, MeanI, MeanL, MeanS

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**MeanD** will calculate the mean from all elements in a **Double** array.

**MeanI** will calculate the mean from all elements in an **Integer** array.

**MeanL** will calculate the mean from all elements in a **Long** array.

**MeanS** will calculate the mean from all elements in a **Single** array.

## Declare Syntax :

```
Declare Function cMeanD Lib "time2win.dll" (array() As Double) As Double
```

```
Declare Function cMeanI Lib "time2win.dll" (array() As Integer) As Double
```

```
Declare Function cMeanL Lib "time2win.dll" (array() As Long) As Double
```

```
Declare Function cMeanS Lib "time2win.dll" (array() As Single) As Double
```

## Call Syntax :

```
mean# = cMeanD(array())
```

```
mean% = cMeanI(array())
```

```
mean& = cMeanL(array())
```

```
mean! = cMeanS(array())
```

## Where :

array() is the array (Double, Integer, Long, Single).

mean is the mean calculated. This value is always a Double value.

## Comments :

**See Also :** [Array](#)

# MinD, MinI, MinL, MinS

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[MinD](#) will return the smallest value in a [Double](#) array.

[MinI](#) will return the smallest value in an [Integer](#) array.

[MinL](#) will return the smallest value in a [Long](#) array.

[MinS](#) will return the smallest value in a [Single](#) array.

## Declare Syntax :

Declare Function cMinD Lib "[time2win.dll](#)" (array() As Double) As Double

Declare Function cMinI Lib "[time2win.dll](#)" (array() As Integer) As Integer

Declare Function cMinL Lib "[time2win.dll](#)" (array() As Long) As Long

Declare Function cMinS Lib "[time2win.dll](#)" (array() As Single) As Single

## Call Syntax :

smallest# = cMinD(array())

smallest% = cMinI(array())

smallest& = cMinL(array())

smallest! = cMinS(array())

## Where :

array() is the array (Double, Integer, Long, Single).

smallest is the smallest value (Double, Integer, Long, Single) from all of the elements of the array (Double, Integer, Long, Single).

## Comments :

**See Also** : [Array](#)

# ReverseSortD, ReverseSortI, ReverseSortL, ReverseSortS, ReverseSortStr

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 \(Win95/WinNT\)](#), [MSOffice 95](#)

## Purpose :

[ReverseSortD](#) will sort, in descending order, all elements in a [Double](#) array.  
[ReverseSortI](#) will sort, in descending order, all elements in an [Integer](#) array.  
[ReverseSortL](#) will sort, in descending order, all elements in a [Long](#) array.  
[ReverseSortS](#) will sort, in descending order, all elements in a [Single](#) array.  
[ReverseSortStr](#) will sort, in descending order, a string divided in basis elements of a fixed length.

## Declare Syntax :

```
Declare Function cReverseSortD Lib "time2win.dll" (array() As Double) As Integer
Declare Function cReverseSortI Lib "time2win.dll" (array() As Integer) As Integer
Declare Function cReverseSortL Lib "time2win.dll" (array() As Long) As Integer
Declare Function cReverseSortS Lib "time2win.dll" (array() As Single) As Integer
Declare Function cReverseSortStr Lib "time2win.dll" (Txt As String, ByVal nItem As Integer, ByVal ItemLength As Integer) As Integer
```

## Call Syntax :

```
status% = cReverseSortD(array())
status% = cReverseSortI(array())
status% = cReverseSortL(array())
status% = cReverseSortS(array())
status% = cReverseSortStr(txt$, nItem%, ItemLength%)
```

## Where :

For [ReverseSortD](#), [ReverseSortI](#), [ReverseSortL](#), [ReverseSortS](#) :

array()	is the array (Double, Integer, Long, Single).
status%	is always TRUE.

For [ReverseSortStr](#) :

txt	is the string to sort.
nItem	is the total element is the string.
ItemLength	is the length for one element.
status	is FALSE if the length of the string is not the 'nItem * ItemLength', or if length of the string is 0. is TRUE if all is OK.

## Comments :

With TIME TO WIN 32-Bit, you can sort numerical array with one and two dimensions.

**See Also** : [Array](#)

# SetD, SetI, SetL, SetS

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SetD](#) fill, with the same value, all of the elements of a [Double](#) array.

[SetI](#) fill, with the same value, all of the elements of an [Integer](#) array.

[SetL](#) fill, with the same value, all of the elements of a [Long](#) array.

[SetS](#) fill, with the same value, all of the elements of a [Single](#) array.

## Declare Syntax :

Declare Function cSetD Lib "[time2win.dll](#)" (array() As Double, ByVal nValue As Double) As Integer

Declare Function cSetI Lib "[time2win.dll](#)" (array() As Integer, ByVal nValue As Integer) As Integer

Declare Function cSetL Lib "[time2win.dll](#)" (array() As Long, ByVal nValue As Long) As Integer

Declare Function cSetS Lib "[time2win.dll](#)" (array() As Single, ByVal nValue As Single) As Integer

## Call Syntax :

status = cSetD(array(), nValue)

status = cSetI(array(), nValue)

status = cSetL(array(), nValue)

status = cSetS(array(), nValue)

## Where :

array() is the array (Double, Integer, Long, Single).

nValue is the value (Double, Integer, Long, Single) to initialize the array.

status is always TRUE.

## Comments :

**See Also :** [Array](#)

# SortD, SortI, SortL, SortS, SortStr

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**SortD** will sort, in ascending order, all elements in a **Double** array.

**SortI** will sort, in ascending order, all elements in an **Integer** array.

**SortL** will sort, in ascending order, all elements in a **Long** array.

**SortS** will sort, in ascending order, all elements in a **Single** array.

**SortStr** will sort, in ascending order, a string divided in basis elements of a fixed length.

## Declare Syntax :

```
Declare Function cSortD Lib "time2win.dll" (array() As Double) As Integer
```

```
Declare Function cSortI Lib "time2win.dll" (array() As Integer) As Integer
```

```
Declare Function cSortL Lib "time2win.dll" (array() As Long) As Integer
```

```
Declare Function cSortS Lib "time2win.dll" (array() As Single) As Integer
```

```
Declare Function cSortStr Lib "time2win.dll" (Txt As String, ByVal nItem As Integer, ByVal ItemLength As Integer) As Integer
```

## Call Syntax :

```
status% = cSortD(array())
```

```
status% = cSortI(array())
```

```
status% = cSortL(array())
```

```
status% = cSortS(array())
```

```
status% = cSortStr(txt$, nItem%, ItemLength%)
```

## Where :

For SortD, SortI, SortL, SortS :

array()	is the array (Double, Integer, Long, Single).
status%	is always TRUE.

For SortStr :

txt	is the string to sort.
nItem	is the total element is the string.
ItemLength	is the length for one element.
status	is FALSE if the length of the string is not the 'nItem * ItemLength', or if length of the string is 0. is TRUE if all is OK.

## Comments :

With TIME TO WIN 32-Bit, you can sort numerical array with one and two dimensions.

**See Also** : [Array](#)

# SumD, SumI, SumL, SumS

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**SumD** will calculate the sum from all elements in a **Double** array.

**SumI** will calculate the sum from all elements in an **Integer** array.

**SumL** will calculate the sum from all elements in a **Long** array.

**SumS** will calculate the sum from all elements in a **Single** array.

## Declare Syntax :

Declare Function cSumD Lib "[time2win.dll](#)" (array() As Double) As Double

Declare Function cSumI Lib "[time2win.dll](#)" (array() As Integer) As Double

Declare Function cSumL Lib "[time2win.dll](#)" (array() As Long) As Double

Declare Function cSumS Lib "[time2win.dll](#)" (array() As Single) As Double

## Call Syntax :

sum# = cSumD(array())

sum% = cSumI(array())

sum& = cSumL(array())

sum! = cSumS(array())

## Where :

array() is the array (Double, Integer, Long, Single).

sum is the sum calculated. This value is always a Double value.

## Comments :

**See Also :** [Array](#)



# CountD, CountI, CountL, CountS

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CountD](#) counts a specific value in a [Double](#) array.

[CountI](#) counts a specific value in an [Integer](#) array.

[CountL](#) counts a specific value in a [Long](#) array.

[CountS](#) counts a specific value in a [Single](#) array.

## Declare Syntax :

Declare Function cCountD Lib "[time2win.dll](#)" (array() As Double, ByVal Value As Double) As Long

Declare Function cCountI Lib "[time2win.dll](#)" (array() As Integer, ByVal Value As Integer) As Long

Declare Function cCountL Lib "[time2win.dll](#)" (array() As Long, ByVal Value As Long) As Long

Declare Function cCountS Lib "[time2win.dll](#)" (array() As Single, ByVal Value As Single) As Long

## Call Syntax :

cnt& = cCountD(array(), Value!)

cnt& = cCountI(array(), Value%)

cnt& = cCountL(array(), Value&)

cnt& = cCountS(array(), Value#)

## Where :

array() is the array (Double, Integer, Long, Single).

Value? is the value (Double, Integer, Long, Single) to count.

cnt& is the returned counted value.

## Comments :

**See Also :** [Array](#)

# SearchD, SearchI, SearchL, SearchS

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SearchD](#) search a specific value in a [Double](#) array.

[SearchI](#) search a specific value in an [Integer](#) array.

[SearchL](#) search a specific value in a [Long](#) array.

[SearchS](#) search a specific value in a [Single](#) array.

## Declare Syntax :

Declare Function cSearchD Lib "[time2win.dll](#)" (array() As Double, ByVal Value As Double) As Long

Declare Function cSearchI Lib "[time2win.dll](#)" (array() As Integer, ByVal Value As Integer) As Long

Declare Function cSearchL Lib "[time2win.dll](#)" (array() As Long, ByVal Value As Long) As Long

Declare Function cSearchS Lib "[time2win.dll](#)" (array() As Single, ByVal Value As Single) As Long

## Call Syntax :

cnt& = cSearchD(array(), Value#)

cnt& = cSearchI(array(), Value%)

cnt& = cSearchL(array(), Value&)

cnt& = cSearchS(array(), Value!)

## Where :

array() is the array (Double, Integer, Long, Single).

Value? is the value to search (Double, Integer, Long, Single).

cnt& > 0 : the position of the searched value;

= -1 : the searched value is not found.

## Comments :

**See Also** : [Array](#)

# Disk array : Overview

The functions/subs used in the Disk Array routines handle big sized arrays on disk.

Each array must give/have a file to handle the information.

The concept of big sized arrays on disk is to use the mass storage (hard disk) in place of memory. This concept minimize the use of the memory for big array but decrease the speed to accessing data.

A fixed string array of 500 rows by 500 cols, 2 Sheets and a string size of 50 take 25.000.000 bytes. I think that this is better to place this array on the disk.

The following functions/subs are used to handle big sized arrays on disk :

<u>DAClear</u>	clear a big sized array.
<u>DAClearCol</u>	clear a single col on on a sheet in a big sized array.
<u>DAClearRow</u>	clear a single row on a sheet in a big sized array.
<u>DAClearSheet</u>	clear a single sheet in a big sized array.
<u>DAClose</u>	close a big sized array and keep it or close a big sized array and destroy it.
<u>DACreate</u>	create a new big sized array on disk or use an existing big sized array on disk.
<u>DAGet</u>	read an element from a big sized array on disk.
<u>DAGGetType</u>	read a type'd variable from a big sized array on disk.
<u>DAPut</u>	save an element to a big sized array on disk.
<u>DAPutType</u>	save a type'd variable to a big sized array on disk.
<u>DArGet</u>	read an element from a big sized array on disk with only one sheet and one row.
<u>DArGetType</u>	read a type'd variable from a big sized array on disk with only one sheet and one row.
<u>DArPut</u>	save an element to a big sized array on disk with only one sheet and one row.
<u>DArPutType</u>	save a type'd variable to a big sized array on disk with only one sheet
<u>DAsClearCol</u>	clear a single col on on a sheet in a big sized array with only one sheet.
<u>DAsClearRow</u>	clear a single row on a sheet in a big sized array with only one sheet.
<u>DAsGet</u>	read an element from a big sized array on disk with only one sheet.
<u>DAsGetType</u>	read a type'd variable from a big sized array on disk with only one sheet.
<u>DAsPut</u>	save an element to a big sized array on disk with only one sheet.
<u>DAsPutType</u>	save a type'd variable to a big sized array on disk with only one sheet.and one row.

To minimize the use of too many functions for the different variable type in VB, DAGet and DAPut uses variant value (integer, long, single, double, currency, string). This can be slow down (a little bit) the speed for accessing the data.

To handle type'd variable, you must use DAGGetType, DAPutType.

When you create a new array on disk, a header (128 chars for VB 3.0 and VB 4.0 (16-Bit), 200 chars for VB 4.0 (32-Bit)) is writed to begin of the associated file. This header is readed when you re-use an existing array to verify that this is a good big sized disk array.

Actually, the maximum number of chars for a string element or for a type'd variable is [4096](#).

# DAClear, DAClearSheet, DAClearCol, DAsClearCol, DAClearRow, DAsClearRow

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[DAClear](#) clear a big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).

[DAClearSheet](#) clear a single Sheet in a big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).

[DAClearCol](#) clear a single Col on one Sheet or on all Sheets in a big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).

[DAsClearCol](#) have the same functionality but with a big sized array with only one sheet.

[DAClearRow](#) clear a single Row on one Sheet or on all Sheets in a big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).

[DAsClearRow](#) have the same functionality but with a big sized array with only one sheet.

## **Declare Syntax :**

```
Declare Function cDAClear Lib "time2win.dll" (DISKARRAY As tagDISKARRAY) As Integer
```

```
Declare Function cDAClearSheet Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Sheet As Long) As Integer
```

```
Declare Function cDAClearCol Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, ByVal Sheet As Long) As Integer
```

```
Declare Function cDAsClearCol Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long) As Integer
```

```
Declare Function cDAClearRow Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Sheet As Long) As Integer
```

```
Declare Function cDAsClearRow Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long) As Integer
```

## **Call Syntax :**

```
ErrCode% = cDAClear(DISKARRAY)
```

```
ErrCode% = cDAClearSheet(DISKARRAY, Sheet&)
```

```
ErrCode% = cDAClearCol(DISKARRAY, Col&, Sheet&)
```

```
ErrCode% = cDAsClearCol(DISKARRAY, Col&)
```

```
ErrCode% = cDAClearRow(DISKARRAY, Row&, Sheet&)
```

```
ErrCode% = cDAsClearRow(DISKARRAY, Row&)
```

## **Where :**

DISKARRAY is a type'd variable (tagDISKARRAY).

Col& is the desired Col.

Row& is the desired Row.

Sheet& is the desired Sheet.

ErrCode% is the returned error code.

## **Comments :**

This function must be used only after you've created a big sized array on disk OR after the using of an existing big sized array on disk.

If you've created a big sized array on disk, the array is already cleared.

For DAClearSheet :

If the big sized array on disk have a single Sheet, this routine have the same effect that cDAClear.

If the Sheet is -1 then all Sheets are used. This parameter have the same functionality that cDAClear

If the Sheet is below 1 and different of -1, the Sheet 1 is used.

If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

For DAClearCol, DAsClearCol :

If the Col is below 1, the Col 1 is used.

If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.

If the Sheet is -1 then all Sheets are used.

If the Sheet is below 1 and different of -1, the Sheet 1 is used.

If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

For DAClearRow, DAsClearRow :

If the Row is below 1, the Row 1 is used.

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.

If the Sheet is -1 then all Sheets are used.

If the Sheet is below 1 and different of -1, the Sheet 1 is used.

If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

### Examples :

Dim ErrCode

As Integer

Dim DA

As tagDISKARRAY

DA.nFilename = "c:\t2w\_tmp\dastring.tmp"

'name of the file to use

DA.nType = 50

'positive value for a string

DA.nIsTyped = False

'init the array with spaces

DA.nRows = 500

'500 rows

DA.nCols = 500

'500 cols

DA.nSheets = 2

'2 sheets

ErrCode = cDACreate(DA, True)

'create a new big sized array on disk

Call cDAPut(DA, 1, 1, 1, "D:1, ABCDEFGHIJ")

'save the string in Row 1, Col 1, Sheet 1

Call cDAPut(DA, 1, DA.nCols, 1, "D:1, abcdefghij")

'save the string in Row 1, Col 500, Sheet 1

Call cDAPut(DA, DA.nRows, 1, 1, "D:1, OPQRSTUVWXYZ")

'save the string in Row 500, Col 1, Sheet 1

Call cDAPut(DA, DA.nRows, DA.nCols, 1, "D:1, oprqstuvwxyz")

'save the string in Row 500, Col 500, Sheet 1

'..... some codes

ErrCode = cDAClear(DA)

disk

'clear all elements in the big sized array on disk

ErrCode = cDAClearSheet(DA, 2)

disk

'clear the Sheet 2 in the big sized array on disk

ErrCode = cDAClearCol(DA, DA.nCols, 2)

array on disk

'clear the last Col in Sheet 2 in the big sized array on disk

ErrCode = cDAsClearCol(DA, DA.nCols)

array on disk

'clear the last Col in Sheet 1 in the big sized array on disk

ErrCode = cDAClearRow(DA, DA.nRows, 2)

array on disk

'clear the last Row in Sheet 2 in the big sized array on disk

ErrCode = cDAsClearRow(DA, DA.nRows)

array on disk

'clear the last Row in Sheet 1 in the big sized array on disk

**See also :** [Disk Array](#)

# DAGet, DAsGet, DArGet

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DAGet](#) read an element from a big sized array on disk.

[DArGet](#) have the same fonctionnality but with a big sized array with only one sheet and only one row.

[DAsGet](#) have the same fonctionnality but with a big sized array with only one sheet.

## Declare Syntax :

Declare Function cDAGet Lib "[time2win.dll](#)" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long) As Variant

Declare Function cDArGet Lib "[time2win.dll](#)" (DISKARRAY As tagDISKARRAY, ByVal Col As Long) As Variant

Declare Function cDAsGet Lib "[time2win.dll](#)" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long) As Variant

## Call Syntax :

Var = cDAGet(DISKARRAY, Row&, Col&, Sheet&)

## Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
Var	is the readed variant value depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.

If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.

If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

## Examples :

see [DACreate](#)

**See also :** [Disk Array](#)

# DAGetType, DAsGetType, DArGetType

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**DAGetType** read a type'd variable from a big sized array on disk.

**DArGetType** have the same fonctionnality but with a big sized array with only one sheet and only one row.

**DAsGetType** have the same fonctionnality but with a big sized array with only one sheet.

## Declare Syntax :

Declare Sub cDAGetType Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)

Declare Sub cDArGetType Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, nType As Any)

Declare Sub cDAsGetType Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)

## Call Syntax :

Call cDAGetType(DISKARRAY, Row&, Col&, Sheet&, nType)

Call cDArGetType(DISKARRAY, Col&, nType)

Call cDAsGetType(DISKARRAY, Row&, Col&, nType)

## Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
nType	is the readed type'd variable depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.

If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.

If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

## Examples :

Dim ErrCode	As Integer
Dim DA	As tagDISKARRAY
Dim TE(1 To 4)	As tagTASKENTRY

DA.nFilename = "c:\t2w_tmp\datatype.tmp"	'name of the file to use
DA.nType = Len(TE(1))	'positive value for a type'd variable
DA.nIsTyped = True	'init the array with chr\$(0) because type'd variable
DA.nRows = 500	'500 rows
DA.nCols = 500	'500 cols
DA.nSheets = 2	'2 sheets

ErrCode = cDAGetType(DA, False) 'use a created big sized array on disk

Call cDAGetType(DA, 1, 1, 1, TE(1)) 'read the type'd variable in Row 1, Col 1, Sheet 1

Call cDAGetType(DA, 1, DA.nCols, 1, TE(2)) 'read the type'd variable in Row 1, Col 500,

Sheet 1

Call cDAGetType(DA, DA.nRows, 1, 1, TE(3))

Sheet 1

Call cDAGetType(DA, DA.nRows, DA.nCols, 1, TE(4))

'read the type'd variable in Row 500, Col 1,

'read the type'd variable in Row 500, Col 500, Sheet 1

**See also :** [Disk Array](#)



# DAPut, DAsPut, DArPut

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DAPut](#) save an element to a big sized array on disk.

[DArPut](#) have the same functionality but with a big sized array with only one sheet and only one row.

[DAsPut](#) have the same functionality but with a big sized array with only one sheet.

## Declare Syntax :

Declare Sub cDAPut Lib "[time2win.dll](#)" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, Var As Variant)

Declare Sub cDArPut Lib "[time2win.dll](#)" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, Var As Variant)

Declare Sub cDAsPut Lib "[time2win.dll](#)" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, Var As Variant)

## Call Syntax :

Call cDAPut(DISKARRAY, Row&, Col&, Sheet&, Var)

Call cDArPut(DISKARRAY, Col&, Var)

Call cDAsPut(DISKARRAY, Row&, Col&, Var)

## Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
Var	is the variant value to save depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.

If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.

If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

## Examples :

see [DACreate](#)

**See also :** [Disk Array](#)

# DAPutType, DAsPutType, DArPutType

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**DAPutType** save a type'd variable from a big sized array on disk.

**DArPutType** have the same fonctionnality but with a big sized array with only one sheet and only one row.

**DAsPutType** have the same fonctionnality but with a big sized array with only one sheet.

## Declare Syntax :

Declare Sub cDAPutType Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)

Declare Sub cDArPutType Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, nType As Any)

Declare Sub cDAsPutType Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)

## Call Syntax :

Call cDAPutType(DISKARRAY, Row&, Col&, Sheet&, nType)

Call cDArPutType(DISKARRAY, Col&, nType)

Call cDAsPutType(DISKARRAY, Row&, Col&, nType)

## Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
nType	is the type'd variable to save depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.

If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.

If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

## Examples :

```
Dim ErrCode           As Integer
Dim DA                As tagDISKARRAY
Dim TE                As tagTASKENTRY
```

DA.nFilename = "c:\t2w_tmp\datatype.tmp"	'name of the file to use
DA.nType = Len(TE)	'positive value for a type'd variable
DA.nIsTyped = True	'init the array with chr\$(0) because type'd variable
DA.nRows = 500	'500 rows
DA.nCols = 500	'500 cols
DA.nSheets = 2	'2 sheets
ErrCode = cDACreate(DA, True)	'create a new big sized array on disk
ErrCode = cTasks(TE, True)	
Call cDAPutType(DA, 1, 1, 1, TE)	'save the type'd variable in Row 1, Col 1, Sheet 1
Sheet 1	

ErrCode = cTasks(TE, False)  
Call cDAPutType(DA, 1, DA.nCols, 1, TE)  
Sheet 1

'save the type'd variable in Row 1, Col 500,

ErrCode = cTasks(TE, False)  
Call cDAPutType(DA, DA.nRows, 1, 1, TE)  
Sheet 1

'save the type'd variable in Row 500, Col 1,

ErrCode = cTasks(TE, False)  
Call cDAPutType(DA, DA.nRows, DA.nCols, 1, TE)  
500, Sheet 1

'save the type'd variable in Row 500, Col

**See also :** [Disk Array](#)

# DACreate

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

DACreate create a new big sized array on disk or use an existing big sized array on disk.

## Declare Syntax :

Declare Function cDACreate Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal CreateOrUse As Integer) As Integer

## Call Syntax :

ErrCode% = cDACreate(DA, CreateOrUse%)

## Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
CreateOrUse%	TRUE : if you want to create a new big sized array on disk, FALSE : if you want to re-use an existing big sized array on disk.
ErrCode%	is the returned <u>error code</u> .

## Comments :

In theory :

The maxixum number of Rows is 2147483647  
The maxixum number of Cols is 2147483647  
The maxixum number of Sheets is 2147483647

You are only limited by the size of the disk on which the big sized array are defined.

The length of the filename can be 64 (VB 3.0, VB 4.0 (16-Bit)) or 128 (VB 4.0 (32-Bit)) chars maximum.

If you create a new big sized array on disk and if the file is already exists, the file is deleted before used.  
If you re-use an existing big sized array on disk, some checkings are made to verify the validity of the big sized array on disk.

Bigger are nRows, nCols or nSheets, bigger is the time to initialize.

When you create a new big sized array on disk, the only parameters that you must initialize are :

DA.nFilename = "c:\t2w_tmp\dastring.tmp"	'name of the file (you must have enough space on the drive).
DA.nType = 50	'the type of the variable to use, see <u>Constants and Types declaration</u> . (DA_x)
DA.nIsTyped = False	'Must be True for a type'd variable.
DA.nRows = 500	'the number of rows to use.
DA.nCols = 500	'the number of cols to use.
DA.nSheets = 2	'the number of sheets to use.

**YOU CAN'T CHANGE THESE PARAMETERS AFTER THE CREATION OF THE BIG SIZED ARRAY.  
YOU CAN'T CHANGE THE OTHER VALUES IN THE TYPE'D VARIABLE.**

When you create a new array, all elements are initialized with chr\$(0) except for string array which are initialized with chr\$(32) (spaces).

However, string array and type'd array use the same positive value to define in .nType, but the type'd array must be initialized with chr\$(0) not with chr\$(32) therefore for a type'd you must specify .nIsTyped on True to initialize it with chr\$(0).

If you use big size array of type'd variable, the type'd variable can be only a mix of fixed variable (variable string length can't be used).

### Examples :

Dim ErrCode               As Integer  
Dim DA                    As tagDISKARRAY  
Dim Var(1 To 8)           As Variant

DA.nFilename = "c:\t2w\_tmp\dastring.tmp"  
DA.nType = 50  
DA.nIsTyped = False  
DA.nRows = 500  
DA.nCols = 500  
DA.nSheets = 2

'name of the file to use  
'positive value for a string  
'init the array with spaces  
'500 rows  
'500 cols  
'2 sheets

ErrCode = cDACreate(DA, True)

'create a new big sized array on disk

Call cDAPut(DA, 1, 1, 1, "D:1, ABCDEFGHIJ")  
Call cDAPut(DA, 1, DA.nCols, 1, "D:1, abcdefghij")  
Call cDAPut(DA, DA.nRows, 1, 1, "D:1, OPQRSTUVWXYZ")  
Call cDAPut(DA, DA.nRows, DA.nCols, 1, "D:1, oprqstuvwxyz")  
500, Sheet 1

'save the string in Row 1, Col 1, Sheet 1  
'save the string in Row 1, Col 500, Sheet 1  
'save the string in Row 500, Col 1, Sheet 1  
'save the string in Row 500, Col 500, Sheet 1

Call cDAPut(DA, 1, 1, 2, "D:2, 1234567890")  
Call cDAPut(DA, 1, DA.nCols, 2, "D:2, 0987654321")  
Call cDAPut(DA, DA.nRows, 1, 2, "D:2, 12345ABCDE")  
Call cDAPut(DA, DA.nRows, DA.nCols, 2, "D:2, VWXYZ54321")

'save the string in Row 1, Col 1, Sheet 2  
'save the string in Row 1, Col 500, Sheet 2  
'save the string in Row 500, Col 1, Sheet 2  
'save the string in Row 500, Col 500, Sheet 2

Var(1) = cDAGet(DA, 1, 1, 1)  
Var(2) = cDAGet(DA, 1, DA.nCols, 1)  
Var(3) = cDAGet(DA, DA.nRows, 1, 1)  
Var(4) = cDAGet(DA, DA.nRows, DA.nCols, 1)

'read the string in Row 1, Col 1, Sheet 1  
'read the string in Row 1, Col 500, Sheet 1  
'read the string in Row 500, Col 1, Sheet 1  
'read the string in Row 500, Col 500, Sheet 1

Var(5) = cDAGet(DA, 1, 1, 2)  
Var(6) = cDAGet(DA, 1, DA.nCols, 2)  
Var(7) = cDAGet(DA, DA.nRows, 1, 2)  
Var(8) = cDAGet(DA, DA.nRows, DA.nCols, 2)

'read the string in Row 1, Col 1, Sheet 2  
'read the string in Row 1, Col 500, Sheet 2  
'read the string in Row 500, Col 1, Sheet 2  
'read the string in Row 500, Col 500, Sheet 2

Call cDAClose(DA, False)

'close the file without delete it.

On my system :

ErrCode = -1

'no error

DA.daSize = 128  
DA.Signature = "MCR\_347"  
DA.nFilename = "c:\t2w\_tmp\dastring.tmp"  
DA.nType = 50  
DA.nRows = 500  
DA.nCols = 500  
DA.nSheets = 2  
DA.rHandle = 0  
DA.rElementSize = 50  
DA.rFileSize = 25000128  
DA.rParts = 762  
chars)  
DA.rRemain = 30784  
DA.rSheetSize = 250000  
DA.rTime = 26639

'internal header size  
'internal signature  
'name fo the file  
'fixed string of 50 chars  
'500 rows  
'500 cols  
'2 sheets  
'internal handle  
'internal size of a element  
'internal size of the file  
'internal number of parts (block of 32768  
chars)  
'internal remain chars  
'internal size of one sheet  
'internal time to perform the operation

Var(1) = "D:1, ABCDEFGHIJ"  
Var(2) = "D:1, abcdefghij"  
Var(3) = "D:1, OPQRSTUVWXYZ"  
Var(4) = "D:1, oprqstuvwxyz"

Var(5) = "D:2, 1234567890"  
Var(6) = "D:2, 0987654321"  
Var(7) = "D:2, 12345ABCDE"  
Var(8) = "D:2, VWXYZ54321"

**See also :** [Disk Array](#)

# DAClose

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[DAClose](#) close a big sized array and keep it or close a big sized array and destroy it.

## **Declare Syntax :**

Declare Sub cDAClose Lib "[time2win.dll](#)" (DISKARRAY As tagDISKARRAY, ByVal DeleteFile As Integer)

## **Call Syntax :**

Call cDAClose(DISKARRAY, DeleteFile%)

## **Where :**

DISKARRAY	is a type'd variable (tagDISKARRAY).
DeleteFile%	TRUE : delete the file FALSE : don't delete the file (the file can be re-used by cDACreate)

## **Comments :**

If you want to re-use the big sized array on disk with the same parameters and whitout a new initialization, don't delete it.

## **Examples :**

see [DACreate](#)

**See also :** [Disk Array](#)

# MDAClose

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[MDAClose](#) close a multiple big sized array and keep it or close a multiple big sized array and destroy it.

## **Declare Syntax :**

Declare Sub cMDAClose Lib "[time2win.dll](#)" (MULTIPLDISKARRAY As tagMULTIPLDISKARRAY, ByVal DeleteFile As Integer)

## **Call Syntax :**

Call cMDAClose(MULTIPLDISKARRAY, DeleteFile%)

## **Where :**

MULTIPLDISKARRAY	is a type'd variable (tagMULTIPLDISKARRAY).
DeleteFile%	TRUE : delete the file FALSE : don't delete the file (the file can be re-used by cMDACreate)

## **Comments :**

If you want to re-use the multiple big sized array on disk with the same parameters and whitout a new initialization, don't delete it.

## **Examples :**

see [MDACreate](#)

**See also :** [Multiple Disk Array](#)



# MDAClear, MDAClearSheet, MDAClearCol, MDAsClearCol, MDAClearRow, MDAsClearRow

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[MDAClear](#) clear a multiple big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).  
[MDAClearSheet](#) clear a single Sheet in a multiple big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).  
[MDAClearCol](#) clear a single Col on one Sheet or on all Sheets in a multiple big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).  
[MDAsClearCol](#) have the same functionality but with a multiple big sized array with only one sheet.  
[MDAClearRow](#) clear a single Row on one Sheet or on all Sheets in a multiple big sized array (fill it with chr\$(0) or chr\$(32) (for string array)).  
[MDAsClearRow](#) have the same functionality but with a multiple big sized array with only one sheet.

## **Declare Syntax :**

```
Declare Function cMDAClear Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY) As Integer
Declare Function cMDAClearSheet Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Sheet As Long) As Integer
Declare Function cMDAClearCol Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long, ByVal Sheet As Long) As Integer
Declare Function cMDAsClearCol Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long) As Integer
Declare Function cMDAClearRow Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Sheet As Long) As Integer
Declare Function cMDAsClearRow Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long) As Integer
```

## **Call Syntax :**

```
ErrCode% = cMDAClear(Array%, MULTIPLEDISKARRAY)
ErrCode% = cMDAClearSheet(Array%, MULTIPLEDISKARRAY, Sheet&)
ErrCode% = cMDAClearCol(Array%, MULTIPLEDISKARRAY, Col&, Sheet&)
ErrCode% = cMDAsClearCol(Array%, MULTIPLEDISKARRAY, Col&)
ErrCode% = cMDAClearRow(Array%, MULTIPLEDISKARRAY, Row&, Sheet&)
ErrCode% = cMDAsClearRow(Array%, MULTIPLEDISKARRAY, Row&)
```

## **Where :**

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Col&	is the desired Col.
Row&	is the desired Row.
Sheet&	is the desired Sheet.
ErrCode%	is the returned <u>error code</u> .

## **Comments :**

This function must be used only after you've created a multiple big sized array on disk OR after the using of an existing multiple big sized array on disk.

If you've created a multiple big sized array on disk, the array is already cleared.

For MDAClearSheet :

If the multiple big sized array on disk have a single Sheet, this routine have the same effect that cMDAClear.

If the Sheet is -1 then all Sheets are used. This parameter have the same fonctionnality that cMDAClear

If the Sheet is below 1 and different of -1, the Sheet 1 is used.

If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

For MDAClearCol, MDAsClearCol :

If the Col is below 1, the Col 1 is used.

If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.

If the Sheet is -1 then all Sheets are used.

If the Sheet is below 1 and different of -1, the Sheet 1 is used.

If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

For MDAClearRow, MDAsClearRow :

If the Row is below 1, the Row 1 is used.

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.

If the Sheet is -1 then all Sheets are used.

If the Sheet is below 1 and different of -1, the Sheet 1 is used.

If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

### Examples :

```
Dim ErrCode           As Integer
Dim MDA               As tagMULTIPLEDISKARRAY

MDA.nFilename = "c:\t2w_tmp\mda.tmp"           'name of the file to use
MDA.nType(1) = 50                             'positive value for a string
MDA.nIsTyped(1) = False                       'init the array with spaces
MDA.nRows(1) = 500                            '500 rows
MDA.nCols(1) = 500                            '500 cols
MDA.nSheets(1) = 2                            '2 sheets

ErrCode = cMDACreate(MDA, True)                'create a new multiple big sized
array on disk

Call cMDAPut(1, MDA, 1, 1, 1, "D:1, ABCDEFGHIJ") 'save the string in Row 1, Col 1,
Sheet 1, Array 1
Call cMDAPut(1, MDA, 1, MDA.nCols(1), 1, "D:1, abcdefghij") 'save the string in Row 1, Col 500,
Sheet 1, Array 1
Call cMDAPut(1, MDA, MDA.nRows(1), 1, 1, "D:1, OPQRSTUVWXYZ") 'save the string in Row 500, Col 1,
Sheet 1, Array 1
Call cMDAPut(1, MDA, MDA.nRows(1), MDA.nCols(1), 1, "D:1, oprqstuvwxyz") 'save the string in Row 500, Col
500, Sheet 1, Array 1

'..... some codes

ErrCode = cMDAClear(1, MDA)                   'clear all elements in the multiple
big sized array on disk

ErrCode = cMDAClearSheet(1, MDA, 1)          'clear the Sheet 1 in the multiple big
sized array on disk
```

ErrCode = cMDAClearCol(1, MDA, MDA.nCols(1), 2)  
big sized array on disk  
ErrCode = cMDAsClearCol(1, MDA, MDA.nCols(1))  
big sized array on disk

'clear the last Col in Sheet 2 in the  
'clear the last Col in Sheet 1 in the

ErrCode = cMDAClearRow(1, MDA, MDA.nRows(1), 2)  
big sized array on disk  
ErrCode = cMDAsClearRow(1, MDA, MDA.nRows(1), 1)  
big sized array on disk

'clear the last Row in Sheet 2 in the  
'clear the last Row in Sheet 1 in the

**See also :** [Multiple Disk Array](#)

# Multiple disk array : Overview

The functions/subs used in the Multiple Disk Array routines handle big sized arrays on disk in only file.

Each array use only a file to handle the information. A file can contain 20 big sized arrays.

The concept of big sized arrays on disk is to use the mass storage (hard disk) in place of memory. This concept minimize the use of the memory for big array but decrease the speed to accessing data.

A fixed string array of 500 rows by 500 cols, 2 Sheets and a string size of 50 take 25.000.000 bytes. I think that this is better to place this array on the disk.

The following functions/subs are used to handle big sized arrays on disk :

<u>MDAClear</u>	clear a multiple big sized array.
<u>MDAClearCol</u>	clear a single col on on a sheet in a multiple big sized array.
<u>MDAClearRow</u>	clear a single row on a sheet in a multiple big sized array.
<u>MDAClearSheet</u>	clear a single sheet in a multiple big sized array.
<u>MDAClose</u>	close a big sized array and keep it or close a multiple big sized array and destroy it.
<u>MDACreate</u>	create a new big sized array on disk or use an existing multiple big sized array on disk.
<u>MDAGet</u>	read an element from a multiple big sized array on disk.
<u>MDAGGetType</u>	read a type'd variable from a multiple big sized array on disk.
<u>MDAPut</u>	save an element to a multiple big sized array on disk.
<u>MDAPutType</u>	save a type'd variable to a multiple big sized array on disk.
<u>MDArGet</u>	read an element from a multiple big sized array on disk with only one sheet and one row.
<u>MDArGetType</u>	read a type'd variable from a big sized array on disk with only one sheet and one row.
<u>MDArPut</u>	save an element to a multiple big sized array on disk with only one sheet and one row.
<u>MDArPutType</u>	save a type'd variable to a multiple big sized array on disk with only one sheet and one row.
<u>MDAsClearCol</u>	clear a single col on on a sheet in a multiple big sized array with only one sheet.
<u>MDAsClearRow</u>	clear a single row on a sheet in a multiple big sized array with only one sheet.
<u>MDAsGet</u>	read an element from a multiple big sized array on disk with only one sheet.
<u>MDAsGetType</u>	read a type'd variable from a multiple big sized array on disk with only one sheet.
<u>MDAsPut</u>	save an element to a multiple big sized array on disk with only one sheet.
<u>MDAsPutType</u>	save a type'd variable to a multiple big sized array on disk with only one sheet.

To minimize the use of too many functions for the different variable type in VB, MDAGet and MDAPut uses variant value (integer, long, single, double, currency, string). This can be slow down (a little bit) the speed for accessing the data.

To handle type'd variable, you must use MDAGGetType, MDAPutType.

When you create a new multiple array on disk, a header (640 chars for VB 3.0 and VB 4.0 (16-Bit)) is writed to begin of the associated file. This header is readed when you re-use an existing array to verify that this is a good big sized disk array.

Actually, the maximum number of chars for a string element or for a type'd variable is [4096](#).

# MDACreate

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[MDACreate](#) create a multiple new big sized array on disk or use an existing multiple big sized array on disk.

## Declare Syntax :

Declare Function cMDACreate Lib "time2win.dll" (MULTIPLDISKARRAY As tagMULTIPLDISKARRAY, ByVal CreateOrUse As Integer) As Integer

## Call Syntax :

ErrCode% = cMDACreate(MDA, CreateOrUse%)

## Where :

MULTIPLDISKARRAY is a type'd variable (tagMULTIPLDISKARRAY).  
CreateOrUse% TRUE : if you want to create a new big sized array on disk,  
FALSE : if you want to re-use an existing big sized array on disk.  
ErrCode% is the returned error code.

## Comments :

In theory :

The maximum number of Arrays is 20  
The maximum number of Rows is 2147483647  
The maximum number of Cols is 2147483647  
The maximum number of Sheets is 2147483647

You are only limited by the size of the disk on which the big sized array are defined.

The length of the filename can be 64 chars for VB 3.0 and VB 4.0 (16-Bit), 128 chars for VB 4.0 (32-Bit) maximum.

If you create a new multiple big sized array on disk and if the file is already exists, the file is deleted before used.  
If you re-use an existing multiple big sized array on disk, some checkings are made to verify the validity of the multiple big sized array on disk.

Bigger are nRows, nCols or nSheets, bigger is the time to initialize.

When you create a new multiple big sized array on disk, the only parameters that you must initialize are :

DA.nFilename = "c:\t2w_tmp\dastring.tmp"	'name of the file (you must have enough space on the drive).
DA.nType(1) = 50	'the type of the variable to use, see <u>Constants and</u>
<u>Types declaration.</u> (DA_x)	
DA.nIsTyped(1) = False	'Must be True for a type'd variable for Array 1.
DA.nRows(1) = 500	'the number of rows to use for Array 1.
DA.nCols(1) = 500	'the number of cols to use for Array 1.
DA.nSheets(1) = 2	'the number of sheets to use for Array 1.
.../...	
DA.nType(20) = 25	'the type of the variable to use, see <u>Constants and</u>
<u>Types declaration.</u> (DA_x)	
DA.nIsTyped(20) = False	'Must be True for a type'd variable for Array 20.
DA.nRows(20) = 500	'the number of rows to use for Array 20.
DA.nCols(20) = 500	'the number of cols to use for Array 20.
DA.nSheets(20) = 2	'the number of sheets to use for Array 20.

**YOU CAN'T CHANGE THESE PARAMETERS AFTER THE CREATION OF THE MULTIPLE BIG SIZED ARRAY.**

**YOU CAN'T CHANGE THE OTHER VALUES IN THE TYPE'D VARIABLE.**

**Don't forget that you create the multiple array of maximum 20 arrays in one call. The order is not important, but you must take in mind that if you use only 3 arrays on the 20, there are only initialization for these 3 arrays and you can't insert other arrays.**

When you create a new array, all elements are initialized with chr\$(0) except for string array which are initialized with chr\$(32) (spaces).

However, string array and type'd array use the same positive value to define in .nType, but the type'd array must be initialized with chr\$(0) not with chr\$(32) therefore for a type'd you must specify .nlsTyped on True to initialize it with chr\$(0).

If you use multiple big size array of type'd variable, the type'd variable can be only a mix of fixed variable (variable string length can't be used).

### Examples :

```
Dim ErrCode           As Integer
Dim MDA               As tagMULTIPLEDISKARRAY
Dim Var(1 To 8)       As Variant
```

```
DA.nFilename = "c:\t2w_tmp\dastring.tmp"      'name of the file to use
DA.nType(1) = 50                              'positive value for a string (array 1)
DA.nlsTyped(1) = False                       'init the array with spaces (array 1)
DA.nRows(1) = 500                            '500 rows (array 1)
DA.nCols(1) = 500                            '500 cols (array 1)
DA.nSheets(1) = 2                            '2 sheets (array 1)
```

```
DA.nType(9) = 25                              'positive value for a string (array 9)
DA.nlsTyped(9) = False                       'init the array with spaces (array 9)
DA.nRows(9) = 100                            '100 rows (array 9)
DA.nCols(9) = 100                            '100 cols (array 9)
DA.nSheets(9) = 5                            '5 sheets (array 9)
```

```
ErrCode = cMDACreate(MDA, True)              'create a new multiple big sized
array on disk
```

```
Call cMDAPut(1, MDA, 1, 1, 1, "D:1, ABCDEFGHIJ") 'save the string in Row 1, Col 1,
Sheet 1, Array 1
```

```
Call cMDAPut(1, MDA, 1, DA.nCols(1), 1, "D:1, abcdefghij") 'save the string in Row 1, Col 500,
Sheet 1, Array 1
```

```
Call cMDAPut(1, MDA, DA.nRows(1), 1, 1, "D:1, OPQRSTUVWXYZ") 'save the string in Row 500, Col 1,
Sheet 1, Array 1
```

```
Call cMDAPut(1, MDA, DA.nRows(1), DA.nCols(1), 1, "D:1, oprqstuvwxyz") 'save the string in Row 500, Col
500, Sheet 1, Array 1
```

```
Call cMDAPut(9, MDA, 1, 1, 5, "D:2, 1234567890") 'save the string in Row 1, Col 1,
Sheet 5, Array 9
```

```
Call cMDAPut(9, MDA, 1, MDA.nCols(9), 5, "D:2, 0987654321") 'save the string in Row 1, Col 100,
Sheet 5, Array 9
```

```
Call cMDAPut(9, MDA, MDA.nRows(9), 1, 5, "D:2, 12345ABCDE") 'save the string in Row 100, Col 1,
Sheet 5, Array 9
```

```
Call cMDAPut(9, MDA, MDA.nRows(9), MDA.nCols(9), 5, "D:2, VWXYZ54321") 'save the string in Row 100, Col
100, Sheet 5, Array 9
```

```
Var(1) = cMDAGet(1, MDA, 1, 1, 1)            'read the string in Row 1, Col 1,
Sheet 1, Array 1
```

```
Var(2) = cMDAGet(1, MDA, 1, MDA.nCols(1), 1) 'read the string in Row 1, Col 500,
Sheet 1, Array 1
```

Var(3) = cMDAGet(1, MDA, MDA.nRows(1), 1, 1)  
Sheet 1, Array 1  
Var(4) = cMDAGet(1, MDA, MDA.nRows(1), MDA.nCols(1), 1)  
500, Sheet 1, Array 1

'read the string in Row 500, Col 1,

'read the string in Row 500, Col

Var(5) = cMDAGet(9, MDA, 1, 1, 5)  
Sheet 5, Array 9

'read the string in Row 1, Col 1,

Var(6) = cMDAGet(9, MDA, 1, MDA.nCols(9), 5)  
Sheet 5, Array 9

'read the string in Row 1, Col 100,

Var(7) = cMDAGet(9, MDA, MDA.nRows(9), 1, 5)  
Sheet 5, Array 9

'read the string in Row 100, Col 1,

Var(8) = cMDAGet(9, MDA, MDA.nRows(9), MDA.nCols(9), 5)  
100, Sheet 5, Array 9

'read the string in Row 100, Col

Call cMDAClose(MDA, False)

'close the file without delete it.

**See also :** [Multiple Disk Array](#)

# MDAGet, MDAsGet, MDARGet

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[MDAGet](#) read an element from a multiple big sized array on disk.

[MDARGet](#) have the same fonctionnality but with a multiple big sized array with only one sheet and only one row.

[MDAsGet](#) have the same fonctionnality but with a multiple big sized array with only one sheet.

## Declare Syntax :

Declare Function cMDAGet Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long) As Variant

Declare Function cMDARGet Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long) As Variant

Declare Function cMDAsGet Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long) As Variant

## Call Syntax :

Var = cMDAGet(Array%, MULTIPLEDISKARRAY, Row&, Col&, Sheet&)

Var = cMDAGet(Array%, MULTIPLEDISKARRAY, Col&, Sheet&)

Var = cMDAGet(Array%, MULTIPLEDISKARRAY, Row&, Col&)

## Where :

MULTIPLEDISKARRAY is a type'd variable (tagMULTIPLEDISKARRAY).

Array% is the array in the multiple array (must be between 1 to 20).

Row& is the row.

Col& is the col.

Sheet& is the sheet.

Var is the readed variant value depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.

If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.

If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

## Examples :

see [MDACreate](#)

**See also :** [Multiple Disk Array](#), [MDAPut](#)



# MDAPut, MDAsPut, MDARPut

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[MDAPut](#) save an element to a big sized array on disk.

[MDARPut](#) have the same functionality but with a multiple big sized array with only one sheet and only one row.

[MDAsPut](#) have the same functionality but with a multiple big sized array with only one sheet.

## Declare Syntax :

Declare Sub cMDAPut Lib "[time2win.dll](#)" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, Var As Variant)

Declare Sub cMDARPut Lib "[time2win.dll](#)" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long) As Variant

Declare Sub cMDAsPut Lib "[time2win.dll](#)" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, Var As Variant)

## Call Syntax :

Call cMDAPut(Array%, MULTIPLEDISKARRAY, Row&, Col&, Sheet&, Var)

Call cMDARPut(Array%, MULTIPLEDISKARRAY, Col&, Var)

Call cMDAsPut(Array%, MULTIPLEDISKARRAY, Row&, Col&, Var)

## Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
Var	is the variant value to save depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.

If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.

If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

## Examples :

see [MDACreate](#)

**See also :** [Multiple Disk Array](#), [MDAGet](#)

# MDAGetType, MDAsGetType, MDARGetType

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**MDAGetType** read a type'd variable from a multiple big sized array on disk.

**MDARGetType** have the same fonctionnality but with a multiple big sized array with only one sheet and only one row.

**MDAsGetType** have the same fonctionnality but with a multiple big sized array with only one sheet.

## Declare Syntax :

```
Declare Sub cMDAGetType Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)
```

```
Declare Sub cMDARGetType Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long, nType As Any)
```

```
Declare Sub cMDAsGetType Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)
```

## Call Syntax :

```
Call cMDAGetType(Array%, MULTIPLEDISKARRAY, Row&, Col&, Sheet&, nType)
```

```
Call cMDARGetType(Array%, MULTIPLEDISKARRAY, Col&, nType)
```

```
Call cMDAsGetType(Array%, MULTIPLEDISKARRAY, Row&, Col&, nType%)
```

## Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
nType	is the readed type'd variable depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.

If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.

If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

## Examples :

```
Dim ErrCode           As Integer
Dim MDA               As tagMULTIPLEDISKARRAY
Dim TE(1 To 4)       As tagTASKENTRY
```

MDA.nFilename = "c:\t2w_tmp\datatype.tmp"	'name of the file to use
MDA.nType(1) = Len(TE(1))	'positive value for a type'd variable
MDA.nIsTyped(1) = True	'init the array with chr\$(0) because type'd variable
MDA.nRows(1) = 500	'500 rows
MDA.nCols(1) = 500	'500 cols
MDA.nSheets(1) = 2	'2 sheets

ErrCode = cMDACreate(MDA, False)  
disk

'use a created multiple big sized array on

Call cDAGetType(1, MDA, 1, 1, 1, TE(1))  
Sheet 1, Array 1.

'read the type'd variable in Row 1, Col 1,

Call cDAGetType(1, MDA, 1, DA.nCols(1), 1, TE(2))  
Sheet 1, Array 1.

'read the type'd variable in Row 1, Col 500,

Call cDAGetType(1, MDA, MDA.nRows(1), 1, 1, TE(3))  
Sheet 1, Array 1.

'read the type'd variable in Row 500, Col 1,

Call cDAGetType(1, MDA, MDA.nRows(1), MDA.nCols(1), 1, TE(4))  
500, Sheet 1, Array 1.

'read the type'd variable in Row 500, Col

**See also :** [Multiple Disk Array](#), [MDAPutType](#)

# MDAPutType, MDAsPutType, MDAPutType

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**MDAPutType** save a type'd variable from a big sized array on disk.

**MDArPutType** have the same fonctionnality but with a big sized array with only one sheet and only one row.

**MDAsPutType** have the same fonctionnality but with a big sized array with only one sheet.

## Declare Syntax :

Declare Sub cMDAPutType Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)

Declare Sub cMDArPutType Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long, nType As Any)

Declare Sub cMDAsPutType Lib "time2win.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)

## Call Syntax :

Call cMDAPutType(Array%, MULTIPLEDISKARRAY, Row&, Col&, Sheet&, nType)

Call cMDArPutType(Array%, MULTIPLEDISKARRAY, Row&, nType)

Call cMDAsPutType(Array%, MULTIPLEDISKARRAY, Row&, Col&, nType)

## Where :

MULTIPLEDISKARRAY is a type'd variable (tagMULTIPLEDISKARRAY).  
Array% is the array in the multiple array (must be between 1 to 20).  
Row& is the row.  
Col& is the col.  
Sheet& is the sheet.  
nType is the type'd variable to save depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.

If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.

If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

## Examples :

Dim ErrCode As Integer  
Dim MDA As tagMULTIPLEDISKARRAY  
Dim TE As tagTASKENTRY

DA.nFilename = "c:\t2w\_tmp\datype.tmp" 'name of the file to use  
DA.nType(1) = Len(TE) 'positive value for a type'd variable  
DA.nIsTyped(1) = True 'init the array with chr\$(0) because type'd variable  
DA.nRows(1) = 500 '500 rows  
DA.nCols(1) = 500 '500 cols  
DA.nSheets(1) = 2 '2 sheets

ErrCode = cMDACreate(MDA, True)	'create a new multiple big sized array on disk
ErrCode = cTasks(TE, True)	
Call cMDAPutType(1, MDA, 1, 1, 1, TE)	'save the type'd variable in Row 1, Col 1,
Sheet 1, Array 1.	
ErrCode = cTasks(TE, False)	
Call cMDAPutType(1, MDA, 1, MDA.nCols(1), 1, TE)	'save the type'd variable in Row 1, Col 500,
Sheet 1, Array 1.	
ErrCode = cTasks(TE, False)	
Call cMDAPutType(1, MDA, MDA.nRows(1), 1, 1, TE)	'save the type'd variable in Row 500, Col 1,
Sheet 1, Array 1.	
ErrCode = cTasks(TE, False)	
Call cMDAPutType(1, MDA, MDA.nRows(1), MDA.nCols(1), 1, TE)	'save the type'd variable in Row 500, Col
500, Sheet 1, Array 1.	

**See also :** [Multiple Disk Array](#), [MDAGetType](#)

# FromBinary, FromBinary2, ToBinary, ToBinary2

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FromBinary](#) converts a binary string (0, 1) to a string

[FromBinary2](#) converts a binary string (custom letters) to a string

[ToBinary](#) converts a string to a binary representation with 0, 1

[ToBinary2](#) converts a string to a binary representation with two custom letters for 0, 1 representation

## Declare Syntax :

Declare Function cFromBinary Lib "time2win.dll" (Text As String) As String

Declare Function cFromBinary2 Lib "time2win.dll" (Text As String, Bin As String) As String

Declare Function cToBinary Lib "time2win.dll" (Text As String) As String

Declare Function cToBinary2 Lib "time2win.dll" (Text As String, Bin As String) As String

## Call Syntax :

```
test$ = cFromBinary(Text)
```

```
test$ = cFromBinary2(Text, Bin)
```

```
test$ = cToBinary(Text)
```

```
test$ = cToBinary2(Text, Bin)
```

## Where :

Text	the string to proceed
Bin	the two custom letters for 0, 1 representation
test\$	the result

## Comments :

## Examples :

```
test$ = cToBinary("MC")
```

```
-> "0100110101000011"
```

```
test$ = cToBinary2("MC","mc")
```

```
-> "cmccmmcmcmccmm"
```

```
test$ = cFromBinary("0100110101000011")
```

```
-> "MC"
```

```
test$ = cFromBinary2("cmccmmcmcmccmm","mc") -> "MC"
```

**See also** : [Binary](#)

# 2-D Geometry : Overview

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[V2Add](#) add two 2D vectors.

[V2Sub](#) subtract two 2D vectors.

[V2Combine](#) combine two 2D vectors.

[V2Copy](#) copy a 2D vector into an another.

[V2Dot](#) calculate the dot of two 2D vectors.

[V2Length](#) calculate the length (magnitude) of a 2D vector.

[V2Length](#) calculate the length squared (magnitude squared) of a 2D vector.

[V2LinearLp](#) perform the linear interpolation of two 2D vector.

[V2Mul](#) multiply two 2D vector.

[V2Neg](#) perform the negate of a 2D vector.

[V2Normalized](#) normalize a 2D vector.

[V2Ortho](#) perform the orthogonal transformation of two 2D vector.

[V2ScaledNewLength](#) change the x,y of a 2D vector from a new length (magnitude).

[V2SegmentLength](#) calculate the length of the segment between the two 2D vector.

## Declare Syntax :

```
Declare Sub cV2Add Lib "time2win.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
```

```
Declare Sub cV2Sub Lib "time2win.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
```

```
Declare Sub cV2Combine Lib "time2win.dll" (u As tagVECTOR2, ByVal c1 As Double, v As tagVECTOR2, ByVal c2 As Double, w As tagVECTOR2)
```

```
Declare Sub cV2Copy Lib "time2win.dll" (u As tagVECTOR2, w As tagVECTOR2)
```

```
Declare Function cV2Dot Lib "time2win.dll" (u As tagVECTOR2, v As tagVECTOR2) As Double
```

```
Declare Function cV2Length Lib "time2win.dll" (u As tagVECTOR2) As Double
```

```
Declare Function cV2LengthSquared Lib "time2win.dll" (u As tagVECTOR2) As Double
```

```
Declare Sub cV2LinearLp Lib "time2win.dll" (lo As tagVECTOR2, hi As tagVECTOR2, ByVal alpha As Double, w As tagVECTOR2)
```

```
Declare Sub cV2Mul Lib "time2win.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
```

```
Declare Sub cV2Neg Lib "time2win.dll" (u As tagVECTOR2)
```

```
Declare Sub cV2Normalized Lib "time2win.dll" (u As tagVECTOR2)
```

```
Declare Sub cV2Ortho Lib "time2win.dll" (u As tagVECTOR2, w As tagVECTOR2)
```

```
Declare Sub cV2ScaledNewLength Lib "time2win.dll" (u As tagVECTOR2, ByVal newlen As Double)
```

```
Declare Function cV2SegmentLength Lib "time2win.dll" (p As tagVECTOR2, q As tagVECTOR2) As Double
```

## Call Syntax :

## Where :

## Comments :

## Examples :

**See also :** [3-D Geometry](#)

# 3-D Geometry : Overview

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[V3Add](#) add two 3D vectors.

[V3Sub](#) subtract two 3D vectors.

[V3Combine](#) combine two 3D vectors.

[V3Copy](#) copy a 3D vector into an another.

[V3Dot](#) calculate the dot of two 3D vectors.

[V3Length](#) calculate the length (magnitude) of a 3D vector.

[V3Length](#) calculate the length squared (magnitude squared) of a 3D vector.

[V3LinearLp](#) perform the linear interpolation of two 3D vector.

[V3Mul](#) multiply two 3D vector.

[V3Neg](#) perform the negate of a 3D vector.

[V3Normalized](#) normalize a 3D vector.

[V3Ortho](#) perform the orthogonal transformation of two 3D vector.

[V3ScaledNewLength](#) change the x,y of a 3D vector from a new length (magnitude).

[V3SegmentLength](#) calculate the length of the segment between the two 3D vector.

[3DWeightAverage](#) calculate the z value of an additional point from four points.

## Declare Syntax :

```
Declare Sub cV3Add Lib "time2win.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
```

```
Declare Sub cV3Sub Lib "time2win.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
```

```
Declare Sub cV3Combine Lib "time2win.dll" (u As tagVECTOR3, ByVal c1 As Double, v As tagVECTOR3, ByVal c2 As Double, w As tagVECTOR3)
```

```
Declare Sub cV3Copy Lib "time2win.dll" (u As tagVECTOR3, w As tagVECTOR3)
```

```
Declare Sub cV3Cross Lib "time2win.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
```

```
Declare Function cV3Dot Lib "time2win.dll" (u As tagVECTOR3, v As tagVECTOR3) As Double
```

```
Declare Function cV3Length Lib "time2win.dll" (u As tagVECTOR3) As Double
```

```
Declare Function cV3LengthSquared Lib "time2win.dll" (u As tagVECTOR3) As Double
```

```
Declare Sub cV3LinearLp Lib "time2win.dll" (lo As tagVECTOR3, hi As tagVECTOR3, ByVal alpha As Double, w As tagVECTOR3)
```

```
Declare Sub cV3Mul Lib "time2win.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
```

```
Declare Sub cV3Neg Lib "time2win.dll" (u As tagVECTOR3)
```

```
Declare Sub cV3Normalized Lib "time2win.dll" (u As tagVECTOR3)
```

```
Declare Sub cV3ScaledNewLength Lib "time2win.dll" (u As tagVECTOR3, ByVal newlen As Double)
```

```
Declare Function cV3SegmentLength Lib "time2win.dll" (p As tagVECTOR3, q As tagVECTOR3) As Double
```

```
Declare Function c3DWeightAverage Lib "time2win.dll" (u13D As tagVECTOR3, l13D As tagVECTOR3, l3D As tagVECTOR3, ur3D As tagVECTOR3, ptToLocate3D As tagVECTOR3) As Double
```

## Call Syntax :

## Where :

## Comments :

## Examples :

For 3DWeightAverage :

```
Dim uLeft3D          As tagVECTOR3
Dim lLeft3D          As tagVECTOR3
```



Dim IRight3D           As tagVECTOR3  
Dim uRight3D           As tagVECTOR3  
Dim ptToLocate3D       As tagVECTOR3

uLeft3D.x = 11  
uLeft3D.y = 21  
uLeft3D.z = 20

lLeft3D.x = 11  
lLeft3D.y = 15  
lLeft3D.z = 17

IRight3D.x = 17  
IRight3D.y = 15  
IRight3D.z = 21

uRight3D.x = 17  
uRight3D.y = 21  
uRight3D.z = 30

ptToLocate3D.x = 15  
ptToLocate3D.y = 20  
ptToLocate3D.z = 0

Debug.Print c3DWeightAverage(uLeft3D, lLeft3D, IRight3D, uRight3D, ptToLocate3D)       '->  
24,0609108270454

ptToLocate3D.x = 15  
ptToLocate3D.y = 19  
ptToLocate3D.z = 0

Debug.Print c3DWeightAverage(uLeft3D, lLeft3D, IRight3D, uRight3D, ptToLocate3D)       '->  
23,3029834668893

**See also :** [2-D Geometry](#)

# TimerClose, TimerOpen, TimerRead, TimerStart

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**TimerOpen** open a timer and return an handle of an available timer (1 to 64).

**TimerStart** start the selected timer's handle.

**TimerRead** read the current value of the selected timer's handle.

**TimerClose** close the selected timer's handle.

## Declare Syntax :

```
Declare Function cTimerOpen Lib "time2win.dll" () As Integer
Declare Function cTimerStart Lib "time2win.dll" (ByVal TimerHandle As Long) As Integer
Declare Function cTimerRead Lib "time2win.dll" (ByVal TimerHandle As Long) As Long
Declare Function cTimerClose Lib "time2win.dll" (ByVal TimerHandle As Long) As Integer
```

## Call Syntax :

```
TimerHandle% = cTimerOpen()
StartOk% = cTimerStart(TimerHandle%)
Test& = cTimerRead(TimerHandle%)
CloseOk% = cTimerClose(TimerHandle%)
```

## Where :

TimerHandle%	>0 is one timer is available, = 0 if no timers available..
StartOk%	TRUE if the starting is successfully, FALSE if the starting fail.
Test&	is the current value of the specified timer handle.
CloseOk%	TRUE if the closing is successfully, FALSE if the closing fail.

## Comments :

These timers functions is independant of the calling program.

The value of timers is in milliseconds.

The accuracy of timers is 1 milliseconds.

## Examples :

```
Dim TimerHandle As Integer
Dim TimerValue As Long

Dim i As Long
Dim n As Long
Dim StartOk As Integer
Dim CloseOk As Integer
```

```
TimerHandle = cTimerOpen()
StartOk = cTimerStart(TimerHandle)
```

```
For i = 1 To 54321
    n = i * 2
Next i
```

```
MsgBox "Time (in milliseconds) to perform the test is " & cTimerRead(TimerHandle) & " milliseconds"
```

```
CloseOk = cTimerClose(TimerHandle)
```

' On my system : "Time (in milliseconds) to perform the test is 330"

**See also :** [Timer](#)

# Timer : Overview

Timer functions performs timing functions for your application. These functions are divided in two parts :

1) Timing which use the GetTickCount() have an accuracy of **55** ms, these functions are available for all applications in memory and share the same memory space. You can have 32 timers. Be carefully, when distributing the DLL on an other computer did use the same DLL.

<a href="#">CheckWait</a>	check if the specified timer has reached the time to wait.
<a href="#">ReadBasisTimer</a>	read the value of the default timer.
<a href="#">ReadTimer</a>	read the value of the specified timer.
<a href="#">SetWait</a>	set the time to wait in a specified timer.
<a href="#">Sleep</a>	suspend the current execution of a routine for a given delay.
<a href="#">StartBasisTimer</a>	start the default timer.
<a href="#">StartTimer</a>	start the specified timer.
<a href="#">StartWait</a>	start the specified timer.
<a href="#">StopBasisTimer</a>	stop the value of the default timer.
<a href="#">StopTimer</a>	stop the value of the specified timer.

2) Timing which use the TimerCountt() have an accuracy of 1 ms, these functions use the concept of handle to permit to have many different application which can use the DLL. You can have 64 handles.

<a href="#">TimerOpen</a>	open a timer and return an handle of an available timer (1 to 64).
<a href="#">TimerStart</a>	start the selected timer's handle.
<a href="#">TimerRead</a>	read the current value of the selected timer's handle.
<a href="#">TimerClose</a>	close the selected timer's handle.

# Sleep

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**Sleep** suspend the current execution of a routine for a given delay.

## Declare Syntax :

Declare Function cSleep Lib "time2win.dll" (ByVal Delay As Long) As Integer

## Call Syntax :

status% = cSleep(Delay)

## Where :

Delay	is the time to sleep the current execution of a routine in milliseconds.
status%	TRUE if all is OK FALSE if the delay is below 0.

## Comments :

Use this function with care.  
Don't set a delay to bigger.  
Don't forget that the delay is in milliseconds.

## Examples :

Dim status As Integer

status% = cSleep(-10)	-> Don't sleep, the delay is negative value.
status% = cSleep(0)	-> A very short sleeping.
status% = cSleep(7000)	-> Sleep for 7 seconds

Call cStartBasisTimer  
status = cSleep(7000)  
MsgBox "Time elapsed for the current sleeping is " & cReadBasisTimer() & " milliseconds"

' On my system : "Time elapsed for the current sleeping is 7031 milliseconds"

**See also :** [Timer](#)

# ReadBasisTimer, StartBasisTimer, StopBasisTimer

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[StartBasisTimer](#) start the default timer.

[ReadBasisTimer](#) read the value of the default timer.

[StopBasisTimer](#) stop the value of the default timer.

## **Declare Syntax :**

```
Declare Sub cStartBasisTimer Lib "time2win.dll" ()
Declare Function cReadBasisTimer Lib "time2win.dll" () As Long
Declare Sub cStopBasisTimer Lib "time2win.dll" ()
```

## **Call Syntax :**

```
Call cStartBasisTimer
test& = cReadBasisTimer()
Call cReadBasisTimer
```

## **Where :**

test&                    the current value of the default timer.

## **Comments :**

The value of the timer is in milliseconds.

The accuracy of the timer is 55 milliseconds (1/18.2 second).

## **Examples :**

```
Dim i                    as Long
Dim n                    as Long
```

```
Call cStartBasisTimer
For i = 1 To 123456
    n = i * 2
Next i
MsgBox "Time (in milliseconds) to perform the test is " & cReadBasisTimer() & " milliseconds"
```

'On my system : "Time (in milliseconds) to perform the test is 769"

**See also :** [Timer](#)

# ReadTimer, StartTimer, StopTimer

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[StartTimer](#) start the specified timer.

[ReadTimer](#) read the value of the specified timer.

[StopTimer](#) stop the value of the specified timer.

## Declare Syntax :

```
Declare Sub cStartTimer Lib "time2win.dll" (ByVal nTimer As Integer)
Declare Function cReadTimer Lib "time2win.dll" (ByVal nTimer As Integer) As Long
Declare Function cStopTimer Lib "time2win.dll" (ByVal nTimer As Integer) As Long
```

## Call Syntax :

```
Call cStartTimer(nTimer)
test& = cReadTimer(nTimer)
test& = cStopTimer(nTimer)
```

## Where :

nTimer            is the timer counter between 1 to 32.  
test&            is the current value of the specified timer.

## Comments :

The value of timers is in milliseconds.  
The accuracy of timers is 55 milliseconds (1/18.2 second).

## Examples :

```
Dim i            as Long
Dim n            as Long
```

```
Call cStartTimer(7)
For i = 1 To 54321
    n = i * 2
Next i
MsgBox "Time (in milliseconds) to perform the test is " & cReadTimer(7) & " milliseconds"
```

' On my system : "Time (in milliseconds) to perform the test is 330"

**See also :** [Timer](#)

# CheckWait, SetWait, StartWait

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SetWait](#) set the time to wait in a specified timer.

[StartWait](#) start the specified timer.

[CheckWait](#) check if the specified timer has reached the time to wait.

## Declare Syntax :

```
Declare Sub cSetWait Lib "time2win.dll" (ByVal nTimer As Integer, ByVal nValue As Long)
```

```
Declare Sub cStartWait Lib "time2win.dll" (ByVal nTimer As Integer)
```

```
Declare Function cCheckWait Lib "time2win.dll" (ByVal nTimer As Integer) As Integer
```

## Call Syntax :

```
Call cSetWait(nTimer, nValue)
```

```
Call cStartWait(nTimer)
```

```
test% = cCheckWait(nTimer)
```

## Where :

nTimer	is the timer counter between 1 TO 32.
nValue	is the value to wait in milliseconds.
test%	TRUE if the time to wait is reached. FALSE if the time to wait is not reached.

## Comments :

The value of timers is in milliseconds.

The accuracy of timers is 55 millisecond (1/18.2 second).

## Examples :

```
Dim i As Long  
Dim n As Long
```

```
i = 0
```

```
Call cStartTimer(32)
```

```
Call cSetWait(7, 1000)
```

```
Call cStartWait(7)
```

```
Do Until (cCheckWait(7) = True)
```

```
    i = i + 1
```

```
    n = i * 2
```

```
Loop
```

```
MsgBox "Total iterations in 1 second (1000 milliseconds) is " & i & ", waiting time is " & cReadTimer(32) & " milliseconds"
```

```
' On my system : "Total iterations in 1 second (1000 milliseconds) is 54929, waiting time is 1043 milliseconds"
```

**See also :** [Timer](#)



' structure for disk array

```
Type tagDISKARRAY
  daSize      As Integer      'size of the type'd
  Signature   As String * 7   'signature
  nFilename   As String * 128 'name of the file
  nTypeAs Integer 'variable type
  nRows       As Long         'number of rows
  nCols As Long         'number of cols
  nSheets     As Long         'number of sheets
  rHandle     As Long         'returned handle for use with other functions
  rElementSize As Long       'returned size of a element
  rFileSize   As Long       'returned size of the file
  rPartsAs Long 'returned total part
  rRemain     As Long       'returned size of the remain part
  rSheetSize  As Long       'size of a sheet
  rOffset1    As Long       'returned offset 1
  rOffset2    As Long       'returned offset 2
  rTime As Long         'time for the last correct transaction
  nIsTyped   As Integer     'is nType a type'd variable
  Dummy      As String * 7  'reserved for future use
End Type
```

' definition for variable type in disk array

```
Public Const DA_TYPE = 0
Public Const DA_BYTE = -1
Public Const DA_INTEGER = -2
Public Const DA_LONG = -3
Public Const DA_SINGLE = -4
Public Const DA_DOUBLE = -5
Public Const DA_CURRENCY = -6
```

' definition for error type in disk array

```
Public Const DA_NO_ERROR = True
Public Const DA_EMPTY_FILENAME = 1
Public Const DA_BAD_FILENAME = 2
Public Const DA_CAN_KILL_FILE = 3
Public Const DA_CAN_NOT_OPEN_FILE = 4
Public Const DA_FILE_NOT_FOUND = 5
Public Const DA_BAD_TYPE = 6
Public Const DA_BAD_ROWS = 7
Public Const DA_BAD_COLS = 8
Public Const DA_BAD_SHEETS = 9
Public Const DA_CAN_NOT_WRITE_HEADER = 10
Public Const DA_CAN_NOT_WRITE_PART = 11
Public Const DA_CAN_NOT_WRITE_REMAIN = 12
Public Const DA_CAN_NOT_READ_HEADER = 13
Public Const DA_HEADER_SIZE = 14
Public Const DA_BAD_SIGNATURE = 15
Public Const DA_FILE_SIZE_MISMATCH = 16
Public Const DA_CAN_NOT_SEEK = 17
Public Const DA_INVALID_HANDLE = 18
Public Const DA_CAN_NOT_READ_PART = 19
Public Const DA_CAN_NOT_READ_REMAIN = 20
```

' structure for multiple disk array

```
Type tagMULTIPLEDISKARRAY
daSize          As Integer      'size of the structure
Signature       As String * 7   'signature
nFilename       As String * 128 'name of the file
nType(1 To 20) As Integer      'standard variable type (for 20 arrays)
nIsTyped(1 To 20) As Integer    'is a type'd (for 20 arrays)
nRows(1 To 20)  As Long         'number of rows (for 20 arrays)
nCols(1 To 20) As Long         'number of cols (for 20 arrays)
nSheets(1 To 20) As Long       'number of sheets (for 20 arrays)
rHandle        As Long         'returned handle for use with other functions
rFileSize      As Long         'returned size of the file
rElementSz(1 To 20) As Long    'returned size of a element (for 20 arrays)
rSheetSz(1 To 20) As Long      'size of a sheet (for 20 arrays)
rOffsetPos(1 To 20) As Long    'position of each array in the file (for 20 arrays)
rOffset1       As Long         'returned offset 1
rOffset2       As Long         'returned offset 2
rTime          As Long         'time for the last correct transaction
Dummy         As String * 28   'reserved for future use
End Type
```

' definition for variable type in multiple disk array

```
Public Const MDA_TYPE = 0
Public Const MDA_BYTE = -1
Public Const MDA_INTEGER = -2
Public Const MDA_LONG = -3
Public Const MDA_SINGLE = -4
Public Const MDA_DOUBLE = -5
Public Const MDA_CURRENCY = -6
```

' definition for error type in multiple disk array

```
Public Const MDA_NO_ERROR = -1
Public Const MDA_EMPTY_FILENAME = 1
Public Const MDA_BAD_FILENAME = 2
Public Const MDA_CAN_KILL_FILE = 3
Public Const MDA_CAN_NOT_OPEN_FILE = 4
Public Const MDA_FILE_NOT_FOUND = 5
Public Const MDA_BAD_TYPE = 6
Public Const MDA_BAD_ROWS = 7
Public Const MDA_BAD_COLS = 8
Public Const MDA_BAD_SHEETS = 9
Public Const MDA_CAN_NOT_WRITE_HEADER = 10
Public Const MDA_CAN_NOT_WRITE_PART = 11
Public Const MDA_CAN_NOT_WRITE_REMAIN = 12
Public Const MDA_CAN_NOT_READ_HEADER = 13
Public Const MDA_HEADER_SIZE = 14
Public Const MDA_BAD_SIGNATURE = 15
Public Const MDA_FILE_SIZE_MISMATCH = 16
Public Const MDA_CAN_NOT_SEEK = 17
Public Const MDA_INVALID_HANDLE = 18
Public Const MDA_CAN_NOT_READ_PART = 19
Public Const MDA_CAN_NOT_READ_REMAIN = 20
Public Const MDA_BAD_MULTIPLE_ARRAY = 21
```

# IEEEEnum : Overview

[CVB](#), [CVC](#), [CVD](#), [CVI](#), [CVL](#) and [CVS](#) returns number in a certain precision given a string containing the IEEE representation of the number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG and SINGLE.

[CVSMBF](#) returns a Single number given a string containing the Microsoft Binary Format representation of the number.

[CVB](#) [CVC](#) [CVD](#) [CVI](#) [CVL](#) [CVS](#) [CVSMBF](#)

[MKB](#), [MKC](#), [MKD](#), [MKI](#), [MKL](#), and [MKS](#) returns a string containing the IEEE representation of a number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG, SINGLE.

[MKN](#) returns a string containing the IEEE representation of a big double number. The big double is not a part of the standard variable type of VB.

[MKSMBF](#) returns a string containing the Microsoft Binary Format representation of a number.

[MKB](#) [MKC](#) [MKD](#) [MKI](#) [MKL](#) [MKN](#) [MKS](#) [MKSMBF](#)

[GetCVB](#), [GetCVC](#), [GetCVD](#), [GetCVI](#), [GetCVL](#) and [GetCVS](#) returns number in a certain precision given a string containing the IEEE representation of the number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG and SINGLE. These functions is a fully replacement of the `cVx(MID$(string, offset, length))`.

[GetCVB](#) [GetCVC](#) [GetCVD](#) [GetCVI](#) [GetCVL](#) [GetCVS](#)

[PutMKB](#), [PutMKC](#), [PutMKD](#), [PutMKI](#), [PutMKL](#), and [PutMKS](#) returns a string containing the IEEE representation of a number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG, SINGLE. These functions is a fully replacement of the `Mid$(string, offset, length) = Value`.

[PutMKB](#) [PutMKC](#) [PutMKD](#) [PutMKI](#) [PutMKL](#) [PutMKS](#)

# Binary : Overview

<a href="#">B2I</a>	convert a binary string into an integer variable.
<a href="#">B2L</a>	convert a binary string into a long variable.
<a href="#">CreateBits</a>	create a string which contains how many bits specified by a number.
<a href="#">FindBitReset</a>	find the first bit Reset starting at the position given for a a given string.
<a href="#">FindBitSet</a>	find the first bit Set starting at the position given for a a given string.
<a href="#">FromBinary</a>	convert a binary string (0, 1) to a string
<a href="#">FromBinary2</a>	convert a binary string (custom letters) to a string
<a href="#">FromHexa</a>	convert a hexa string to an ascii string.
<a href="#">FromZ9</a>	convert a Z9 string to an ascii string.
<a href="#">GetBit</a>	return if a given bit in a given string is Set or Reset.
<a href="#">GetBitB</a>	return if a given bit in a given BYTE is Set or Reset.
<a href="#">GetBitD</a>	return if a given bit in a given DOUBLE is Set or Reset.
<a href="#">GetBitI</a>	return if a given bit in a given INTEGER is Set or Reset.
<a href="#">GetBitL</a>	return if a given bit in a given LONG is Set or Reset.
<a href="#">GetBitS</a>	return if a given bit in a given SINGLE is Set or Reset.
<a href="#">GetBitB2</a>	return if a given bit (real bit position) in a given BYTE is Set or Reset.
<a href="#">GetBitI2</a>	return if a given bit (real bit position) in a given INTEGER is Set or Reset.
<a href="#">GetBitL2</a>	return if a given bit (real bit position) in a given LONG is Set or Reset.
<a href="#">GiveBitPalindrome</a>	return all chars on which bit 0 is bit 7, bit 1 is bit 6, bit 2 is bit 5, bit 3 is bit 4.
<a href="#">H2I</a>	convert a hexa string into an integer variable.
<a href="#">H2L</a>	convert a hexa string into a long variable.
<a href="#">I2B</a>	convert an integer value into a binary string.
<a href="#">I2Bext</a>	convert an integer value into a binary string (custom chars).
<a href="#">IsBitPalindrome</a>	check if a string is Bit palindrome.
<a href="#">L2B</a>	convert a long value into a binary string.
<a href="#">L2Bext</a>	convert a long value into a binary string (custom chars).
<a href="#">ReverseAllBits</a>	reverse all bits in a given string.
<a href="#">ReverseAllBitsByChar</a>	reverse all bits by each char in a given string.
<a href="#">SetAllBits</a>	set all bits of a given string to Set state or Reset state.
<a href="#">SetBit</a>	set a given bit in a given string to Set state or Reset state.
<a href="#">SetBitB</a>	set a given bit in a given BYTE to Set state or Reset state.
<a href="#">SetBitD</a>	set a given bit in a given DOUBLE to Set state or Reset state.
<a href="#">SetBitI</a>	set a given bit in a given INTEGER to Set state or Reset state.
<a href="#">SetBitL</a>	set a given bit in a given LONG to Set state or Reset state.
<a href="#">SetBitS</a>	set a given bit in a given SINGLE to Set state or Reset state.
<a href="#">SetBitB2</a>	set a given bit (real bit position) in a given BYTE to Set state or Reset state.
<a href="#">SetBitI2</a>	set a given bit (real bit position) in a given INTEGER to Set state or Reset state.
<a href="#">SetBitL2</a>	set a given bit (real bit position) in a given LONG to Set state or Reset state.
<a href="#">SetBitToFalse</a>	set a given bit in a given string to Reset state.
<a href="#">SetBitToTrue</a>	set a given bit in a given string to Set state.
<a href="#">ToBinary</a>	convert a string to a binary representation with 0, 1
<a href="#">ToBinary2</a>	convert a string to a binary representation with two custom letters for 0, 1representation
<a href="#">ToggleAllBits</a>	toggle all bits in a given string. If a bit is in Set state, it comes in Reset state. If a bit is in
<a href="#">ToggleBit</a>	toggle a given bit in a given string. If a bit is in Set state, it comes in Reset state. If a bit is
<a href="#">ToggleBit2</a>	toggle a given bit (real bit position) in a given BYTE to Set state or Reset state.
<a href="#">ToggleBitI2</a>	toggle a given bit (real bit position) in a given INTEGER to Set state or Reset state.
<a href="#">ToggleBitL2</a>	toggle a given bit (real bit position) in a given LONG to Set state or Reset state.
<a href="#">ToHexa</a>	convert a ascii string to hexa string.
<a href="#">ToZ9</a>	convert a ascii string to Z9 string.

# CVx

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CVB](#), [CVC](#), [CVD](#), [CVI](#), [CVL](#) and [CVS](#) return number in a certain precision given a string containing the IEEE representation of the number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG and SINGLE.

[CVSMBF](#) return a Single number given a string containing the Microsoft Binary Format representation of the number.

## Declare Syntax :

```
Declare Function cCVB Lib "time2win.dll" (Value As String) As Integer
Declare Function cCVC Lib "time2win.dll" (Value As String) As Currency
Declare Function cCVD Lib "time2win.dll" (Value As String) As Double
Declare Function cCVI Lib "time2win.dll" (Value As String) As Integer
Declare Function cCVL Lib "time2win.dll" (Value As String) As Long
Declare Function cCVS Lib "time2win.dll" (Value As String) As Single
Declare Function cCVSMBF Lib "time2win.dll" (Value As String) As Single
```

## Call Syntax :

```
test% = cCVB(Value$)
test@ = cCVC(Value$)
test# = cCVD(Value$)
test% = cCVI(Value$)
test& = cCVL(Value$)
test! = cCVS(Value$)
test! = cCVSMBF(Value$)
```

## Where :

test? receives the value represented by the IEEE string held in Value\$

For CVSMBF : test! receives the value represented by the MBF string held in Value\$

## Comments :

**See also :** [MKx](#)

# MKx

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**MKB**, **MKC**, **MKD**, **MKI**, **MKL**, and **MKS** return a string containing the IEEE representation of a number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG, SINGLE.

**MKN** return a string containing the IEEE representation of a big double number. The big double is not a part of the standard variable type of VB.

**MKSMBF** return a string containing the Microsoft Binary Format representation of a number.

## Declare Syntax :

```
Declare Function cMKB Lib "time2win.dll" (ByVal Value As Integer) As String
Declare Function cMKC Lib "time2win.dll" (ByVal Value As Currency) As String
Declare Function cMKD Lib "time2win.dll" (ByVal Value As Double) As String
Declare Function cMKI Lib "time2win.dll" (ByVal Value As Integer) As String
Declare Function cMKL Lib "time2win.dll" (ByVal Value As Long) As String
Declare Function cMKS Lib "time2win.dll" (ByVal Value As Single) As String
Declare Function cMKSMBF Lib "time2win.dll" (ByVal Value As Single) As String
```

```
Declare Function cMKN Lib "time2win.dll" (ByVal Value As String) As String
```

## Call Syntax :

```
Nm$ = cMKB(Value%)
Nm$ = cMKC(Value@)
Nm$ = cMKD(Value#)
Nm$ = cMKI(Value%)
Nm$ = cMKL(Value&)
Nm$ = cMKS(Value!)
Nm$ = cMKSMBF(Value!)
```

```
Nm$ = cMKN(Value$)
```

## Where :

Nm\$ receives the IEEE representation of Value?.

For MKSMBF : Nm\$ receives the MBF representation of Value!.

## Comments :

For cMKN :

Arithmetics operations on big double value must be use the function defined in [cBig.x](#).

To convert a standard value to a big double value, you must pass the string representation of the value.

The string representation of the value must be founded by using STR\$ not FORMAT\$.

In fact, the FORMAT\$ convert the decimal separator into the separator defined in the Control Panel (Number format). The STR\$ doesn't change the decimal separator.

The length of the string representation of a big double is always 10 chars.

**See also :** [CVx](#)

# FromHexa, ToHexa

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FromHexa](#) convert a hexa string to an ascii string.

[ToHexa](#) convert a ascii string to hexa string.

## **Declare Syntax :**

Declare Function cFromHexa Lib "[time2win.dll](#)" (Text As String) As String

Declare Function cToHexa Lib "[time2win.dll](#)" (Text As String) As String

## **Call Syntax :**

```
test$ = cFromHexa(Text)
```

```
test$ = cToHexa(Text)
```

## **Where :**

Text                the string to proceed

test\$               the result

## **Comments :**

The returned string from ToHexa is always a multiple of 2

If the size of the string passed to FromHexa is not a multiple of 2, only n-1 chars are used

## **Examples :**

```
test$ = cToHexa("ABCDEFGH")
```

```
' "41424344454647"
```

```
test$ = cFromHexa("47464544434241")
```

```
' "GFEDCBA"
```

**See also :** [Binary](#)

# B2I, B2L, H2I, H2L, I2B, I2Bext, L2B, L2Bext

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**B2I** convert a binary string into an integer variable.  
**B2L** convert a binary string into a long variable.  
**H2I** convert a hexa string into an integer variable.  
**H2L** convert a hexa string into a long variable.  
**I2B** convert an integer value into a binary string.  
**I2Bext** converts an integer value into a binary string (custom chars).  
**L2B** convert a long value into a binary string.  
**L2Bext** converts a long value into a binary string (custom chars).

## Declare Syntax :

```
Declare Function cB2I Lib "time2win.dll" (ByVal Txt As String) As Integer
Declare Function cB2L Lib "time2win.dll" (ByVal Txt As String) As Long
Declare Function cH2I Lib "time2win.dll" (ByVal Txt As String) As Integer
Declare Function cH2L Lib "time2win.dll" (ByVal Txt As String) As Long
Declare Function cI2B Lib "time2win.dll" (ByVal Value As Integer) As String
Declare Function cI2Bext Lib "time2win.dll" (ByVal Value As Integer, Bin As String) As String
Declare Function cL2B Lib "time2win.dll" (ByVal Value As Long) As String
Declare Function cL2Bext Lib "time2win.dll" (ByVal Value As Long, Bin As String) As String
```

## Call Syntax :

```
Test% = cB2I(txtBinary$)
Test& = cB2L(txtBinary$)
Test% = cH2I(txtHexa$)
Test& = cH2L(txtHexa$)
Test$ = cI2B(intValue)
Test$ = cI2B(intValue, Bin$)
Test$ = cL2B(IngValue)
Test$ = cL2B(intValue, Bin$)
```

## Where :

txtBinary\$	is a binary string (only combinaison of 0, 1)
txtHexa\$	is a hexa string (only combinaison of A-Z, a-z, 0-9)
Bin\$	the two custom letters for 0, 1 representation
intValue	is an integer value
IngValue	is a long value

## Comments :

For B2I, B2L, H2I, H2L :

If the function detects that that a char is not valid, the conversion is stopped and the returned value is truncated.

## Examples :

```
Debug.Print cB2I("1")           ' -> 1
Debug.Print cB2I("11")          ' -> 3
Debug.Print cB2I("11111111")    ' -> 255
Debug.Print cB2I("11111111111111") ' -> -1
Debug.Print cB2I("01010101010101") ' -> 21845
Debug.Print cB2I("10101010101010") ' -> -21846
```



```

Debug.Print cB2L("1") ' -> 1
Debug.Print cB2L("1111111111111111") ' -> 65535
Debug.Print cB2L("11111111111111111111111111111111") ' -> -1
Debug.Print cB2L("01010101010101010101010101010101") ' -> 1431655765
Debug.Print cB2L("10101010101010101010101010101010") ' -> -1431655766

Debug.Print cH2L("0") ' -> 0
Debug.Print cH2L("A1") ' -> 161
Debug.Print cH2L("A1B") ' -> 2587
Debug.Print cH2L("7FFF") ' -> 32767
Debug.Print cH2L("A1B2") ' -> -24142
Debug.Print cH2L("FFFF") ' -> -1

Debug.Print cH2L("0") ' -> 0
Debug.Print cH2L("A1") ' -> 161
Debug.Print cH2L("A1B") ' -> 2587
Debug.Print cH2L("A1B2") ' -> 41394
Debug.Print cH2L("7FFFFFFF") ' -> 2147483647
Debug.Print cH2L("B2A1A1B2") ' -> -1298030158
Debug.Print cH2L("FFFFFFFF") ' -> -1

Debug.Print cl2B(12345) ' -> 11000000111001
Debug.Print cl2B(-12345) ' -> 1100111111000111
Debug.Print cL2B(1234567890) ' -> 1001001100101100000001011010010
Debug.Print cL2B(-1234567890) ' -> 10110110011010011111110100101110

Debug.Print cl2Bext(12345, "X=") ' -> XX=====XXX==X
Debug.Print cl2Bext(-12345, "X=") ' -> XX==XXXXXXXX==XXX
Debug.Print cL2Bext(1234567890, "X=") ' ->
X==X==XX==X=XX=====X=XX=X==X=
Debug.Print cL2Bext(-1234567890, "X=") ' ->
X=XX=XX==XX=X==XXXXXXXXX=X==X=XXX=

```

**See also :** [Binary](#)

# SetAllBits, SetBit, SetBitToFalse, SetBitToTrue

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SetAllBits](#) set all bits of a given string to Set state or Reset state.

[SetBit](#) set a given bit in a given string to Set state or Reset state.

[SetBitToFalse](#) set a given bit in a given string to Reset state.

[SetBitToTrue](#) set a given bit in a given string to Set state.

## Declare Syntax :

```
Declare Sub cSetAllBits Lib "time2win.dll" (Txt As String, ByVal Value As Integer)
```

```
Declare Sub cSetBit Lib "time2win.dll" (Txt As String, ByVal Position As Integer, ByVal Value As Integer)
```

```
Declare Sub cSetBitToFalse Lib "time2win.dll" (Txt As String, ByVal Position As Integer)
```

```
Declare Sub cSetBitToTrue Lib "time2win.dll" (Txt As String, ByVal Position As Integer)
```

## Call Syntax :

```
Call cSetAllBits(Txt$, Value)
```

```
Call cSetBit(Txt$, Position, Value)
```

```
Call cSetBitToFalse(Txt$, Position)
```

```
Call cSetBitToTrue(Txt$, Position)
```

## Where :

Txt\$	the string to proceed
Position	the bit position
Value	TRUE to Set the bit FALSE to Reset the bit

## Comments :

The first bit in the string is the bit 0.

For cSetBitToFalse :

This routine is a short-cut routine from cSetBit(Txt, Position, FALSE)

For cSetBitToTrue :

This routine is a short-cut routine from cSetBit(Txt, Position, TRUE)

**See also :** [Binary](#)

# FindBitReset, FindBitSet

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FindBitReset](#) find the first bit Reset starting at the position given for a a given string.

[FindBitSet](#) find the first bit Set starting at the position given for a a given string.

## Declare Syntax :

Declare Function cFindBitReset Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Integer) As Integer

Declare Function cFindBitSet Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Integer) As Integer

## Call Syntax :

```
test = cFindBitReset(Txt$, Position)
```

```
test = cFindBitSet(Txt$, Position)
```

## Where :

Txt\$	the string to proceed
Position	the starting position
test	TRUE if no bit founded
	<> TRUE if a bit founded

## Comments :

For cFindBitReset :

This function is useful to find or scan a string for the bit Reset. The first bit in the string to start the test is -1.

For cFindBitSet :

This function is useful to find or scan a string for the bit Set. The first bit in the string to start the test is -1.

**See also :** [Binary](#)

# ToggleAllBits, ToggleBit

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ToggleAllBits](#) toggle all bits in a given string. If a bit is in Set state, it comes in Reset state. If a bit is in Reset state, it comes is Set state.

[ToggleBit](#) toggle a given bit in a given string. If a bit is in Set state, it comes in Reset state. If a bit is in Reset state, it comes is Set state.

## Declare Syntax :

```
Declare Sub cToggleAllBits Lib "time2win.dll" (Txt As String)
Declare Sub cToggleBit Lib "time2win.dll" (Txt As String, ByVal Position As Integer)
```

## Call Syntax :

```
Call cToggleAllBits(Txt$)
Call cToggleBit(Txt, Position)
```

## Where :

Txt\$	the string to proceed
Position	the bit position

## Comments :

The first bit in the string is the bit 0.

**See also** : [Binary](#)

# ReverseAllBits, ReverseAllBitsByChar

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[ReverseAllBits](#) reverse all bits in a given string.

[ReverseAllBitsByChar](#) reverse all bits by each char in a given string.

## **Declare Syntax :**

Declare Sub cReverseAllBits Lib "[time2win.dll](#)" (Txt As String)

Declare Sub cReverseAllBitsByChar Lib "[time2win.dll](#)" (Txt As String)

## **Call Syntax :**

Call cReverseAllBits(Txt\$)

Call cReverseAllBitsByChar(Txt\$)

## **Where :**

Txt\$                    the string to proceed

## **Comments :**

**See also :** [Binary](#)

# IsBitPalindrome

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[IsBitPalindrome](#) check if a string is Bit palindrome.

## **Declare Syntax :**

Declare Function clsBitPalindrome Lib "[time2win.dll](#)" (Txt As String) As Integer

## **Call Syntax :**

```
test = clsBitPalindrome(Txt$)
```

## **Where :**

Txt\$	the string to proceed
test	TRUE if the string is Bit palindrome FALSE if the string is not Bit Palindrome

## **Comments :**

**See also :** [Binary](#)

# CreateBits

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CreateBits](#) create a string which contains how many bits specified by a number.

## Declare Syntax :

Declare Function cCreateBits Lib "[time2win.dll](#)" (ByVal nBits As Integer) As String

## Call Syntax :

```
test = cCreateBits(nBits)
```

## Where :

nBits	number of bits wished
test	the result

## Comments :

## Examples :

```
nBits = 10  
test = cCreateBits(nBits)           ' test will be a size of 2 chars
```

**See also :** [Binary](#)

# GetBit

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetBit](#) return if a given bit in a given string is Set or Reset.

## **Declare Syntax :**

Declare Function cGetBit Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Integer) As Integer

## **Call Syntax :**

test = cGetBit(Txt, Position)

## **Where :**

Txt	the string to proceed
Position	the bit position
test	TRUE if the bit is Set FALSE if the bit is Reset

## **Comments :**

The first bit in the string is the bit 0.

**See also :** [Binary](#)



# GiveBitPalindrome

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GiveBitPalindrome](#) return all chars on which bit 0 is bit 7, bit 1 is bit 6, bit 2 is bit 5, bit 3 is bit 4.

## **Declare Syntax :**

Declare Function cGiveBitPalindrome Lib "[time2win.dll](#)" () As String

## **Call Syntax :**

test = cGiveBitPalindrome

## **Where :**

test                    the result

## **Comments :**

**See also :** [Binary](#)

# Get, GetBlock, GetIn, GetInPart, GetInPartR, GetInR, TokenIn

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 \(Win95/WinNT\)](#), [MSOffice 95](#)

## Purpose :

[Get](#) extracts a sub-string delimited by '|' in a given string.

[GetBlock](#) reads a block of n chars starting at a given block in a given string.

[GetIn](#) extracts a left sub-string delimited by a separator in a given string.

[GetInPart](#) extracts the first left sub-string or the rest after the first sub-string delimited by a separator in a given string.

[GetInPartR](#) extracts the first right sub-string or the rest after the first sub-string delimited by a separator in a given string.

[GetInR](#) extracts a right sub-string delimited by a separator in a given string.

[TokenIn](#) extracts a sub-string delimited by a separator's list in a given string.

## Declare Syntax :

```
Declare Function cGet Lib "time2win.dll" (Txt As String, ByVal Position As Long) As String
```

```
Declare Function cGetBlock Lib "time2win.dll" (Txt As String, ByVal Position As Long, ByVal Length As Integer) As String
```

```
Declare Function cGetIn Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Long) As String
```

```
Declare Function cGetInPart Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Long) As String
```

```
Declare Function cGetInPartR Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Long) As String
```

```
Declare Function cGetInR Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Long) As String
```

```
Declare Function cTokenIn Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Long) As String
```

## Call Syntax :

```
test$ = cGet(Txt, Position)
```

```
test$ = cGetBlock(Txt, Position, Length)
```

```
test$ = cGetIn(Txt, Separator, Position)
```

```
test$ = cGetInPart(Txt, Separator, Position)
```

```
test$ = cGetInPartR(Txt, Separator, Position)
```

```
test$ = cGetInR(Txt, Separator, Position)
```

```
test$ = cTokenIn(Txt, SeparatorList, Position)
```

## Where :

Txt	the string to proceed.
Position	the position of the sub-string or the block.
Length	the length of each block.
Separator	the delimiter for each sub-string.
SeparatorList	the separator's list for each sub-string.
test\$	the result.

## Comments :

\* If the size of the string is 0 or if the position is < 1 or greater than the maximum block is the string or if the length is 0. The returned string is an empty string.

\* The function cGet is a subset of the cGetIn function.

\* The function cGetBlock is similar to MID\$(Txt, 1 + ((n-1) \* m), m)

\* The function cTokenIn is a superset of the cGetIn function, in the fact that you can pass a separator's list.

\* For the function cGetInPart, cGetInPartR, you must set Position to TRUE for first part (left or right) and to FALSE for second part (left or right).

\* The function cGetInPartR is very usefull when you must isolate a file extension or the full directory and the filename function.

## Examples :

test\$ = cGet("A BC DEF G", 1)	-> "A"
test\$ = cGet("A BC DEF G", 3)	-> "DEF"
test\$ = cGetIn("A/BC/DEF/G", "/", 4)	-> "G"
test\$ = cGetIn("A/BC/DEF/G", "D", 2)	-> "EF/G"
test\$ = cGetInR("A/BC/DEF/G", "/", 4)	-> "A"
test\$ = cGetInR("A/BC/DEF/G", "D", 2)	-> "A/BC/"
test\$ = cGetInPart("A/BC/DEF/G", "/", True)	-> "A"
test\$ = cGetInPart("A/BC/DEF/G", "/", False)	-> "BC/DEF/G"
test\$ = cGetInPartR("c:\vberr.hnd\test.mak", ".", True)	-> "mak"
test\$ = cGetInPartR("c:\vberr.hnd\test.mak", ".", False)	-> "c:\vberr.hnd\test"
test\$ = cGetBlock("A/BC/DEF/G", 1, 2)	-> "A/"
test\$ = cGetBlock("A/BC/DEF/G", 4, 2)	-> "EF"
test\$ = cTokenIn("A/BC:DEF G", ":\ ", 4)	-> "G"
test\$ = cTokenIn("A/BC:DEF G", ":\ ", 3)	-> "DEF"

**See also :** [String](#)

# String : Overview

<a href="#">AddDigit</a>	sum all numerics chars in a given string.
<a href="#">Align</a>	align a give string (left, center, right) into an another new string.
<a href="#">AndToken</a>	check if all items of a list of token separated by ' ' is present in a specified string.
<a href="#">AndTokenIn</a>	check if all items of a list of token separated by a separator is present in a specified string.
<a href="#">ArabicToRoman</a>	convert an integer or a long integer into Roman representation.
<a href="#">BlockCharFromLeft</a>	read n chars from the left of a string.
<a href="#">BlockCharFromRight</a>	read n chars from the right of a string.
<a href="#">ChangeChars</a>	change all chars speciefien by others chars in a string.
<a href="#">ChangeCharsUntil</a>	change all chars speciefien by others chars in a string until a char is encountered.
<a href="#">CheckChars</a>	verify that all chars speciefien are present in a string.
<a href="#">CheckNumericity</a>	check if a string is a numeric string.
<a href="#">CnvASCIItoEBCDIC</a>	convert an ASCII string into EBCDIC equivalent.
<a href="#">CnvEBCDICtoASCII</a>	convert an EBCDIC string into ASCII equivalent.
<a href="#">Compact</a>	compact a string composed of numeric chars.
<a href="#">Compress</a>	remove all chr\$(0):ASCII NULL, chr\$(9):TAB, chr\$(32):SPACE from a string.
<a href="#">CompressTab</a>	pack all n space chars into a tab char.
<a href="#">CplAlpha</a>	return the complementary string from a given string composed with ascii chars.
<a href="#">CplDigit</a>	return the complementary string from a given string composed with numerics chars.
<a href="#">Count</a>	count the number of a specified char in a string.
<a href="#">CreateAndFill</a>	create a string with the specified size and fill it with some chars.
<a href="#">DeleteSubString</a>	deletes all occurances of one string, in another string.
<a href="#">ExpandTab</a>	unpack all tab chars into n space chars.
<a href="#">Extract</a>	extracts a sub-string with a key in a string.
<a href="#">ExtractIsolate</a>	extracts a left/right part of a string from a key and a field separator.
<a href="#">Fill</a>	fill a string with some chars.
<a href="#">FilterBlocks</a>	remove one or more sub-string separated by two delimiters in a given string.
<a href="#">FilterChars</a>	remove some chars speciefien in a given string.
<a href="#">FilterFirstChar</a>	remove some chars beginning at first position of a given string.
<a href="#">FilterNotChar</a>	remove all chars except speciefien chars in a given string.
<a href="#">Get</a>	extract a sub-string delimited by ' ' in a given string.
<a href="#">GetBlock</a>	read a block of n chars starting at a given block in a given string.
<a href="#">GetIn</a>	extract a left sub-string delimited by a separator in a given string.
<a href="#">GetInPart</a>	extract the first left sub-string or the rest after the first sub-string delimited by a separator
<a href="#">GetInPartR</a>	extract the first right sub-string or the rest after the first sub-string delimited by a separator
<a href="#">GetInR</a>	extract a right sub-string delimited by a separator in a given string.
<a href="#">InsertBlocks</a>	insert different block of char in a given string separated by '~'.
<a href="#">InsertBlocksExt</a>	insert different block of char in a given string separated by '~' (It can handle empty insert string).
<a href="#">InsertBlocksBy</a>	insert different block of char in a given string separated by a given separator.
<a href="#">InsertByMask</a>	replace the specified char by a string in a given string.
<a href="#">InsertChars</a>	insert a string starting at a given position in a given string.
<a href="#">InStr</a>	finds the position of the first occurrence of one string within another (like VB Instr function).
<a href="#">LFill</a>	pads a string to the left, to a know length.
<a href="#">Lrc</a>	calculate the LRC of a given string.
<a href="#">LSetIn</a>	inserts a string (to the left) into a sub-string delimited by a separator in a given string.
<a href="#">MatchTable</a>	compares a string with a set of strings delimited by a separator.
<a href="#">MixChars</a>	will mix all chars in a given string in a random position.
<a href="#">Morse</a>	convert a string to a morse string.
<a href="#">NumDigit</a>	sum and sums all numerics chars in a given string to have a maximum value of 9.
<a href="#">OneCharFromLeft</a>	read 1 char at a position starting from the left of a string.
<a href="#">OneCharFromRight</a>	read 1 char at a position starting from the right of a string.
<a href="#">OrToken</a>	check if one item of a list of token separated by ' ' is present in a specified string.
<a href="#">OrTokenIn</a>	check if one item of a list of token separated by a separator is present in a specified string.
<a href="#">PatternExtMatch</a>	search if a given pattern can be found is a given string.
<a href="#">PatternExtMatchS</a>	search if a given pattern can be found is a given string (case-sensitive or not).
<a href="#">PatternMatch</a>	search if a given pattern can be found is a given string.
<a href="#">PatternMatchS</a>	search if a given pattern can be found is a given string (case-sensitive or not).

<a href="#"><u>ProperName</u></a>	convert the first letter of each word separated by a space in a string to upper case.
<a href="#"><u>ProperName2</u></a>	convert the first letter of some words separated by a space or punctuation in upper letter case.
<a href="#"><u>RemoveBlockChar</u></a>	remove a block of chars at the specified position in a string.
<a href="#"><u>RemoveOneChar</u></a>	remove one char at the specified position in a string.
<a href="#"><u>ResizeString</u></a>	resize the size of a string to a new length.
<a href="#"><u>ResizeStringAndFill</u></a>	resize the size of a string to a new length and fill it with chars if the new length is greater than the current length.
<a href="#"><u>Reverse</u></a>	reverse all chars in a given string.
<a href="#"><u>RFill</u></a>	pads a string to the right, to a know length.
<a href="#"><u>RomanToArabic</u></a>	convert a Roman string into an integer or a long integer.
<a href="#"><u>RSetIn</u></a>	inserts a string (to the right) into a sub-string delimited by a separator in a given string.
<a href="#"><u>ScrollL</u></a>	scroll one char to the left of a specified string.
<a href="#"><u>ScrollR</u></a>	scroll one char to the right of a specified string.
<a href="#"><u>StringReplace</u></a>	searches for known strings, and replaces them with another string.
<a href="#"><u>StringSAR</u></a>	search and replace a string by an another in the specified string.
<a href="#"><u>TokenIn</u></a>	extract a sub-string delimited by a separator's list in a given string.
<a href="#"><u>Uncompact</u></a>	uncompact a string composed of numeric chars.
<a href="#"><u>WrapLine</u></a>	wraps a line in multiple lines with a maximum length by line.

# BlockCharFromLeft, BlockCharFromRight, OneCharFromLeft, OneCharFromRight

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 \(Win95/WinNT\)](#), [MSOffice 95](#)

## Purpose :

[BlockCharFromLeft](#) read n chars from the left of a string.

[BlockCharFromRight](#) read n chars from the right of a string.

[OneCharFromLeft](#) read 1 char at a position starting from the left of a string.

[OneCharFromRight](#) read 1 char at a position starting from the right of a string.

## Declare Syntax :

Declare Function cBlockCharFromLeft Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Integer) As String

Declare Function cBlockCharFromRight Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Integer) As String

Declare Function cOneCharFromLeft Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Integer) As String

Declare Function cOneCharFromRight Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Integer) As String

## Call Syntax :

```
test = cBlockCharFromLeft(Txt, Position)
```

```
test = cBlockCharFromRight(Txt, Position)
```

```
test = cOneCharFromLeft(txt, position)
```

```
test = cOneCharFromRight(Txt, Position)
```

## Where :

Txt                   the string to extract some chars

Position             the number of chars to read

Test                  the result

## Comments :

For cBlockCharFromLeft :

This fonction is the same that Left\$(Txt, Position) but doesn't generate an Error if a problem occurs.

For cBlockCharFromRight :

This fonction is the same that Right\$(Txt, Position) but doesn't generate an Error if a problem occurs.

From cOneCharFromLeft :

This function is the same that MID\$(Txt, Position, 1)

From cOneCharFromRight :

This function is the same that MID\$(Txt, Len(Txt) - Position + 1, 1)

## Examples :

For cBlockCharFromLeft :

```
Txt = "ABCDEF"
```

```
Position = 3
```

```
Test = cBlockCharFromLeft(Txt, Position)
```

```
' Test = "ABC"
```

For cBlockCharFromRight :

```
Txt = "ABCDEF"  
Position = 3  
Test = cBlockCharFromRight(Txt, Position)          ' Test = "DEF"
```

For cOneCharFromLeft :

```
Txt = "ABCDEF"  
Position = 3  
Test = cOneCharFromLeft(Txt, Position)            ' Test = "C"
```

For cOneCharFromRight :

```
Txt = "ABCDEF"  
Position = 3  
Test = cOneCharFromRight(Txt, Position)          ' Test = "D"
```

**See also :** [String](#)

# InsertBlocks, InsertBlocksExt, InsertBlocksBy, InsertByMask, InsertChars

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**InsertBlocks** insert different block of char in a given string separated by '~'.

**InsertBlocksExt** insert different block of char in a given string separated by '~' (It can handle empty insert string).

**InsertBlocksBy** insert different block of char in a given string separated by a given separator.

**InsertByMask** replace the specified char by a string in a given string.

**InsertChars** insert a string starting at a given position in a given string.

## Declare Syntax :

```
Declare Function cInsertBlocks Lib "time2win.dll" (Txt As String, Insert As String) As String
Declare Function cInsertBlocksExt Lib "time2win.dll" (Txt As String, Insert As String) As String
Declare Function cInsertBlocksBy Lib "time2win.dll" (Txt As String, Insert As String, Delimiter As String) As String
Declare Function cInsertByMask Lib "time2win.dll" (Txt As String, Mask As String, Insert As String) As String
Declare Function cInsertChars Lib "time2win.dll" (Txt As String, ByVal Position As Integer, Insert As String) As String
```

## Call Syntax :

```
test$ = cInsertBlocks(Txt, Insert)
test$ = cInsertBlocksExt(Txt, Insert)
test$ = cInsertBlocksBy(Txt, Insert, Delimiter)
test$ = cInsertByMask(Txt, Mask, Insert)
test$ = cInsertChars(Txt, Position, Insert)
```

## Where :

Txt	the string to proceed
Insert	the string to insert
Delimiter	the delimiter to use for the insert string
Mask	the mask to use for the insert string
Position	the position to use for the insert string

## Comments :

- \* If the size of the string is 0 The returned string is an empty string.
- \* The function cInsertBlocks is a subset of the cInsertBlocksBy function.
- \* The function cInsertBlocksExt is a subset of the cInsertBlocksBy function and can handle an empty insert string..
- \* The number of blocks for cInsertBlocks, cInsertBlocksBy functions in the string to proceed must be greater than one from the number of block in the insert string.
- \* The function cInsertChars is similar to LEFT\$(Txt, n) + Insert + RIGHT\$(Txt, LEN(Txt) - n)

## Examples :

```
test$ = cInsertBlocks("A~BC~DEF", "x~yz")           ' "AxBcYzDEF"
test$ = cInsertBlocksExt("A~BC~DEF", "")           ' "ABCDEF"
test$ = cInsertBlocksBy("U/VW/XYZ", "a/bc", "/")   ' "UaVWbcXYZ"
test$ = cInsertByMask("Nr ## Price $###.##", "#", "0705995") ' "Nr 07 Price $059.95"
test$ = cInsertChars("ABCDEFGH", 3, "wxyz")        ' "ABCwxyzDEFG"
test$ = cInsertChars("ABCDEFGH", 90, "wxyz")       ' "ABCDEFGHwxyz"
test$ = cInsertChars("ABCDEFGH", 0, "wxyz")        ' "wxyzABCDEFGH"
```



**See also :** [String](#)

# AndToken, AndTokenIn, OrToken, OrTokenIn

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**AndToken** check if all items of a list of token separated by '|' is present in a specified string.

**AndTokenIn** check if all items of a list of token separated by a separator is present in a specified string.

**OrToken** check if one item of a list of token separated by '|' is present in a specified string.

**OrTokenIn** check if one item of a list of token separated by a separator is present in a specified string.

## Declare Syntax :

```
Declare Function cAndToken Lib "time2win.dll" (ByVal Txt As String, ByVal Token As String) As Integer
Declare Function cAndTokenIn Lib "time2win.dll" (ByVal Txt As String, ByVal Token As String, ByVal Separator As String) As Integer
Declare Function cOrToken Lib "time2win.dll" (ByVal Txt As String, ByVal Token As String) As Integer
Declare Function cOrTokenIn Lib "time2win.dll" (ByVal Txt As String, ByVal Token As String, ByVal Separator As String) As Integer
```

## Call Syntax :

```
Test% = cAndToken(Txt$, Token$)
Test% = cAndTokenIn(Txt$, Token$, Separator$)
```

```
Test% = cOrToken(Txt$, Token$)
Test% = cOrTokenIn(Txt$, Token$, Separator$)
```

## Where :

Txt\$	is the specified string.
Token\$	is the list of token.
Separator\$	is the specified separator (default is ' ').
Test%	TRUE if one of the list of token is present, FALSE if not

## Comments :

AndToken, AndTokenIn, OrToken, OrTokenIn works only with string without embedded chr\$(0).  
AndToken, AndTokenIn, OrToken, OrTokenIn are case-sensitive. Use UCase\$ or LCase\$ to perform no case-sensitivity.

## Examples :

```
Dim Txt           As String
Dim Token         As String
Dim Separator     As String
Dim Test         As Integer
```

```
Txt = "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"
```

```
Token = "THE|DOG|QUICK"
Test = cOrToken(Txt, Token)           ' True
```

```
Token = "the|dog|quick"
Test = cOrToken(Txt, Token)           ' False
```

```
Token = "the\dog\quick"
Separator = "\"
Test = cOrTokenIn(LCase$(Txt), LCase$(Token), Separator) ' True
```

Token = "THE|DOG|QUICK"  
Test = cAndToken(Txt, Token) ' True

Token = "the|dog|quick"  
Test = cAndToken(Txt, Token) ' False

Token = "the\dog\quick"  
Separator = "\"  
Test = cAndTokenIn(lcase\$(Txt), lcase\$(Token), Separator) ' True

**See also :** [String](#)

# FilterBlocks, FilterChars, FilterFirstChars, FilterNotChars

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**FilterBlocks** remove one or more sub-string separated by two delimiters in a given string.

**FilterChars** remove some chars specifen in a given string.

**FilterFirstChar** remove some chars beginning at first position of a given string.

**FilterNotChar** remove all chars except speficien chars in a given string.

## Declare Syntax :

Declare Function cFilterBlocks Lib "time2win.dll" (Txt As String, Delimitior As String) As String

Declare Function cFilterChars Lib "time2win.dll" (Txt As String, charSet As String) As String

Declare Function cFilterFirstChars Lib "time2win.dll" (Txt As String, charSet As String) As String

Declare Function cFilterNotChars Lib "time2win.dll" (Txt As String, charSet As String) As String

## Call Syntax :

test = cFilterBlocks(Txt, Delimitior)

test = cFilterChars(Txt, charSet)

test = cFilterFirstChars(Txt, charSet)

test = cFilterNotChars(Txt, charSet)

## Where :

Txt                    the string to proceed

Delimitior           two chars for filter the string

charSet              the chars for filter the string

test                  the result

## Comments :

## Examples :

```
Txt = "A/BC/DEF/GHIJ"
Delimiter = "/"
test = cFilterBlocks(Txt, Delimitior)
' test = "ADEF"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "B/"
test = cFilterChars(Txt, charSet)
' test = "ACDEFGHIJ"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "A/"
test = cFilterFirstChars(Txt, charSet)
' test = "BC/DEF/GHIJ"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "B/"
test = cFilterNotChars(Txt, charSet)
' test = "A//"
```

```
Txt = "A/BC/DEF/GHIJ"
Delimiter = "B|"
test = cFilterBlocks(Txt, Delimitior)
' test = "A/J"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "AF/"
test = cFilterChars(Txt, charSet)
' test = "BCDEFGHIJ"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "A/BC/"
test = cFilterFirstChars(Txt, charSet)
' test = "DEF/GHIJ"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "AF/"
test = cFilterNotChars(Txt, charSet)
' test = "A//F"
```

**See also :** [String](#)

# AddDigit, CplDigit, NumDigit, CplAlpha

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**AddDigit** sum all numerics chars in a given string.

**CplDigit** return the complementary string from a given string composed with numerics chars.

**NumDigit** sum and sums all numerics chars in a given string to have a maximum value of 9.

**CplAlpha** return the complementary string from a given string composed with ascii chars.

## Declare Syntax :

```
Declare Function cAddDigit Lib "time2win.dll" (Txt as string) As Integer
Declare Function cCplDigit Lib "time2win.dll" (Txt as string) As String
Declare Function cNumDigit Lib "time2win.dll" (Txt as string) As Integer
Declare Function cCplAlpha Lib "time2win.dll" (Txt As String) As String
```

## Call Syntax :

```
test% = cAddDigit(Txt)
test$ = cCplDigit(Txt)
test% = cNumDigit(Txt)
test$ = cCplAlpha(Txt)
```

## Where :

Txt\$	the string to proceed
test%	the result
test\$	the result for CplAlpha

## Comments :

For AddDigit, CplDigit, NumDigit if one or more chars are different from digit, the value for each one is 0

## Examples :

```
test% = cAddDigit("1234567890987654321712345678909876543217") ' 194
test% = cNumDigit("1234567890987654321712345678909876543217")' 5
```

```
test$ = cCplDigit("1234567890987654321712345678909876543217") '
"8765432109012345678287654321090123456782"
```

```
test% = cAddDigit("8765432109012345678287654321090123456782") ' 166
test% = cNumDigit("8765432109012345678287654321090123456782")' 4
```

```
test$ = cCplAlpha("ÀÁÂÃÄÅÆ") ' ">=<;:9"
```

**See also :** [String](#)

# CnvASCIIttoEBCDIC, CnvEBCDICtoASCII

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CnvASCIIttoEBCDIC](#) convert an ASCII string into EBCDIC equivalent.

[CnvEBCDICtoASCII](#) convert an EBCDIC string into ASCII equivalent.

## Declare Syntax :

Declare Sub cCnvASCIIttoEBCDIC Lib "[time2win.dll](#)" (Txt As String)

Declare Sub cCnvEBCDICtoASCII Lib "[time2win.dll](#)" (Txt As String)

## Call Syntax :

Call cCnvASCIIttoEBCDIC(Txt\$)

Call cCnvEBCDICtoASCII(Txt\$)

## Where :

Txt\$                                    the string to convert

## Comments :

## Examples :

```
Dim Tmp                    As String
```

```
Tmp = "A/BC/DEF/GHIJ"
```

```
Call cCnvASCIIttoEBCDIC(Tmp)
```

```
Debug.Print Tmp                                    ' ÁaÂÃäÅÄÆaÇÈÉÑ
```

```
Call cCnvEBCDICtoASCII(Tmp)
```

```
Debug.Print Tmp                                    ' A/BC/DEF/GHIJ
```

**See also :** [String](#)

# ArabicToRoman, RomanToArabic

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[ArabicToRoman](#) convert an integer or a long integer into Roman representation.

[RomanToArabic](#) convert a Roman string into an integer or a long integer.

## Declare Syntax :

Declare Function cArabicToRoman Lib "time2win.dll" (Var As Variant) As String

Declare Function cRomanToArabic Lib "time2win.dll" (Txt As String) As Variant

## Call Syntax :

testAR = cArabicToRoman(var)

testRA = cRomanToArabic(txt)

## Where :

var is the integer or long integer value  
testAR return the Roman representation of var

txt is a Roman string.  
testRA return the Arabic representation of txt.

## Comments :

For cArabicToRoman :

The string returned by this function is always in lowercase.

For cRomanToArabic :

The value returned by this function is an integer or a long integer.

## Examples :

```
testAR = cArabicToRoman(1994)      ' testAR -> MCMXCIV
testAR = cArabicToRoman(1995)      ' testAR -> MCMXCV
testAR = cArabicToRoman(1993)      ' testAR -> MCMXCIII
```

```
testRA = cRomanToArabic("MCMXCIV") ' testRA -> 1994
testRA = cRomanToArabic("MCMXCV")  ' testRA -> 1995
testRA = cRomanToArabic("MCMXCIII") ' testRA -> 1993
```

**See also :** [String](#)

## Days and Months in different language : Overview

<u>GetAscTime</u>	retrieve the current date and time in a 26 chars string from a language number.
<u>GetTinyDay</u>	return the specified day into one letter.
<u>GetSmallDay</u>	return the specified day into two letters.
<u>GetShortDay</u>	return the specified day into three letters.
<u>GetLongDay</u>	return the specified day into full day name.
<u>GetTinyMonth</u>	return the specified month into one letter.
<u>GetShortMonth</u>	return the specified month into three letters.
<u>GetLongMonth</u>	return the specified month into full month name.



# Get.X.Day, Get.X.Month

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[GetTinyDay](#) return the specified day into one letter.  
[GetSmallDay](#) return the specified day into two letters.  
[GetShortDay](#) return the specified day into three letters.  
[GetLongDay](#) return the specified day into full day name.  
[GetTinyMonth](#) return the specified month into one letter.  
[GetShortMonth](#) return the specified month into three letters.  
[GetLongMonth](#) return the specified month into full month name.

## Declare Syntax :

```
Declare Function cGetTinyDay Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetSmallDay Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetShortDay Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetLongDay Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetTinyMonth Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As String
Declare Function cGetShortMonth Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As String
Declare Function cGetLongMonth Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As String
```

## Call Syntax :

```
test$ = GetTinyDay(nLanguage, nDay)
test$ = GetSmallDay(nLanguage, nDay)
test$ = GetShortDay(nLanguage, nDay)
test$ = GetLongDay(nLanguage, nDay)
test$ = GetTinyMonth(nLanguage, nMonth)
test$ = GetShortMonth(nLanguage, nMonth)
test$ = GetLongMonth(nLanguage, nMonth)
```

## Where :

nLanguage	is the language number
nDay	is the day number
nMonth	is the month number

## Comments :

nLanguage must be a correct [language number](#).  
If the language number is not correct, the french language is always returned.

nDay is the day of the week between 1 and 7. You can use the VB WeekDay() fonction to retrieve it from a date.

nMonth is a month between 1 and 12. You can use the VB Month() fonction to retrieve it from a date.

## Examples :

```
test$ = cGetShortDay(LNG_FRENCH, 1)      ' "Dim"
test$ = cGetLongDay(LNG_FRENCH, 1)      ' "Dimanche"
test$ = cGetShortDay(LNG_FRENCH, 7)     ' "Sam"
test$ = cGetLongDay(LNG_FRENCH, 7)     ' "Samedi"

test$ = cGetShortDay(LNG_DUTCH, 1)      ' "Zon"
test$ = cGetLongDay(LNG_DUTCH, 1)      ' "Zondag"
test$ = cGetShortDay(LNG_DUTCH, 7)     ' "Zat"
test$ = cGetLongDay(LNG_DUTCH, 7)     ' "Zaterdag"
```

test\$ = cGetShortMonth(LNG_FRENCH, 3)	' "Mar"
test\$ = cGetLongMonth(LNG_FRENCH, 3)	' "Mars"
test\$ = cGetShortMonth(LNG_FRENCH, 12)	' "Déc"
test\$ = cGetLongMonth(LNG_FRENCH, 12)	' "Decembre"
test\$ = cGetShortMonth(LNG_DUTCH, 3)	' "Maa"
test\$ = cGetLongMonth(LNG_DUTCH, 3)	' "Maart"
test\$ = cGetShortMonth(LNG_DUTCH, 12)	' "Dec"
test\$ = cGetLongMonth(LNG_DUTCH, 12)	' "December"

**See also :** [Days and months in different language](#)

Public Const LNG\_FRENCH = 1  
Public Const LNG\_DUTCH = 2  
Public Const LNG\_GERMAN = 3  
Public Const LNG\_ENGLISH = 4  
Public Const LNG\_ITALIAN = 5  
Public Const LNG\_SPANISH = 6  
Public Const LNG\_CATALAN = 7  
Public Const LNG\_POLISH = 8

# GetAscTime

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetAscTime](#) retrieve the current date and time in a 26 chars string from a language number.

## **Declare Syntax :**

Declare Function cGetAscTime Lib "[time2win.dll](#)" (ByVal nLanguage As Integer) As String

## **Call Syntax :**

```
test$ = cGetAscTime(nLanguage)
```

## **Where :**

nLanguage is the language number

## **Comments :**

nLanguage must be a correct [language number](#).

If the language number is not correct, the french language is always returned.

A 24-hour clock is used.

All fields have a constant width.

## **Examples :**

```
test$ = cGetAscTime(LNG_FRENCH) -> "Mer Déc 14 22:31:51 1994"
```

```
test$ = cGetAscTime(LNG_DUTCH) -> "Woe Dec 14 22:32:11 1994"
```

```
test$ = cGetAscTime(LNG_ENGLISH) -> "Wed Dec 14 22:32:29 1994"
```

**See also :** [Days and months in different language](#)

# StringCompress, StringExpand

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[StringCompress](#) compress a string into a compressed format.

[StringExpand](#) expand a compressed string into a normal format.

## Declare Syntax :

```
Declare Function cStringCompress Lib "time2win.dll" (Txt As String) As String
Declare Function cStringExpand Lib "time2win.dll" (Txt As String) As String
```

## Call Syntax :

```
Test$ = cStringCompress(Txt$)
```

```
Test$ = cStringExpand(Txt$)
```

## Where :

Txt\$ is the original string.

Test\$ is the compressed string.

## Comments :

The compression gives the better result on TEXT string.

## Examples :

```
Dim Str1 As String
Dim Str2 As String
Dim Str3 As String
```

```
Str1 = "T2WIN-32 is a powerfull DLL for VB 4.0 (32-Bit) under Win95/WinNT"
```

```
Str2 = cStringCompress(Str1)
```

```
Str3 = cStringExpand(Str2)
```

```
If (Str1 = Str3) Then Debug.Print "Success !" Else Debug.Print "Error !"
```

**See also :** [Compression](#)

# GZIPStringCompress, GZIPStringExpand

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[GZIPStringCompress](#) compress a string into a compressed format using GZIP compression method.

[GZIPStringExpand](#) expand a compressed string into a normal format using GZIP compression method.

[GZIPStringCompress2](#) compress a string into a compressed format using GZIP compression method.

[GZIPStringExpand2](#) expand a compressed string into a normal format using GZIP compression method.

## Declare Syntax :

```
Declare Function cGZIPStringCompress Lib "time2win.dll" (Txt As String) As String
```

```
Declare Function cGZIPStringExpand Lib "time2win.dll" (Txt As String) As String
```

```
Declare Function cGZIPStringCompress2 Lib "time2win.dll" (Txt As String) As String
```

```
Declare Function cGZIPStringExpand2 Lib "time2win.dll" (Txt As String) As String
```

## Call Syntax :

```
Test$ = cGZIPStringCompress(Txt$)
```

```
Test$ = cGZIPStringExpand(Txt$)
```

```
Test$ = cGZIPStringCompress2(Txt$)
```

```
Test$ = cGZIPStringExpand2(Txt$)
```

## Where :

Txt\$ is the original string.

Test\$ is the compressed string.

## Comments :

The compression gives the better result on TEXT string.

The difference between GZIPStringCompress and GZIPStringCompress2 is the fact that GZIPStringCompress2 contains in the first 4 chars, the length of the original string.

The difference between GZIPStringExpand and GZIPStringExpand2 is the fact that GZIPStringCompress2 contains in the first 4 chars, the length of the original string.

## Examples :

```
Dim Str1 As String
```

```
Dim Str2 As String
```

```
Dim Str3 As String
```

```
Str1 = "T2WIN-32 is a powerfull DLL for VB 4.0 (32-Bit) under Win95/WinNT"
```

```
Str2 = cGZIPStringCompress(Str1)
```

```
Str3 = cGZIPStringExpand(Str2)
```

```
If (Str1 = Str3) Then Debug.Print "Success !" Else Debug.Print "Error !"
```

**See also :** [Compression](#)

# FileCompress, FileExpand

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FileCompress](#) compress a file into a compressed format.

[FileExpand](#) expand a compressed file into a normal format.

## Declare Syntax :

Declare Function cFileCompress Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String) As Long  
Declare Function cFileExpand Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String) As Long

## Call Syntax :

Test& = cFileCompress(FileIn\$, FileOut\$)

Test& = cFileExpand(FileIn\$, FileOut\$)

## Where :

FileIn\$	is the original/compressed file.
FileOut\$	is the compressed/original file.
Test&	<0, an error has occurred. >=0, the length of the created file.

## Comments :

The compression gives the better result on TEXT file.

The following constants are used to explain the error code :

```
Public Const CMPEXP_FILEIN_CANT_BE_NULL = -1
' occurs when the FileIn is an empty string
Public Const CMPEXP_FILEOUT_CANT_BE_NULL = -2
' occurs when the FileOut is an empty string
Public Const CMPEXP_FILEIN_AND_FILEOUT_CANT_BE_THE_SAME = -3
' occurs when the FileIn and FileOut are the same
Public Const CMPEXP_FILEIN_CANT_BE_OPENED = -4
' occurs when the FileIn can't be opened (not valid, not exist or disk error)
Public Const CMPEXP_FILEOUT_CANT_BE_CREATED = -5
' occurs when the FileOut can't be created (not valid or disk error)
Public Const CMPEXP_COMPRESS_OR_EXPAND_ERROR = -6
' occurs when compression or expansion can't be performed (disk error)
Public Const CMPEXP_CANT_GET_FILEOUT_SIZE = -7
' occurs when the length of FileOut can be read (disk error)
```

## Examples :

```
Dim FileIn As String
Dim FileOut As String
Dim FileOut2 As String
Dim LengthIn As Long
Dim LengthOut As Long
```

```
FileIn = "c:\win95\system\msjt3032.dll"
FileOut = "c:\tmp\test.cmp"
FileOut = "c:\tmp\test.uncmp"
```

```
LengthOut = cFileCompress(FileIn, FileOut)
LengthIn = cFileExpand(FileOut, FileOut2)
```

**See also :** [Compression](#)



# Is.X

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

These routines checks if the specified string is :

<a href="#">IsAlnum</a>	Alphanumeric ('A'-'Z', 'a'-'z', or '0'-'9')
<a href="#">IsAlpha</a>	Letter ('A'-'Z' or 'a'-'z')
<a href="#">IsAscii</a>	ASCII character (0x00 - 0x7F)
<a href="#">IsCsym</a>	Letter, underscore, or digit
<a href="#">IsCsymf</a>	Letter or underscore
<a href="#">IsDigit</a>	Digit ('0'-'9')
<a href="#">IsISBN</a>	International Standard Book Numbers (ISBNs)
<a href="#">IsLower</a>	Lowercase letter ('a'-'z')
<a href="#">IsPalindrome</a>	the string and the reverse string are the same
<a href="#">IsPunct</a>	Punctuation character
<a href="#">IsSpace</a>	White-space character (0x09 - 0x0D or 0x20)
<a href="#">IsUpper</a>	Uppercase letter ('A'-'Z')
<a href="#">IsXdigit</a>	Hexadecimal digit ('A'-'F', 'a'-'f', or '0'-'9')
<a href="#">IsBalance</a>	test if the specified balance is a valid balance
<a href="#">IsDate</a>	test if the specified date is a valid date
<a href="#">IsHour</a>	test if the specified hour is a valid hour
<a href="#">IsLeapYear</a>	test if the specified year is a leap year

## Declare Syntax :

```
Declare Function clsAlnum Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsAlpha Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsAscii Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsCsym Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsCsymf Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsDigit Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsISBN Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsLower Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsPalindrome Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsPunct Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsSpace Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsUpper Lib "time2win.dll" (Txt As String) As Integer
Declare Function clsXDigit Lib "time2win.dll" (Txt As String) As Integer
```

```
Declare Function clsBalance Lib "time2win.dll" (ByVal nHour As Long, ByVal nMinute As Integer, ByVal nSecond As Integer) As Integer
Declare Function clsDate Lib "time2win.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Integer
Declare Function clsHour Lib "time2win.dll" (ByVal nHour As Integer, ByVal nMinute As Integer, ByVal nSecond As Integer) As Integer
Declare Function clsLeapYear Lib "time2win.dll" (ByVal nYear As Integer) As Integer
```

## Call Syntax :

```
test = clsAlnum(Txt)
test = clsAlpha(Txt)
test = clsAscii(Txt)
test = clsCsym(Txt)
test = clsCsymf(Txt)
test = clsDigit(Txt)
test = clsLower(Txt)
test = clsPalindrome(Txt)
```

```
test = clsPunct(Txt)
test = clsSpace(Txt)
test = clsUpper(Txt)
test = clsXdigit(Txt)
```

```
test = clsBalance(nHour, nMinute, nSecond)
test = clsDate(nYear, nMonth, nDay)
test = clsHour(nHour, nMinute, nSecond)
test = clsLeapYear(nYear)
```

**Where :**

Txt	the string to proceed
nHour	the hour to test (can be negative and/or greater than 1439 for clsBalance)
nMinute	the minute to test
nSecond	the second to test
nYear	the year to test
nMonth	the month to test
nDay	the day to test
test	TRUE if test is OK FALSE if the test fails

**Comments :**

**Examples :**

Txt = "ABCDEFGH"

test = clsAlnum(Txt)	TRUE
test = clsAlpha(Txt)	TRUE
test = clsAscii(Txt)	TRUE
test = clsCsym(Txt)	TRUE
test = clsCsymf(Txt)	TRUE
test = clsDigit(Txt)	FALSE
test = clsLower(Txt)	FALSE
test = clsPalindrome(Txt)	FALSE
test = clsPunct(Txt)	FALSE
test = clsSpace(Txt)	FALSE
test = clsUpper(Txt)	TRUE
test = clsXdigit(Txt)	FALSE

test = clsBalance(-1200, 58, 34)	TRUE
test = clsDate(1995, 2, 29)	FALSE
test = clsHour(23, 60, 10)	FALSE
test = clsLeapYear(1996)	TRUE

**See also : [ls](#)**

# HMAPutType, HMArPutType, HMAsPutType

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[HMAPutType](#) save a type'd variable from a huge array.

[HMArPutType](#) have the same fonctionnality but with a huge array with only one sheet and only one row.

[HMAsPutType](#) have the same fonctionnality but with a huge array with only one sheet.

## Declare Syntax :

Declare Sub cHMAPutType Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)

Declare Sub cHMArPutType Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Col As Long, nType As Any)

Declare Sub cHMAsPutType Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, nType As Any)

## Call Syntax :

Call cHMAPutType(HMA, Row&, Col&, Sheet&, nType)

## Where :

HMA	is a type'd variable (tagHMA).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
nType	is the type'd variable to save depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than HMA.nRows, the Row HMA.nRows is used.

If the Col is greater than HMA.nCols, the Col HMA.nCols is used.

If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

## Examples :

Dim ErrCode	As Integer	
Dim HMA	As tagHMA	
Dim TE	As tagTASKENTRY	
HMA.nType = Len(TE)		' positive value for a type'd variable
HMA.nIsTyped = True		' init the array with chr\$(0) because type'd variable
HMA.nRows = 500		' 500 rows
HMA.nCols = 500	' 500 cols	
HMA.nSheets = 2		' 2 sheets
ErrCode = cHMACreate(HMA)		' create a new huge array
ErrCode = cTasks(TE, True)		
Call cHMAPutType(HMA, 1, 1, 1, TE)		' save the type'd variable in Row 1, Col 1, Sheet 1
ErrCode = cTasks(TE, False)		
Call cHMAPutType(HMA, 1, HMA.nCols, 1, TE)		' save the type'd variable in Row 1, Col 500, Sheet 1

```
ErrCode = cTasks(TE, False)
Call cHMAPutType(HMA, HMA.nRows, 1, 1, TE)
Sheet 1
ErrCode = cTasks(TE, False)
Call cHMAPutType(HMA, HMA.nRows, HMA.nCols, 1, TE)
500, Sheet 1
```

' save the type'd variable in Row 500, Col 1,

' save the type'd variable in Row 500, Col

**See also :** [Huge memory array](#)

# Is : Overview

These routines checks if the specified string is :

<u>IsAlnum</u>	Alphanumeric ('A'-'Z', 'a'-'z', or '0'-'9')
<u>IsAlpha</u>	Letter ('A'-'Z' or 'a'-'z')
<u>IsAscii</u>	ASCII character (0x00 - 0x7F)
<u>IsCsym</u>	Letter, underscore, or digit
<u>IsCsymf</u>	Letter or underscore
<u>IsDigit</u>	Digit ('0'-'9')
<u>IsISBN</u>	International Standard Book Numbers (ISBNs)
<u>IsLower</u>	Lowercase letter ('a'-'z')
<u>IsPalindrome</u>	the string and the reverse string are the same
<u>IsPunct</u>	Punctuation character
<u>IsSpace</u>	White-space character (0x09 - 0x0D or 0x20)
<u>IsUpper</u>	Uppercase letter ('A'-'Z')
<u>IsXdigit</u>	Hexadecimal digit ('A'-'F', 'a'-'f', or '0'-'9')
<u>IsBalance</u>	test if the specified balance is a valid balance
<u>IsDate</u>	test if the specified date is a valid date
<u>IsHour</u>	test if the specified hour is a valid hour
<u>IsLeapYear</u>	test if the specified year is a leap year

These routines checks if the specified value is :

<u>IsPrime</u>	prime
----------------	-------

The routines checks if a specified file has or not the specified attribute.

- IsFileArchive check if the specified file is an ARCHIVE file.
- IsFileEmpty check if the specified file contains or not data (size > 0).
- IsFileHidden check if the specified file is a HIDDEN file.
- IsFilenameValid check if the specified file follows the DOS or WIN95 or WINNT syntax for a file.
- IsFileNormal check if the specified file is a NORMAL file.
- IsFileReadOnly check if the specified file is a READ-ONLY file.
- IsFileSubDir check if the specified file is a SUB-DIRECTORY file.
- IsFileSystem check if the specified file is a SYSTEM file.
- IsFileVold check if the specified file is a VOLUME-ID file.
- IsFileFlag check if the specified file have the specified attributes.

# IsFile.X

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**IsFileArchive** check if the specified file is an ARCHIVE file.  
**IsFileEmpty** check if the specified file contains or not data (size > 0).  
**IsFileHidden** check if the specified file is a HIDDEN file.  
**IsFilenameValid** check if the specified file follows the DOS or WIN95 or WINNT syntax for a file.  
**IsFileNormal** check if the specified file is a NORMAL file.  
**IsFileReadOnly** check if the specified file is a READ-ONLY file.  
**IsFileSubDir** check if the specified file is a SUB-DIRECTORY file.  
**IsFileSystem** check if the specified file is a SYSTEM file.  
**IsFileVold** check if the specified file is a VOLUME-ID file.  
**IsFileFlag** check if the specified file have the specified attributes.

## Declare Syntax :

```
Declare Function clsFileArchive Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFileEmpty Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFileHidden Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFilenameValid Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFileNormal Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFileReadOnly Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFileSubDir Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFileSystem Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFileVold Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function clsFileFlag Lib "time2win.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer
```

## Call Syntax :

```
test% = clsFileArchive(nFilename)
test% = clsFileEmpty(nFilename)
test% = clsFileHidden(nFilename)
test% = clsFilenameValid(nFilename)
test% = clsFileNormal(nFilename)
test% = clsFileReadOnly(nFilename)
test% = clsFileSubDir(nFilename)
test% = clsFileSystem(nFilename)
test% = clsFileVold(nFilename)
test% = clsFileFlag(nFilename, nStatus)
```

## Where :

nFilename	the filename to check
nStatus	the status to check (only for clsFileFlag) combine file <u>attributes</u> with logical OR.
test	TRUE if the specified flag is present FALSE if the specified flag is not present

## Comments :

IsFilenameValid checks only the validity of a file (normal file or network file) not the presence on a disk, the returned code can be :

IFV_ERROR	bad char in the filename
IFV_NAME_TOO_LONG	the length of the file part is too long (> 8)
IFV_EXT_TOO_LONG	the length of the extension part is too long (> 3)
IFV_TOO_MANY_BACKSLASH	too many successing backslash (> 2)
IFV_BAD_DRIVE_LETTER	bad drive letter before the colon ':'

IFV_BAD_COLON_POS	bad colon ':' position (<>2)
IFV_EXT_WITHOUT_NAME	extension without a name

If the filename is not a good filename or if the filename not exist or if an error occurs when accessing the filename, the return value is always FALSE.

**See also :** [ls](#)

# Huge memory array : Overview

The functions/subs used in the Huge Memory Arrays routines handle Huge Arrays.

Huge Arrays is based on the same principle that DISK ARRAY and MULTIPLE DISK ARRAY.

An bigger advantage of Huge Arrays is the speed.

The maximum number of Huge Arrays is 8192.

This number is a theoretical maximum and is depending of any application loaded in memory.

The following functions/subs are used to handle big sized arrays on disk :

<u>HMAClear</u>	Clear a Huge Array (fill it with chr\$(0) or chr\$(32) (for string array)).
<u>HMAClearCol</u>	Clear a single Col on on one Sheet or on all sheets in a Huge Array (see above).
<u>HMAClearRow</u>	Clear a single Row on one Sheet or on all Sheets in a Huge Array (see above).
<u>HMAClearSheet</u>	Clear a single Sheet in a Huge Array (fill it with chr\$(0) or chr\$(32) (for string array)).
<u>HMACreate</u>	Create a Huge Array.
<u>HMAFree</u>	Free a Huge Array.
<u>HMAGet</u>	Read an element from a Huge Array.
<u>HMAGetType</u>	Read a type'd variable from a Huge Array.
<u>HMAOnDisk</u>	Get/Put a Huge Array from/to a file on disk.
<u>HMAPut</u>	Save an element to a Huge Array.
<u>HMAPutType</u>	Save a type'd variable to a Huge Array.
<u>HMArGet</u>	Read an element from a Huge Array with only one sheet and one row.
<u>HMArGetType</u>	Read a type'd variable from a Huge Array with only one sheet and one row.
<u>HMArPut</u>	Save an element from a Huge Array with only one sheet and one row.
<u>HMArPutType</u>	Save a type'd variable from a Huge Array with only one sheet and one row.
<u>HMAAsClearCol</u>	Clear a single Col in a Huge Array with only one sheet.
<u>HMAAsClearRow</u>	Clear a single Row in a Huge Array with only one sheet.
<u>HMAAsGet</u>	Read an element from a Huge Array with only one sheet.
<u>HMAAsGetType</u>	Read a type'd variable from a Huge Array with only one sheet.
<u>HMAAsPut</u>	Save an element from a Huge Array with only one sheet.
<u>HMAAsPutType</u>	Save a type'd variable from a Huge Array with only one sheet.

Don't forget that any Huge Memory Arrays must be destroyed before quitting the application. If you not destroy all Huge Memory Arrays that you've created, the memory used will be only released when the DLL will be unloaded from memory.



# HMAPut, HMArPut, HMAsPut

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[HMAPut](#) save an element to a huge array.

[HMArPut](#) have the same fonctionnality but with a huge array with only one sheet and only one row.

[HMAsPut](#) have the same fonctionnality but with a huge array with only one sheet.

## Declare Syntax :

Declare Sub cHMAPut Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, Var As Variant)

Declare Sub cHMArPut Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Col As Long) As Variant

Declare Sub cHMAsPut Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, Var As Variant)

## Call Syntax :

Call cHMAPut(HMA, Row&, Col&, Sheet&, Var)

## Where :

HMA is a type'd variable (tagHMA).

Row& is the row.

Col& is the col.

Sheet& is the sheet.

Var is the variant value to save depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than HMA.nRows, the Row HMA.nRows is used.

If the Col is greater than HMA.nCols, the Col HMA.nCols is used.

If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

## Examples :

see [HMACreate](#)

**See also :** [Huge memory array](#)

# UUCP : Overview

[FileUUCP](#)

uuencode/uudecode a file (this is can be usefull for Internet).

# HMAGetType, HMArGetType, HMAsGetType

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[HMAGetType](#) read a type'd variable from a huge array.

[HMArGetType](#) have the same fonctionnality but with a huge array with only one sheet and only one row.

[HMAsGetType](#) have the same fonctionnality but with a huge array with only one sheet.

## Declare Syntax :

Declare Sub cHMAGetType Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)

Declare Sub cHMArGetType Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Col As Long, nType As Any)

Declare Sub cHMAsGetType Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, nType As Any)

## Call Syntax :

Call cHMAGetType(HMA, Row&, Col&, Sheet&, nType)

## Where :

HMA	is a type'd variable (tagHMA).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
nType	is the readed type'd variable depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than HMA.nRows, the Row HMA.nRows is used.

If the Col is greater than HMA.nCols, the Col HMA.nCols is used.

If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

## Examples :

```
Dim ErrCode           As Integer
Dim HMA               As tagHMA
Dim TE(1 To 4)        As tagTASKENTRY
```

```
HMA.nType = Len(TE(1))           ' positive value for a type'd variable
HMA.nIsTyped = True             ' init the array with chr$(0) because type'd
variable
HMA.nRows = 500                 ' 500 rows
HMA.nCols = 500                 ' 500 cols
HMA.nSheets = 2                 ' 2 sheets

ErrCode = cHMACreate(HMA)       ' use a created huge array

Call cHMAGetType(HMA, 1, 1, 1, TE(1)) ' read the type'd variable in Row 1, Col 1,
Sheet 1
Call cHMAGetType(HMA, 1, HMA.nCols, 1, TE(2)) ' read the type'd variable in Row 1, Col 500,
Sheet 1
Call cHMAGetType(HMA, HMA.nRows, 1, 1, TE(3)) ' read the type'd variable in Row 500, Col 1,
Sheet 1
```

Call `chMAGetType(HMA, HMA.nRows, HMA.nCols, 1, TE(4))`  
500, Sheet 1

' read the type'd variable in Row 500, Col

**See also :** [Huge memory array](#)

# HMAGet, HMArGet, HMAsGet

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[HMAGet](#) read an element from a huge array.

[HMArGet](#) have the same fonctionnality but with a huge array with only one sheet and only one row.

[HMAsGet](#) have the same fonctionnality but with a huge array with only one sheet.

## Declare Syntax :

Declare Function cHMAGet Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long) As Variant

Declare Function cHMArGet Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Col As Long) As Variant

Declare Function cHMAsGet Lib "[time2win.dll](#)" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long) As Variant

## Call Syntax :

Var = cHMAGet(HMA, Row&, Col&, Sheet&)

## Where :

HMA	is a type'd variable (tagHMA).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
Var	is the readed variant value depending of the variable type used in the creation.

## Comments :

If the Row is below 1, the Row 1 is used.

If the Col is below 1, the Col 1 is used.

If the Sheet is below 1, the Sheet 1 is used.

If the Row is greater than HMA.nRows, the Row HMA.nRows is used.

If the Col is greater than HMA.nCols, the Col HMA.nCols is used.

If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

## Examples :

see [HMACreate](#)

**See also :** [Huge memory array](#)

# HMAFree

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[HMAFree](#) free the memory used by a huge array.

## **Declare Syntax :**

Declare Function cHMAFree Lib "[time2win.dll](#)" (HMA As tagHMA) As Integer

## **Call Syntax :**

ErrCode = cHMAFree(HMA)

## **Where :**

HMA is a type'd variable (tagHMA).  
ErrCode% is the returned error code.

## **Comments :**

## **Examples :**

see [HMACreate](#)

**See also :** [Huge memory array](#)

# HMACreate

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

HMACreate create a new huge array.

## Declare Syntax :

Declare Function cHMACreate Lib "time2win.dll" (HMA As tagHMA) As Integer

## Call Syntax :

ErrCode% = cHMACreate(HMA)

## Where :

HMA is a type'd variable (tagHMA).  
ErrCode% is the returned error code.

## Comments :

In theory :

The maxixum number of Rows is 2147483647  
The maxixum number of Cols is 2147483647  
The maxixum number of Sheets is 2147483647

You are only limited by the size of the memory.

Bigger are nRows, nCols or nSheets, bigger is the time to initialize.

When you create a new huge array, the only parameters that you must initialize are :

HMA.nType = 50	'the type of the variable to use, see <u>Constants and</u>
<u>Types declaration</u> . (HMA_x)	
HMA.nIsTyped = False	'Must be True for a type'd variable.
HMA.nRows = 50	'the number of rows to use.
HMA.nCols = 50	'the number of cols to use.
HMA.nSheets = 2	'the number of sheets to use.

**YOU CAN'T CHANGE THESE PARAMETERS AFTER THE CREATION OF THE HUGE ARRAY.  
YOU CAN'T CHANGE THE OTHER VALUES IN THE TYPE'D VARIABLE.**

When you create a new array, all elements are initialized with chr\$(0) except for string array which are initialized with chr\$(32) (spaces).

However, string array and type'd array use the same positive value to define in .nType, but the type'd array must be initialized with chr\$(0) not with chr\$(32) therefore for a type'd you must specify .nIsTyped on True to initialize it with chr\$(0).

If you use huge array of type'd variable, the type'd variable can be only a mix of fixed variable (variable string length can't be used).

## Examples :

```
Dim ErrCode As Integer
Dim HMA As tagHMA
Dim Var(1 To 8) As Variant
```

```

HMA.nType = 50           ' positive value for a string
HMA.nIsTyped = False   ' init the array with spaces
HMA.nRows = 50          ' 50 rows
HMA.nCols = 50          ' 50 cols
HMA.nSheets = 2         ' 2 sheets

ErrCode = cHMACreate(HMA) ' create a new huge array

Call cHMAPut(HMA, 1, 1, 1, "D:1, ABCDEFGHIJ") ' save the string in Row 1, Col 1, Sheet 1
Call cHMAPut(HMA, 1, HMA.nCols, 1, "D:1, abcdefghij") ' save the string in Row 1, Col 50, Sheet 1
Call cHMAPut(HMA, HMA.nRows, 1, 1, "D:1, OPQRSTUVWXYZ") ' save the string in Row 50, Col 1, Sheet 1
Call cHMAPut(HMA, HMA.nRows, HMA.nCols, 1, "D:1, oprqstuvwxyz") ' save the string in Row 50, Col 50, Sheet 1

Call cHMAPut(HMA, 1, 1, 2, "D:2, 1234567890") ' save the string in Row 1, Col 1, Sheet 2
Call cHMAPut(HMA, 1, HMA.nCols, 2, "D:2, 0987654321") ' save the string in Row 1, Col 50, Sheet 2
Call cHMAPut(HMA, HMA.nRows, 1, 2, "D:2, 12345ABCDE") ' save the string in Row 50, Col 1, Sheet 2
Call cHMAPut(HMA, HMA.nRows, HMA.nCols, 2, "D:2, VWXYZ54321") ' save the string in Row 50, Col 50, Sheet 2

Var(1) = cHMAGet(HMA, 1, 1, 1) ' read the string in Row 1, Col 1, Sheet 1
Var(2) = cHMAGet(HMA, 1, HMA.nCols, 1) ' read the string in Row 1, Col 50, Sheet 1
Var(3) = cHMAGet(HMA, HMA.nRows, 1, 1) ' read the string in Row 50, Col 1, Sheet 1
Var(4) = cHMAGet(HMA, HMA.nRows, HMA.nCols, 1) ' read the string in Row 50, Col 50, Sheet 1

Var(5) = cHMAGet(HMA, 1, 1, 2) ' read the string in Row 1, Col 1, Sheet 2
Var(6) = cHMAGet(HMA, 1, HMA.nCols, 2) ' read the string in Row 1, Col 50, Sheet 2
Var(7) = cHMAGet(HMA, HMA.nRows, 1, 2) ' read the string in Row 50, Col 1, Sheet 2
Var(8) = cHMAGet(HMA, HMA.nRows, HMA.nCols, 2) ' read the string in Row 50, Col 50, Sheet 2

ErrCode = cHMAFree(HMA) ' free the memory used.

On my system :

ErrCode = -1 ' no error

HMA.daSize = 64 ' internal header size
HMA.nType = 50 ' fixed string of 50 chars
HMA.nRows = 50 ' 50 rows
HMA.nCols = 50 ' 50 cols
HMA.nSheets = 2 ' 2 sheets
HMA.rHandle = 0 ' internal handle
HMA.rElementSize = 50 ' internal size of a element
HMA.rFileSize = 250000 ' internal size of the memory used
HMA.rParts = 3 ' internal number of parts (block of 64000
chars)
HMA.rRemain = 58000 ' internal remain chars
HMA.rSheetSize = 2500 ' internal size of one sheet

Var(1) = "D:1, ABCDEFGHIJ"
Var(2) = "D:1, abcdefghij"
Var(3) = "D:1, OPQRSTUVWXYZ"
Var(4) = "D:1, oprqstuvwxyz"

Var(5) = "D:2, 1234567890"
Var(6) = "D:2, 0987654321"
Var(7) = "D:2, 12345ABCDE"
Var(8) = "D:2, VWXYZ54321"

```

**See also :** [Huge memory array](#)



# HMAClear, HMACClearSheet, HMACClearCol, HMAAsClearCol, HMACClearRow, HMAAsClearRow

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[HMAClear](#) clear a huge array (fill it with chr\$(0) or chr\$(32) (for string array)).

[HMACClearSheet](#) clear a single Sheet in a huge array (fill it with chr\$(0) or chr\$(32) (for string array)).

[HMACClearCol](#) clear a single Col on one Sheet or on all Sheets in a huge array (fill it with chr\$(0) or chr\$(32) (for string array)).

[HMAAsClearCol](#) have the same fonctionnality but with a huge array with only one sheet.

[HMACClearRow](#) clears a single Row on one Sheet or on all Sheets in a huge array (fill it with chr\$(0) or chr\$(32) (for string array)).

[HMAAsClearRow](#) have the same fonctionnality but with a huge array with only one sheet.

## **Declare Syntax :**

```
Declare Function cHMAClear Lib "time2win.dll" (HMA As tagHMA) As Integer
```

```
Declare Function cHMACClearSheet Lib "time2win.dll" (HMA As tagHMA, ByVal Sheet As Long) As Integer
```

```
Declare Function cHMACClearCol Lib "time2win.dll" (HMA As tagHMA, ByVal Col As Long, ByVal Sheet As Long) As Integer
```

```
Declare Function cHMAAsClearCol Lib "time2win.dll" (HMA As tagHMA, ByVal Col As Long) As Integer
```

```
Declare Function cHMACClearRow Lib "time2win.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Sheet As Long) As Integer
```

```
Declare Function cHMAAsClearRow Lib "time2win.dll" (HMA As tagHMA, ByVal Row As Long) As Integer
```

## **Call Syntax :**

```
ErrCode% = cHMAClear(HMA)
```

```
ErrCode% = cHMACClearSheet(HMA, Sheet&)
```

```
ErrCode% = cHMACClearCol(HMA, Col&, Sheet&)
```

```
ErrCode% = cHMAAsClearCol(HMA, Col&)
```

```
ErrCode% = cHMACClearRow(HMA, Row&, Sheet&)
```

```
ErrCode% = cHMAAsClearRow(HMA, Row&)
```

## **Where :**

HMA is a type'd variable (tagHMA).

Col& is the desired Col.

Row& is the desired Row.

Sheet& is the desired Sheet.

ErrCode% is the returned error code.

## **Comments :**

This function must be used only after you've created a huge array.

If you've created a huge array, the array is already cleared.

For cHMACClearSheet :

If the huge array have a single Sheet, this routine have the same effect that cHMAClear.

If the Sheet is -1 then all Sheets are used. This parameter have the same fonctionnality that cHMAClear

If the Sheet is below 1 and different of -1, the Sheet 1 is used.

If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.



# HMAOnDisk

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

HMAOnDisk read/write a Huge Array from/to a file.

## Declare Syntax :

Declare Function cHMAOnDisk Lib "time2win.dll" (HMA As tagHMA, ByVal hsFile As String, ByVal hsGetPut As Integer) As Long

## Call Syntax :

hsFileLength& = cHMAOnDisk(HMA, hsFile\$, hsGetPut%)

## Where :

HMA is a type'd variable (tagHMA).  
hsFile\$ is the name of the file to read/write the Huge Array.  
hsGetPut% PUT\_ARRAY\_ON\_DISK to put the array on disk,  
GET\_ARRAY\_ON\_DISK to get the array from disk.  
hsFileLength& >=0 is the returned length of the file,  
< 0 is an error occurs.

## Comments :

The file length is the size of the Huge Array.

## Examples :

```
Dim HMA As tagHMA
Dim ErrCode As Integer
```

```
HMA.nType = 50 ' positive value for a string
HMA.nIsTyped = False ' init the array with spaces
HMA.nRows = 50 ' 50 rows
HMA.nCols = 50 ' 50 cols
HMA.nSheets = 2 ' 2 sheets
```

```
ErrCode = cHMACreate(HMA)
```

```
If (ErrCode <> 0) Then
    MsgBox "Huge Array of " & HMA.rMemorySize & " bytes has been created with handle (" & HMA.rHandle & ")"
Else
    MsgBox "Huge Array of " & HMA.rMemorySize & " bytes can't be created."
End If
```

```
Call cHMAPut(HMA, 1, 1, 1, "D:1, ABCDEFGHIJ") ' save the string in Row 1, Col 1, Sheet 1
Call cHMAPut(HMA, 1, HMA.nCols, 1, "D:1, abcdefghij") ' save the string in Row 1, Col 50, Sheet 1
Call cHMAPut(HMA, HMA.nRows, 1, 1, "D:1, OPQRSTUVWXYZ") ' save the string in Row 50, Col 1, Sheet 1
Call cHMAPut(HMA, HMA.nRows, HMA.nCols, 1, "D:1, oprqstuvwxyz") ' save the string in Row 50, Col 50, Sheet 1
```

```
MsgBox "The length of the saved file is " & cHMAOnDisk(HMA, "c:\hugestr.tmp", PUT_ARRAY_ON_DISK)
```

```
ErrCode = cHMAClear(HMA)
```

```
MsgBox "The length of the readed file is " & cHMAOnDisk(HMA, "c:\hugestr.tmp", GET_ARRAY_ON_DISK)
```

```
ErrCode = cHMAFree(HMA)
```

```
If (ErrCode = TRUE) Then
    MsgBox "Huge Array (" & hsHandle & ") has been destroyed."
Else
    MsgBox "Huge Array (" & hsHandle & ") can't be destroyed."
End If
```

**See also :** [Huge memory array](#)

' structure for huge memory array

```
Type tagHMA
  daSize      As Integer      'size of the type'd
  nTypeAs Integer 'variable type
  nRows       As Long         'number of rows
  nCols As Long 'number of cols
  nSheets     As Long         'number of sheets
  rHandle     As Long         'returned handle for use with other functions
  rElementSize As Long        'returned size of a element
  rMemorySize As Long         'returned size of the memory used
  rPartsAs Long 'returned total part
  rRemain     As Long         'returned size of the remain part
  rSheetSize  As Long         'size of a sheet
  rOffset     As Long         'returned offset
  nIsTyped    As Integer      'is nType a type'd variable
  Dummy      As String * 20   'reserved for future use
End Type
```

' definition for variable type in huge memory array

```
Public Const HMA_TYPE = 0
Public Const HMA_BYTE = -1
Public Const HMA_INTEGER = -2
Public Const HMA_LONG = -3
Public Const HMA_SINGLE = -4
Public Const HMA_DOUBLE = -5
Public Const HMA_CURRENCY = -6
```

' definition for error type in huge memory array

```
Public Const HMA_NO_ERROR = True
Public Const HMA_NO_MEMORY = 1
Public Const HMA_BAD_TYPE = 2
Public Const HMA_BAD_ROWS = 3
Public Const HMA_BAD_COLS = 4
Public Const HMA_BAD_SHEETS = 5
Public Const HMA_INVALID_HANDLE = 6
```

# Encryption : Overview

<a href="#">Decrypt</a>	decode a string encoded with Encrypt function.
<a href="#">DESdecrypt</a>	decode a string with a password using the U.S. Data Encryption Standard cipher.
<a href="#">DIAMONDdecrypt</a>	decode a string with a password using the Diamond Encryption Algorithm (4 modes).
<a href="#">DESdecryptFile</a>	copy one file to an another file but with U.S. Data Encryption Standard cipher.
<a href="#">DIAMONDdecryptFile</a>	copy one file to an another file but with Diamond Encryption Algorithm (4 modes).
<a href="#">DESencrypt</a>	encode a string with a password using the U.S. Data Encryption Standard cipher.
<a href="#">DIAMONDencrypt</a>	encode a string with a password using the Diamond Encryption Algorithm (4 modes).
<a href="#">DESencryptFile</a>	copy one file to an another file but with U.S. Data Encryption Standard cipher.
<a href="#">DIAMONDencryptFile</a>	copy one file to an another file but with Diamond Encryption Algorithm (4 modes).
<a href="#">Encrypt</a>	encode a string with a password/key.
<a href="#">FileEncrypt</a>	copy one file to an another file but with encryption.
<a href="#">FileDecrypt</a>	copy one file to an another file but with decryption.
<a href="#">IDEAdecrypt</a>	decode a string with a password using the International Data Encryption Algorithm cipher.
<a href="#">IDEAdecryptFile</a>	copy one file to an another file but with the International Data Encryption Algorithm cipher.
<a href="#">IDEAencrypt</a>	encode a string with a password using the International Data Encryption Algorithm cipher.
<a href="#">IDEAencryptFile</a>	copy one file to an another file but with the International Data Encryption Algorithm cipher.
<a href="#">RUBYdecrypt</a>	decode a string with a password using the RUBY algorithm (7 modes).
<a href="#">RUBYdecryptFile</a>	copy one file to an another file but with RUBY algorithm (7 modes).
<a href="#">RUBYencrypt</a>	encode a string with a password using the RUBY algorithm (7 modes).
<a href="#">RUBYencryptFile</a>	copy one file to an another file but with RUBY algorithm (7 modes).

# Encrypt, Decrypt

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Encrypt](#) encode a string with a password/key.

[Decrypt](#) decode a string encoded with Encrypt function.

## Declare Syntax :

Declare Function cEncrypt Lib "time2win.dll" (Txt As String, password As String, ByVal level As Integer) As String

Declare Function cDecrypt Lib "time2win.dll" (Txt As String, password As String, ByVal level As Integer) As String

## Call Syntax :

testE = cEncrypt(Txt, password, level)

testD = cDecrypt(Txt, password, level)

## Where :

Txt	is the string to encrypt/decrypt
password	is the key to use for encryption/decryption
level	level of the encryption/decryption
test	is the string encrypted/decrypted

## Comments :

The password/key is case sensitive.

The [level](#) is a number between **0** and **4**.

Higher is the level, better is the encryption

You must use the same level for encrypt/decrypt a given string.

## Examples :

Txt = "Under the blue sky, the sun is yellow"

password = "a new encryption"

level = ENCRYPT\_LEVEL\_4

test = cEncrypt(Txt, password, level)

Txt = cDecrypt(test, password, level)

**See also :** [Encryption](#)

# FileEncrypt, FileDecrypt

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileEncrypt](#) copy one file to an another file but with encryption.

[FileDecrypt](#) copy one file to an another file but with decryption.

## Declare Syntax :

Declare Function cFileEncrypt Lib "[time2win.dll](#)" (ByVal file1 As String, ByVal file2 As String, Password As String, ByVal Level As Integer) As Long

Declare Function cFileDecrypt Lib "[time2win.dll](#)" (ByVal file1 As String, ByVal file2 As String, Password As String, ByVal Level As Integer) As Long

## Call Syntax :

```
test& = cFileEncrypt(file1, file2, password, level)
```

```
test& = cFileDecrypt(file1, file2, password, level)
```

## Where :

file1\$	is the source file.
file2\$	is the destination file.
password	is the key to use for encryption/decryption.
level	level of the encryption/decryption.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

The password/key is case sensitive.

The [level](#) is a number between **0** and **4**.

Higher is the level, better is the encryption.

You must use the same level for encrypt/decrypt a given string.

The returned value can be negative and have the following value :

-1	the password is an EMPTY string.
-32720	the number of chars in a block for writing differs from the number of chars for reading.
-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.
-32760	allocation error for memory buffer 1.
-32761	allocation error for memory buffer 2.

## Examples :

```
test& = cFileEncrypt("c:\autoexec.bat", "c:\autoexec.tb1", "Time To Win", ENCRYPT_LEVEL_4)
```

```
test& = cFileDecrypt("c:\autoexec.tb1", "c:\autoexec.tb2", "Time To Win", ENCRYPT_LEVEL_4)
```

**See also :** [Encryption](#)



' definition for encrypt/decrypt

Public Const ENCRYPT\_LEVEL\_0 = 0

Public Const ENCRYPT\_LEVEL\_1 = 1

Public Const ENCRYPT\_LEVEL\_2 = 2

Public Const ENCRYPT\_LEVEL\_3 = 3

Public Const ENCRYPT\_LEVEL\_4 = 4

## Crc32 : Overview

FileCRC32 calculate a 32 bits CRC for a given file.  
StringCRC32 calculate a 32 bits CRC for a given string.

# Crypt, FileCrypt

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**Crypt** encrypt/decrypt a string with a password.

**FileCrypt** encrypt/decrypt a file with a password.

## Declare Syntax :

Declare Function cCrypt Lib "time2win.dll" (Txt As String, ByVal Password As String) As String

Declare Function cFileCrypt Lib "time2win.dll" (ByVal File1 As String, ByVal File2 As String, ByVal Password As String) As Long

## Call Syntax :

strResult\$ = cCrypt(Txt\$, Password\$)

lngResult& = cFileCrypt(File1\$, File2\$, Password\$)

## Where :

Txt\$	is the string to be encrypted/decrypted
Password\$	is the string to encrypt/decrypt
File1\$	is the file to be encrypted/decrypted
File2\$	is the file encrypted/decrypted
strResult\$	is the string encrypted/decrypted
lngResult&	< 0 : an error has occurred > 0 : length of the file encrypted

## Comments :

## Examples :

For cCrypt :

```
Dim lngResult As Long
Dim strResult As String
Dim strDisplay As String
```

```
Dim Str1 As String
Dim Str2 As String
Dim Str3 As String
```

```
strResult = ""
strDisplay = ""
```

```
Str1 = "T2WIN-32, t2win-32, T2WIN-32, t2win-32, T2WIN-32, t2win-32"
Str2 = cCrypt(Str1, "1234567")
Str3 = cCrypt(Str2, "1234567")
```

```
strDisplay = strDisplay & "Crypt " & Str1 & "" & vbCrLf & "with password '1234567'" & vbCrLf & "is" & vbCrLf & "" & Str2 & "" & vbCrLf & vbCrLf
strDisplay = strDisplay & "Crypt " & Str2 & "" & vbCrLf & "with password '1234567'" & vbCrLf & "is" & vbCrLf & "" & Str3 & "" & vbCrLf & vbCrLf
strDisplay = strDisplay & "Compare string contents (not sensitive) is " & If(LCase$(Str1) = LCase$(Str3), "same", "not same") & vbCrLf & vbCrLf
```

```
Str1 = String$(30, "a") + String$(6, "b") + String$(5, "c") + String$(4, "d")
```

```
Str2 = cCrypt(Str1, "1234567")
Str3 = cCrypt(Str2, "1234567")
```

```
strDisplay = strDisplay & "Crypt " & Str1 & "" & vbCrLf & "with password '1234567'" & vbCrLf & "is" & vbCrLf & "" & Str2 & "" & vbCrLf & vbCrLf
strDisplay = strDisplay & "Crypt " & Str2 & "" & vbCrLf & "with password '1234567'" & vbCrLf & "is" & vbCrLf & "" & Str3 & "" & vbCrLf & vbCrLf
strDisplay = strDisplay & "Compare string contents (not sensitive) is " & IIf(LCase$(Str1) = LCase$(Str3), "same", "not same") & vbCrLf & vbCrLf
```

```
Debug.Print strDisplay
```

For cFileCrypt :

```
Dim lngResult As Long
Dim strResult As String
Dim strDisplay As String
```

```
Dim Str1 As String
Dim Str2 As String
Dim Str3 As String
```

```
strResult = ""
strDisplay = ""
```

```
File1 = T2WFileTest
File2 = "autoexec.hi-encrypted"
File3 = "autoexec.hi-decrypte"
```

```
strDisplay = strDisplay & "File Crypt " & File1 & " to " & File2 & " with password '1234567' is " & cFileCrypt(File1, File2, "1234567") & vbCrLf
strDisplay = strDisplay & "File Crypt " & File2 & " to " & File3 & " with password '1234567' is " & cFileCrypt(File2, File3, "1234567") & vbCrLf
strDisplay = strDisplay & "Compare File contents (not sensitive) " & File1 & " with " & File3 & " is " & IIf(cCmpFileContents(File1, File3, False) = -1, "same", "not same") & vbCrLf & vbCrLf
```

```
Debug.Print strDisplay
```

**See also :** [Hi-Crypt](#)

# FileCRC32

## Purpose :

[FileCRC32](#) calculate a 32 bits CRC for a given file.

## Declare Syntax :

Declare Function cFileCRC32 Lib "[time2win.dll](#)" (ByVal lpFilename As String, ByVal mode As Integer) As Long

## Call Syntax :

```
test = cFileCRC32(lpFilename, mode)
```

## Where :

lpFilename	the file to proceed
mode	OPEN_MODE_BINARY (calculates the CRC on the full length of the file). This is the default mode. OPEN_MODE_TEXT (calculates the CRC until a EOF is encountered)
test	the calculated CRC 32 bits in a LONG.

## Comments :

The returned value can be negative and have only a value :

-1 If the filename is not a good filename or if the filename not exist or if an error occurs when accessing the filename.

## Examples :

```
test = cFileCRC32("C:\COMMAND.COM") ' &h1131ADD3 (MS-DOS 6.22)
```

**See also :** [Crc32](#)

# StringCRC32

**Purpose :**

[StringCRC32](#) calculate a 32 bits CRC for a given string.

**Declare Syntax :**

Declare Function cStringCRC32 Lib "[time2win.dll](#)" (Txt As String) As Long

**Call Syntax :**

```
test = cStringCRC32(Txt)
```

**Where :**

Txt	the string to proceed
test	the calculated CRC 32 bits in a LONG.

**Comments :**

if the string if empty, the return value is always -1 (&hFFFFFFF).

**Examples :**

test = cStringCRC32("ABCDEFGG")	' &hE6F94BC
test = cStringCRC32("GFEDCBA")	' &hF0EC0AB3

**See also :** [Crc32](#)

' definition for crc32

Public Const OPEN\_MODE\_BINARY = 0

Public Const OPEN\_MODE\_TEXT = 1

' structure for file attributes

Type FileAttributeType

ErrNo As Integer

Archive As Integer

Hidden As Integer

Normal As Integer

ReadOnly As Integer

SubDir As Integer

System As Integer

Compressed As Integer

End Type



# Hi-Crypt : Overview

Crypt

encrypt/decrypt a string with a password.

FileCrypt

encrypt/decrypt a file with a password.

# Serialization : Overview

Serialization is a set of routines primarily intended for developers so that they may append a serial number (or other identifier) to the end of an .exe, .dll or any static files in size, put/modify or get serial numbers or any string to 50 characters. Users may use to initialize purchased software applications with ownership, security-related, or other identifying marks.

A unique serial number going out with each copy of an application affords the developer with a possible opportunity to identify, if need be, the registered client of a particular copy. The end-user is normally unaware of the existence of such a mark, its location, its method of placement or the method of reading/verifying. Its absence or modification may provide evidence of tampering.

The serialization of a file adds an overhead of 200 bytes (in 16-Bit) and 280 bytes (in 32-Bit) to the specified file.

<u>IsSerial</u>	check if a file has been serialized.
<u>SerialGet</u>	get the serialization information from a serialized file.
<u>SerialInc</u>	increment by a value the serialized number part of a serialized file.
<u>SerialPut</u>	put a serialization information to a serialized file.
<u>SerialRmv</u>	remove the serialization information from a serialized file.

# IsSerial, SerialGet, SerialInc, SerialPut, SerialRmv

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**IsSerial** check if a file has been serialized.

**SerialGet** get the serialization information from a serialized file.

**SerialInc** increment by a value the serialized number part of a serialized file.

**SerialPut** put a serialization information to a serialized file.

**SerialRmv** remove the serialization information from a serialized file.

## Declare Syntax :

```
Declare Function clsSerial Lib "time2win.dll" (ByVal File As String) As Integer
Declare Function cSerialGet Lib "time2win.dll" (ByVal file As String, SERIALDATA As tagSERIALDATA) As Integer
Declare Function cSerialInc Lib "time2win.dll" (ByVal file As String, ByVal Increment As Long) As Integer
Declare Function cSerialPut Lib "time2win.dll" (ByVal file As String, SERIALDATA As tagSERIALDATA) As Integer
Declare Function cSerialRmv Lib "time2win.dll" (ByVal File As String) As Integer
```

## Call Syntax :

```
Test% = clsSerial(File$)
Test% = cSerialGet(File$, SERIALDATA)
Test% = cSerialInc(File$, Increment&)
Test% = cSerialPut(File$, SERIALDATA)
Test% = cSerialRmv(File$)
```

## Where :

File\$	is the specified file.
SERIALDATA	is a type'd variable ( <a href="#">tagSERIALDATA</a> ).
Increment&	is the increment (positive or negative).
Test%	TRUE if all is ok, <> TRUE if an error has occurred.

## Comments :

For 16-Bit :

The length of the serialization string is maximum 50 characters (SERIALDATA.Description1, SERIALDATA.Description2).

You can add/remove the serialization on the same EXE file for application. In other words, the EXE file corresponding to an application can add/remove the serialization on itself.

For 32-Bit :

The length of the serialization string is maximum 52 characters (SERIALDATA.Description1, SERIALDATA.Description2).

**Due to some limitations (or some protections) in Win95/WinNT, you can't add/remove the serialization on the same EXE file for application. In other words, the EXE file corresponding to an application can't add/remove the serialization on itself.**

For SerialInc :

If you pass a 0 value, the serialization number is reset to 0 (be care).

**Examples :**

```
Dim putSERIALDATA As tagSERIALDATA
Dim getSERIALDATA As tagSERIALDATA
```

```
putSERIALDATA.Description1 = "1234567890123456789012345"
putSERIALDATA.Description2 = ""
putSERIALDATA.Number = 987654321
Debug.Print cSerialPut("c:\tmp\sample.exe", putSERIALDATA)
Debug.Print cSerialGet("c:\tmp\sample.exe", getSERIALDATA)
Debug.Print getSERIALDATA.Description1 & Chr$(13) & getSERIALDATA.Description2 & Chr$(13) &
getSERIALDATA.Number
```

```
putSERIALDATA.Description2 = "ABCDEFGHJKLMNOPQRSTUVWXYZ"
putSERIALDATA.Number = 123456789
Debug.Print cSerialPut("c:\tmp\sample.exe", putSERIALDATA)
Debug.Print cSerialGet("c:\tmp\sample.exe", getSERIALDATA)
Debug.Print getSERIALDATA.Description1 & Chr$(13) & getSERIALDATA.Description2 & Chr$(13) &
getSERIALDATA.Number
```

```
Debug.Print cSerialInc("c:\tmp\sample.exe", 123)
Debug.Print cSerialGet("c:\tmp\sample.exe", getSERIALDATA)
Debug.Print getSERIALDATA.Description1 & Chr$(13) & getSERIALDATA.Description2 & Chr$(13) &
getSERIALDATA.Number
```

```
Debug.Print cSerialRmv("c:\tmp\sample.exe")
```

**See also :** [Serialization](#)

# Compress

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[Compress](#) remove all chr\$(0):ASCII NULL, chr\$(9):TAB, chr\$(32):SPACE from a string.

## **Declare Syntax :**

Declare Function cCompress Lib "[time2win.dll](#)" (Txt As String) As String

## **Call Syntax :**

test = cCompress(Txt)

## **Where :**

Txt	the string to proceed
test	the string returned without any chr\$(0), chr\$(9), chr\$(32)

## **Comments :**

**See also** : [String](#)

' structure for serialization

Type tagSERIALDATA

Description1	As String * 52	' serialization description 1
Description2	As String * 52	' serialization description 2
Number	As Long	' serialization number
Dummy	As String * 52	' reserved for future use

End Type

' definition for error type in SERIAL DATA

Public Const SD\_SERIAL\_NOT\_FOUND = 1

Public Const SD\_CAN\_NOT\_OPEN\_FILE = 2

# CompressTab, ExpandTab

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CompressTab](#) pack all n space chars into a tab char.

[ExpandTab](#) unpack all tab chars into n space chars.

## Declare Syntax :

Declare Function cCompressTab Lib "[time2win.dll](#)" (Txt As String, ByVal nTab As Integer) As String

Declare Function cExpandTab Lib "[time2win.dll](#)" (Txt As String, ByVal nTab As Integer) As String

## Call Syntax :

```
test = cCompressTab(Txt, nTabC)
```

```
test = cExpandTab(Txt, nTabE)
```

## Where :

Txt	the string to proceed.
nTabC	the number of space chars to replace by a tab char.
nTabE	the number of space chars which replace a tab char.
test	the result.

## Comments :

## Examples :

```
Txt = "A" + space$(2) + "B" + space$(3) + "C" + space$(4) + "D"
```

```
nTabC = 2
```

```
test = cCompressTab(Txt, nTabC)           ' test = "A" + chr$(9) + "B" + chr$(9) + space$(1) + "C" + chr$(9) + chr$(9) + "D"
```

```
Txt = test = "A" + chr$(9) + "B" + chr$(9) + space$(1) + "C" + chr$(9) + chr$(9) + "D"
```

```
nTabE = 2
```

```
test = cExpandTab(Txt, nTabE)           ' test = "A" + space$(2) + "B" + space$(3) + "C" + space$(4) + "D"
```

**See also :** [String](#)

# ChangeChars, ChangeCharsUntil

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ChangeChars](#) change all chars speciefin by others chars in a string.

[ChangeCharsUntil](#) change all chars speciefin by others chars in a string until a char is encountered.

## Declare Syntax :

Declare Sub cChangeChars Lib "[time2win.dll](#)" (Txt As String, charSet As String, newCharSet As String)

Declare Sub cChangeCharsUntil Lib "[time2win.dll](#)" (Txt As String, charSet As String, newCharSet As String, nUntil As String)

## Call Syntax :

Call cChangeChars(Txt, charSet, newCharSet)

Call cChangeCharsUntil(Txt, charSet, newCharSet, nUntil)

## Where :

Txt	the string to process.
charSet	the chars in the string to be changed.
newCharSet	the new chars.
nUntil	the char to stop the change.

## Comments :

Normally, the size of the newCharSet and charSet must be the same.

If the size are not the same, the smallest size is used.

For cChangeCharsUntil :

If the size of nUntil is 0 then all chars of the string is proceeded.

If the size of nUntil is >1 only the first char is used.

## Examples :

For cChangeChars :

```
Txt = "ABCDEF"  
charSet = "ACE"  
newCharSet = "ace"  
Call cChangeChars(Txt, charSet, newCharSet) ' Txt = "aBcDeF"
```

For cChangeCharsUntil :

```
Txt = "ABCDEF"  
charSet = "ACE"  
newCharSet = "ace"  
nUntil = "D"  
Call cChangeCharsUntil(Txt, charSet, newCharSet, nUntil) ' Txt = "aBcDEF"
```

**See also** : [String](#)



# CheckChars

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CheckChars](#) verify that all chars specifiens are present in a string.

## Declare Syntax :

Declare Function cCheckChars Lib "[time2win.dll](#)" (Txt As String, charSet As String) As Integer

## Call Syntax :

```
status = cCheckChars(Txt, charSet)
```

## Where :

Txt	the string to proceed.
charSet	the chars to be verified.
status	TRUE if all chars specifiens in charSet are present in Txt. FALSE if all chars specifiens in charSet are not present in Txt.

## Comments :

## Examples :

```
Txt = "ABCDEFGH"  
charSet = "CAD"  
status = cCheckChars(Txt, charSet)' status = TRUE
```

```
Txt = "ABCDEFGH"  
charSet = "CADZ"  
status = cCheckChars(Txt, charSet)' status = FALSE
```

**See also :** [String](#)

# RemoveBlockChar, RemoveOneChar

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[RemoveBlockChar](#) remove a block of chars at the specified position in a string.

[RemoveOneChar](#) remove one char at the specified position in a string.

## Declare Syntax :

Declare Function cRemoveBlockChar Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Long, ByVal Length As Long) As String

Declare Function cRemoveOneChar Lib "[time2win.dll](#)" (Txt As String, ByVal Position As Long) As String

## Call Syntax :

Test\$ = cRemoveBlockChar(Txt\$, Position&, Length&)

Test\$ = cRemoveOneChar(Txt\$, Position&)

## Where :

Txt\$	is the string to proceed.
Position&	is the starting position to remove the char(s).
Length&	is the number of chars to remove
Test\$	is the result

## Comments :

## Examples :

```
Txt$ = "This is an another test"  
Debug.Print cRemoveBlockChar(Txt$, 10, 9)           ' "This is a test"
```

```
Txt$ = "This is an test"  
Debug.Print cRemoveOneChar(Txt$, 10)              ' "This is  test"
```

**See also :** [String](#)

# Reverse

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[Reverse](#) reverse all chars in a given string.

## **Declare Syntax :**

Declare Function cReverse Lib "[time2win.dll](#)" (Txt As String) As String

## **Call Syntax :**

Test\$ = cReverse(Txt\$)

## **Where :**

Txt\$	is the specified string
Test\$	is the string reversed

## **Comments :**

## **Examples :**

Test\$ = cReverse("TIME TO WIN")            ' "NIW OT EMIT"

**See also :** [String](#)

# ScrollL, ScrollR

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[ScrollL](#) scroll one char to the left of a specified string.

[ScrollR](#) scroll one char to the right of a specified string.

## **Declare Syntax :**

```
Declare Function cScrollL Lib "time2win.dll" (Txt As String) As String
Declare Function cScrollR Lib "time2win.dll" (Txt As String) As String
```

## **Call Syntax :**

```
test$ = cScrollL(Txt$)
test$ = cScrollR(Txt$)
```

## **Where :**

Txt\$                            is the string to scroll.  
test\$                           is the string scrolled to the left or to the right.

## **Comments :**

The size of the string must be greater than 1.

## **Examples :**

```
Txt$ = "TIME TO WIN "
```

```
test$ = cScrollL(Txt$)                    "IME TO WIN T"
test$ = cScrollR(Txt$)                   " TIME TO WIN"
```

**See also :** [String](#)

# Count

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[Count](#) count the number of a specified char in a string.

## **Declare Syntax :**

Declare Function cCount Lib "[time2win.dll](#)" (Txt As String, Separator As String) As Integer

## **Call Syntax :**

```
test = cCount(Txt, Separator)
```

## **Where :**

Txt	the string to proceed
Separator	the char to be counted
test	the total number of Separator in the string

## **Comments :**

## **Examples :**

```
Txt = "A/BC/DEF/G"  
Separator = "/"  
test = cCount(Txt, Separator)           ' test = 3
```

**See also :** [String](#)

# ResizeString, ResizeStringAndFill

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ResizeString](#) resize the size of a string to a new length.

[ResizeStringAndFill](#) resize the size of a string to a new length and fill it with chars if the new length is greater than the current length.

## Declare Syntax :

```
Declare Function cResizeString Lib "time2win.dll" (Txt As String, ByVal newLength As Integer) As String
Declare Function cResizeStringAndFill Lib "time2win.dll" (Txt As String, ByVal newLength As Integer, Fill As String) As String
```

## Call Syntax :

```
Test$ = cResizeString(Txt$, Length%)
Test$ = cResizeStringAndFill(Txt$, Length%, Fill$)
```

## Where :

Txt\$ is the specified string.  
Length% is the new length (can be shorter than the current length).  
Fill\$ is a char or a string to use to fill the new string.  
Test\$ is the new string.

## Comments :

For cResizeString :

The new length can be greater than the current length. In this case, chr\$(0) is used to fill the rest of the string.

For cResizeStringAndFill :

The new length can be greater than the current length. In this case, the fill string is used to fill the rest of the string.

## Examples :

```
Test$ = cResizeString("TIME TO WIN", 7)           ' "TIME TO"

Test$ = cResizeStringAndFill("TIME TO WIN", 21, "@")  ' "TIME TO WIN@@@@@@@@@@@@@@"
Test$ = cResizeStringAndFill("TIME TO WIN", 21, "time") ' "TIME TO WINtimetimetit"
```

**See also :** [String](#)

# SwapD, SwapI, SwapL, SwapS, SwapStr

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SwapD](#) swap two Double values.

[SwapI](#) swap two Integer values.

[SwapL](#) swap two Long values.

[SwapS](#) swap two Single values.

[SwapStr](#) swap two strings.

## Declare Syntax :

```
Declare Sub cSwapD Lib "time2win.dll" (swap1 As Double, swap2 As Double)
```

```
Declare Sub cSwapI Lib "time2win.dll" (swap1 As Integer, swap2 As Integer)
```

```
Declare Sub cSwapL Lib "time2win.dll" (swap1 As Long, swap2 As Long)
```

```
Declare Sub cSwapS Lib "time2win.dll" (swap1 As Single, swap2 As Single)
```

```
Declare Sub cSwapStr Lib "time2win.dll" (swap1 As String, swap2 As String)
```

## Call Syntax :

```
Call cSwapD(swap1, swap2)
```

```
Call cSwapI(swap1, swap2)
```

```
Call cSwapL(swap1, swap2)
```

```
Call cSwapS(swap1, swap2)
```

```
Call cSwapStr(swap1, swap2)
```

## Where :

swap1            first Double/Integer/Long/Single/String value.

swap2            second Double/Integer/Long/Single/String value.

## Comments :

## Examples :

```
swap1 = 2345.12
```

```
swap2 = 5432.21
```

```
Call cSwapD(swap1, swap2)                    ' swap1 = 5432.21; swap2 = 2345.12
```

```
swap1 = "Hello"
```

```
swap2 = "World"
```

```
Call cSwapStr(swap1, swap2)                 ' swap1 = "World"; swap2 = "Hello"
```

**See Also :** [Miscellaneous](#)

# CreateAndFill

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CreateAndFill](#) create a string with the specified size and fill it with some chars.

## Declare Syntax :

Declare Function cCreateAndFill Lib "[time2win.dll](#)" (ByVal Length As Integer, Txt As String) As String

## Call Syntax :

```
test = cCreateAndFill(Length, Txt)
```

## Where :

Length	the length of the result string
Txt	the chars to fill in the result string
test	the result

## Comments :

## Examples :

```
Length = 14  
Txt = "aBc"  
test = cCreateAndFill(Length, Txt)           ' test = "aBcaBcaBcaBcaB"
```

**See also :** [String](#)



# Fill

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[Fill](#) fill a string with some chars.

## **Declare Syntax :**

Declare Sub cFill Lib "[time2win.dll](#)" (Txt As String, Fill As String)

## **Call Syntax :**

Call cCreateAndFill(Txt, Fill)

## **Where :**

Txt	the string to proceed
Fill	the chars to fill in the string

## **Comments :**

This routine is a superset of String\$. In fact, STRING\$ can only use a char to fill a string.

## **Examples :**

```
Txt = space$(14)
Fill = "AbC"
Call cFill(Txt, Fill)      ' test = "AbCAbCAbCAbCAb"
```

**See also :** [String](#)

# Lrc

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

Lrc calculate the LRC of a given string.

## **Declare Syntax :**

Declare Function cLrc Lib "time2win.dll" (Txt As String) As String

## **Call Syntax :**

```
test$ = cLrc(Txt)
```

## **Where :**

Txt	the string to proceed
test\$	the LRC calculated

## **Comments :**

The LRC is always an Hexa string of two chars.  
This function is used for communication between a program and a clocking terminal

## **Examples :**

```
test$ = cLrc(chr$(2) & "0a12721536")           ' "54"
```

**See also :** [String](#)

# Compact, Uncompact

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Compact](#) compact a string composed of numeric chars.

[Uncompact](#) uncompact a string composed of numeric chars.

## Declare Syntax :

Declare Function cCompact Lib "time2win.dll" (Txt As String) As String

Declare Function cUncompact Lib "time2win.dll" (Txt As String) As String

## Call Syntax :

test = cCompact(Txt)

test = cUncompact(Txt)

## Where :

Txt is the string (only numeric chars) to compact/uncompact.

test return the string compacted/uncompact.

## Comments :

For Compact :

If the size of the string is not a multiple of 2, the size used is the nearest below multiple of 2.

For Uncompact :

The size of the returned string is always a multiple of 2.

## Examples :

Txt = "39383736353433323130"

test = cCompact(Txt)

' test = "9876543210"

Txt = "0123456789"

test = cUncompact(Txt)

' test = "30313233343536373839"

**See also** : [String](#)

# MixChars

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[MixChars](#) will mix all chars in a given string in a random position.

## **Declare Syntax :**

Declare Function cMixChars Lib "[time2win.dll](#)" (Txt As String) As String

## **Call Syntax :**

test\$ = cMixChars(Txt)

## **Where :**

Txt                                    is the string to mix all chars.  
test\$                                   is the returned mixed string.

## **Comments :**

MixChars use a random number generator to perform the mix of the chars.  
The starting random number is depending of the actual date and time.

If the passed string is an EMPTY string, the returned string is an EMPTY string.

## **Examples :**

test1\$ = cMixChars("TIME TO WIN")                                    ' "ON EI WMTIT"  
test2\$ = cMixChars("Nothing can beat the fox")                                    ' "Nt honn ia ttechx baefog"

**See also :** [String](#)

# Align

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Align](#) align a give string (left, center, right) into an another new string.

## Declare Syntax :

Declare Function cAlign Lib "[time2win.dll](#)" (Txt As String, ByVal TypeAlign As Integer, ByVal NewLength As Integer) As String

## Call Syntax :

Test\$ = cAlign(Txt\$, TypeAlign%, NewLength%)

## Where :

Txt\$	is the specified string
TypeAlign%	< 0 : left align, = 0 : center align, > 0 : right align.
NewLength%	the length of the new string
Test\$	is the string aligned

## Comments :

If NewLength is below that the length of the string, the left part of the string is returned. The new string is padded with spaces.

## Examples :

Test\$ = cAlign("TIME TO WIN", -1, 20)	' "TIME TO WIN      "
Test\$ = cAlign("TIME TO WIN", 0, 20)	' "   TIME TO WIN   "
Test\$ = cAlign("TIME TO WIN", 1, 20)	' "           TIME TO WIN"

**See also :** [String](#)

# ProperName

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ProperName](#) convert the first letter of each word separated by a space in a string to upper case.

## Declare Syntax :

Declare Function cProperName Lib "[time2win.dll](#)" (Txt As String) As String

## Call Syntax :

Test\$ = cProperName(Txt\$)

## Where :

Txt\$                      is the specified string.  
Test\$                     is the returned string.

## Comments :

## Examples :

macdonald	becomes	Macdonald
mac donald	becomes	Mac Donald
John fitz,jr	becomes	John Fitz,jr
john Fitz, jr	becomes	John Fitz, Jr

**See also :** [String](#)

# ProperName2

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**ProperName2** convert the first letter of some words separated by a space or punctuation in upper letter case.

## Declare Syntax :

Declare Function cProperName2 Lib "time2win.dll" (Txt As String, ByVal TokenToUse As String, ByVal Options As Integer) As String

## Call Syntax :

Test\$ = cProperName2(Txt\$, TokenToUse\$, Options%)

## Where :

Txt\$	is the text to convert.
TokenToUse\$	is the token list that can't be converted.
Options%	PN_UPPERCASE, works with upper case text. PN_PUNCTUATION, separator can be a space or a punctuation. PN_KEEP_ORIGINAL, keep case letter in the token list. PN_ONLY_LEADER_SPACE, don't use the leader trailer space for search in the token list.

## Comments :

TokenToUse can be empty.

TokenToUse is a list of all words (separated by '/') which can't be converted (b.e. : "the/and/a/an/or/of")

## Examples :

ProperName2 of 'JOHN FITZ,JR' is 'John Fitz,Jr'  
ProperName2 of 'john Fitz,jr' is 'John Fitz,Jr'  
ProperName2 of 'macdonald' is 'Macdonald'  
ProperName2 of 'mac donald' is 'Mac Donald'  
ProperName2 of 'a.l. greene jr.' is 'A.L. Greene Jr.'  
ProperName2 of 'shale and sandstone and till' is 'Shale and Sandstone and Till'  
ProperName2 of 'a sandstone or a shale' is 'a Sandstone or a Shale'

**See also :** [String](#)

# DecrI, DecrL

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[DecrI](#) auto-decrement an integer value by 1.

[DecrL](#) auto-decrement a long value by 1.

## **Declare Syntax :**

Declare Sub cDecrI Lib "[time2win.dll](#)" (Value As Integer)

Declare Sub cDecrL Lib "[time2win.dll](#)" (Value As Long)

## **Call Syntax :**

Call cDecrI(Value%)

Call cDecrL(Value&)

## **Where :**

Value% is the integer value to auto-decrement.

Value& is the long value to auto-decrement.

## **Comments :**

These routines are slower than the VB equivalent : Value = Value - 1 but are shorter to type.

## **Examples :**

```
Dim Value As Integer
```

```
Value = 5
```

```
Call cDecrI(Value)           ' 4
```

```
Call cDecrL(Value)          ' 3
```

**See also :** [Miscellaneous](#)



# StringSAR

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**StringSAR** search and replace a string by an another in the specified string.

## Declare Syntax :

Declare Function cStringSAR Lib "time2win.dll" (ByVal Txt As String, ByVal Search As String, ByVal Replace As String, ByVal Sensitivity As Integer) As String

## Call Syntax :

Test\$ = cStringSAR(Txt\$, Search\$, Replace\$, Sensitivity%)

## Where :

Txt\$	the string to proceed.
Search\$	the string to be searched.
Replace\$	the replacement string.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
Test\$	the returned string with replacement.

## Comments :

If the search string is an EMPTY string, the returned string is the passed string.

If an error occurs when creating buffer, the returned string is the passed string.

The length of the replace string can be > or < of the search string.

The replace string can be an EMPTY string. In this case, the search string is removed from the string.

The length of the returned string can't be greater than 25 times the length of the Txt string.

## Examples :

```
Dim Txt           As String
Dim Search        As String
Dim Replace       As String
Dim Test         As String
```

```
Txt = "TIME TO WIN, TIME TO WIN IS A DLL"
```

```
Search = "TIME TO WIN"
```

```
Replace = "TIME2WIN"
```

```
Test = cStringSAR(Txt, Search, Replace, False)
```

```
Debug.Print Test           ' "TIME2WIN, TIME2WIN IS A DLL"
```

```
Search = "TIME to WIN"
```

```
Replace = "TIME2WIN"
```

```
Test = cStringSAR(Txt, Search, Replace, True)
```

```
Debug.Print Test           ' "TIME TO WIN, TIME TO WIN IS A DLL"
```

```
Search = " TO "
```

```
Replace = "2"
```

```
Test = cStringSAR(Txt, Search, Replace, True)
```

Debug.Print Test

' "TIME2WIN, TIME2WIN IS A DLL"

**See also :** [String](#)

## Miscellaneous : Overview

<a href="#">BaseConversion</a>	convert a number string (long integer) from a radix to another radix.
<a href="#">Between</a>	check to see if a value is between two other values.
<a href="#">Combination</a>	compute $C(n,m)$ which is the number of combinations of $n$ items, taken $m$ at a time.
<a href="#">Convert</a>	perform conversion between date, time, hour, minute, hundred.
<a href="#">DecrI</a>	auto-decrement an integer value by 1.
<a href="#">DecrL</a>	auto-decrement a long value by 1.
<a href="#">Fraction</a>	return a value into the form of a fraction.
<a href="#">IncrI</a>	auto-increment an integer value by 1.
<a href="#">IncrL</a>	auto-increment a long value by 1.
<a href="#">Max</a>	return the highest value of the two VARIANT value (INTEGER or LONG).
<a href="#">Min</a>	return the smallest value of the two VARIANT value (INTEGER or LONG).
<a href="#">SpellMoney</a>	spell money value with hundredth.
<a href="#">TrueBetween</a>	check to see if a value is fully between two other values.

# PatternMatch, PatternMatchS

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[PatternMatch](#) search if a given pattern can be found in a given string.

[PatternMatchS](#) (only VB 4.0 (32-Bit)) search if a given pattern can be found in a given string (case-sensitive or not).

## Declare Syntax :

Declare Function cPatternMatch Lib "time2win.dll" (ByVal Txt As String, ByVal Pattern As String) As Integer

Declare Function cPatternMatchS Lib "time2win.dll" (ByVal Txt As String, ByVal Pattern As String, ByVal Sensitivity As Integer) As Integer

## Call Syntax :

```
test% = cPatternMatch(Txt$, Pattern$)
```

```
test% = cPatternMatch(Txt$, Pattern$, Sensitivity%)
```

## Where :

Txt\$	the string to proceed
Pattern\$	the pattern to match
Sensitivity%	is TRUE for case sensitive search is FALSE for case insensitive search
test%	TRUE if the pattern match FALSE if the pattern not match

## Comments :

The char '?' is used to match a single char.

The char '\*' is used to match a block of char.

For PatternMatch : the matching of all chars (not '?', '\*') is case-sensitive.

For PatternMatchS : the matching of all chars (not '?', '\*') is depending of the Sensitivity parameter.

## Examples :

```
test% = cPatternMatch("Under the blue sky, the sun lights", "**")           ' is TRUE
test% = cPatternMatch("Under the blue sky, the sun lights", "**????*?")     ' is TRUE
test% = cPatternMatch("Under the blue sky, the sun lights", "**Under*")     ' is TRUE
test% = cPatternMatch("Under the blue sky, the sun lights", "**sky*")       ' is TRUE
test% = cPatternMatch("Under the blue sky, the sun lights", "**lights")     ' is TRUE
test% = cPatternMatch("Under the blue sky, the sun lights", "Under*")       ' is TRUE
test% = cPatternMatch("Under the blue sky, the sun lights", "??der*sky*ligh??") ' is TRUE
test% = cPatternMatch("Under the blue sky, the sun lights", "Under?the * s?? **") ' is TRUE

test% = cPatternMatch("Under the blue sky, the sun lights", "**under*")     ' is FALSE
test% = cPatternMatch("Under the blue sky, the sun lights", "Under*sun")    ' is FALSE
test% = cPatternMatch("Under the blue sky, the sun lights", "Under t??e*")  ' is FALSE
test% = cPatternMatch("Under the blue sky, the sun lights", "under?the * s?? **") ' is FALSE

test% = cPatternMatchS("Under the blue sky, the sun lights", "under?the * s?? **", False) ' is TRUE
```

**See also :** [String](#)

# PatternExtMatch, PatternExtMatchS

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**PatternExtMatch** search if a given pattern can be found in a given string.

**PatternExtMatchS** (only VB 4.0 (32-Bit)) search if a given pattern can be found in a given string (case-sensitive or not).

## Declare Syntax :

Declare Function cPatternExtMatch Lib "time2win.dll" (ByVal Txt As String, ByVal Pattern As String) As Integer

Declare Function cPatternExtMatchS Lib "time2win.dll" (ByVal Txt As String, ByVal Pattern As String, ByVal Sensitivity As Integer) As Integer

## Call Syntax :

test% = cPatternMatch(Txt\$, Pattern\$)

test% = cPatternMatch(Txt\$, Pattern\$, Sensitivity%)

## Where :

Txt\$                    the string to proceed

Pattern\$                the pattern to match

Sensitivity%            is TRUE for case sensitive search  
                          is FALSE for case insensitive search

test%                   TRUE if the pattern match,  
                          <> TRUE if the pattern not match or if an error has occurs

## Comments :

PatternExtMatch is a superset of PatternMatch and is a little bit faster.

The char '?' is used to match a single char.

The char '\*' is used to match a block of char.

The construct [x-y] is used to match a single char in range of chars (b.e. : [a-m], [n-z], [abcABC], [abgx-y]).

The construct [!x-y] or [^x-y] is used to match a single char not in range of chars (b.e. : [!A-Z], [^ - Z], [!abcABC], [^abgx-y]).

The hexa '~xy' is used to match a hexa char (b.e. : ~FF, ~A0, ~78, ~4, ~0A, ~0D).

For PatternExtMatch : the matching of all chars (not '?', '\*') is case-sensitive.

For PatternExtMatchS : the matching of all chars (not '?', '\*') is depending of the Sensitivity parameter.

If you want to suppress the special syntactic significance of any of `[]\*?!^-\~', and match the character exactly, precede it with a `\'.

The returned value can be the following :

MATCH_HEXA	match failure on hexa char &xy
MATCH_INTERNAL_ERROR	internal error
MATCH_PATTERN	bad pattern
MATCH_LITERAL	match failure on literal match
MATCH_RANGE	match failure on [...] construct
MATCH_ABORT	premature end of text string
MATCH_END	premature end of pattern string
MATCH_VALID	valid match
PATTERN_VALID	valid pattern
PATTERN_INVALID	invalid pattern
PATTERN_ESC	literal escape at end of pattern



' definition for error type for PATTERNMATCHEXT

```
Public Const MATCH_HEX = 17  
Public Const MATCH_INTERNAL_ERROR = 16  
Public Const MATCH_PATTERN = 15  
Public Const MATCH_LITERAL = 14  
Public Const MATCH_RANGE = 13  
Public Const MATCH_ABORT = 12  
Public Const MATCH_END = 11  
Public Const MATCH_VALID = -1
```

```
Public Const PATTERN_VALID = 0  
Public Const PATTERN_INVALID = 1  
Public Const PATTERN_ESC = 2  
Public Const PATTERN_RANGE = 3  
Public Const PATTERN_CLOSE = 4  
Public Const PATTERN_EMPTY = 5  
Public Const PATTERN_INTERNAL_ERROR = 6  
Public Const PATTERN_HEX = 7
```

```
' definition for error type for PROPERNAME2  
Public Const PN_UPPERCASE = 1  
Public Const PN_PUNCTUATION = 2  
Public Const PN_KEEP_ORIGINAL = 4  
Public Const PN_ONLY_LEADER_SPACE = 8
```



# CheckNumericity

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[CheckNumericity](#) check if a string is a numeric string.

## **Declare Syntax :**

Declare Function cCheckNumericity Lib "[time2win.dll](#)" (Txt As String) As Integer

## **Call Syntax :**

Test% = cCheckNumericity(Txt\$)

## **Where :**

Txt\$	is the specified string
Test%	TRUE : if the string is numeric FALSE : if the string is not numeric

## **Comments :**

## **Examples :**

```
Test% = cCheckNumericity("123456789")      ' TRUE
Test% = cCheckNumericity("A0B1")          ' FALSE
```

**See also :** [String](#)

# Morse

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Morse](#) convert a string to a morse string.

## Declare Syntax :

Declare Function cMorse Lib "[time2win.dll](#)" (ByVal morse As String) As String

## Call Syntax :

```
test$ = cMorse(morse$)
```

## Where :

morse\$                    is the string to proceed  
test\$                     is the returned string in morse

## Comments :

Only the following chars are valid :

```
space  
, - . / 0 1 2 3 4 5 6 7 8 9 ? A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

All other chars are filtered.

Each morse char is separated by a letter space (' ').  
Each block of char is separated by a word space('~').

These 2 chars (' ', '~') are not part of the morse coding. It will be used to facilitate the reading of the morse coding.

## Examples :

```
test$ = cMorse("SOS")                    ' " _ _ _ . . . _ _ _ "  
test$ = cMorse("TIME TO WIN")         ' ". - - . . - ~ . . . . ~ . - - . - "
```

**See also :** [String](#)

# Max, Min

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

**Max** return the highest value of the two VARIANT value (INTEGER or LONG).

**Min** return the smallest value of the two VARIANT value (INTEGER or LONG).

## **Declare Syntax :**

Declare Function cMax Lib "time2win.dll" (Var1 As Variant, Var2 As Variant) As Variant

Declare Function cMin Lib "time2win.dll" (Var1 As Variant, Var2 As Variant) As Variant

## **Call Syntax :**

```
test = cMax(Var1, Var2)
```

```
test = cMin(Var1, Var2)
```

## **Where :**

Var1            the first value.

Var2            the second value.

test            the highest/smallest value of the two.

## **Comments :**

## **Examples :**

```
test = cMax(1234, 4321)            ' 4321
```

```
test = cMin(1234, 4321)           ' 1234
```

**See Also :** [Miscellaneous](#)

# IncrI, IncrL

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[IncrI](#) auto-increment an integer value by 1.

[IncrL](#) auto-increment a long value by 1.

## **Declare Syntax :**

Declare Sub cIncrI Lib "[time2win.dll](#)" (Value As Integer)

Declare Sub cIncrL Lib "[time2win.dll](#)" (Value As Long)

## **Call Syntax :**

Call cIncrI(Value%)

Call cIncrL(Value&)

## **Where :**

Value% is the integer value to auto-increment.

Value& is the long value to auto-increment.

## **Comments :**

These routines are slower than the VB equivalent : Value = Value + 1 but are shorter to type.

## **Examples :**

```
Dim Value As Integer
```

```
Value = 5
```

```
Call cIncrI(Value) ' 6
```

```
Call cIncrI(Value) ' 7
```

**See also :** [Miscellaneous](#)

# Rnd, RndInit, RndD, RndI, RndL, RndS

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[RndInit](#) initialize the random generator.

[RndD](#) return a double random number.

[RndI](#) return an integer random number.

[RndL](#) return a long random number.

[RndS](#) return a single random number.

[Rnd](#) return a double random number between 0.0 and 1.0.

## Declare Syntax :

```
Declare Sub cRndInit Lib "time2win.dll" (ByVal nRnd As Long)
```

```
Declare Function cRndD Lib "time2win.dll" () As Double
```

```
Declare Function cRndI Lib "time2win.dll" () As Integer
```

```
Declare Function cRndL Lib "time2win.dll" () As Long
```

```
Declare Function cRndS Lib "time2win.dll" () As Single
```

```
Declare Function cRnd Lib "time2win.dll" () As Double
```

## Call Syntax :

```
Call cRndInit(nRnd&)
```

```
Test% = cRndI()
```

```
Test& = cRndL()
```

```
Test! = cRndS()
```

```
Test# = cRndD()
```

```
Test# = cRnd()
```

## Where :

nRnd < 0 : initialization with the current date and time.  
> 0 : initialization with the passed value.

Test? the returned random number.

## Comments :

## Examples :

```
Call cRndInit(-1)
```

```
Debug.Print cRndI()           ' 316
Debug.Print cRndL()           ' 45980750
Debug.Print cRndS()           ' 1,330308E+38
Debug.Print cRndD()           ' 1,87044922807943E+304
Debug.Print cRnd()            ' 1,87044922807943E+304
```

**See Also :** [Miscellaneous](#)

# SpellMoney

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[SpellMoney](#) spell money value with hundredth.

## **Declare Syntax :**

Declare Function cSpellMoney Lib "[time2win.dll](#)" (ByVal Value As Double, ByVal Units As String, ByVal Cents As String) As String

## **Call Syntax :**

Test\$ = cSpellMoney(Value#, Units\$, Cents\$)

## **Where :**

Value#	is the money value to spell.
Units\$	is the text string for units part.
Cents\$	is the text string for cents part.
Test\$	is the returned spelled money value.

## **Comments :**

## **Examples :**

Test\$ = cSpellMoney("98765.43", "dollars", "cents")

SpellMoney of '4.12' is 'Four dollars and Twelve cents'

SpellMoney of '16' is 'Sixteen dollars'

SpellMoney of '25' is 'Twenty-Five dollars'

SpellMoney of '34' is 'Thirty-Four dollars'

SpellMoney of '43' is 'Forty-Three dollars'

SpellMoney of '61' is 'Sixty-One dollars'

SpellMoney of '98765.43' is 'Ninety-Eight Thousand Seven Hundred Sixty-Five dollars and Forty-Three cents'

SpellMoney of '123456789.75' is 'One Hundred Twenty-Three Million Four Hundred Fifty-Six Thousand Seven Hundred Eighty-Nine dollars and Seventy-Five cents'

**See also :** [Miscellaneous](#)

# Fraction

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Fraction](#) return a value into the form of a fraction.

## Declare Syntax :

Declare Function cFraction Lib "[time2win.dll](#)" (ByVal nValue As Double, nNumerator As Double, nDenominator As Double) As Double

## Call Syntax :

Test# = cFraction(Value#, Numerator#, Denominator#)

## Where :

Value#	is the value to proceed.
Numerator#	is the returned numerator.
Denominator#	is the returned denominator.
Test#	is the returned value (Numerator# / Denominator#).

## Comments :

## Examples :

```
Dim Value           As Double
Dim Numerator       As Double
Dim Denominator     As Double
Dim CalculatedValue As Double
```

```
Value = 0.75
CalculatedValue = cFraction(Value, Numerator, Denominator)
```

```
' Numerator = 3
' Denominator = 4
' CalculatedValue = 0.75
```

```
Value = 3.14159265
CalculatedValue = cFraction(Value, Numerator, Denominator)
```

```
' Numerator = 3017882801
' Denominator = 960621932
' CalculatedValue = 3,14159265
```

**See also :** [Miscellaneous](#)

# Between, TrueBetween

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Between](#) check to see if a value is between two other values.

[TrueBetween](#) check to see if a value is fully between two other values.

## Declare Syntax :

Declare Function cBetween Lib "[time2win.dll](#)" (Var As Variant, Var1 As Variant, Var2 As Variant) As Integer

Declare Function cTrueBetween Lib "[time2win.dll](#)" (Var As Variant, Var1 As Variant, Var2 As Variant) As Integer

## Call Syntax :

```
test = cBetween(var, var1, var2)
```

## Where :

var	value to test
var1	first value
var2	second value
test	TRUE if var is between/fully between var1 and var2 FALSE if var is not between/fully between var1 and var2

## Comments :

var, var1, var2 are Variant value.

In this routine, only Integer, Long, Single, Double are supported.

## Examples :

```
var = 5
var1 = 1
var2 = 10
test = cBetween(var, var1, var2)           ' test = TRUE
test = cTrueBetween(var, var1, var2)      ' test = TRUE

var = 10
test = cBetween(var, var1, var2)          ' test = TRUE
test = cTrueBetween(var, var1, var2)     ' test = FALSE
```

**See Also :** [Miscellaneous](#)



# Type : Overview

[TypesCompare](#)

compare two Type'd variable.

[CompareTypeString](#)

compare a Type'd to a String.

[CompareStringType](#)

compare a String to a Type'd.

[TypeClear](#)

clear a Type'd variable.

[TypeMid](#)

extract information from a Type'd variable.

[TypesCopy](#)

copy a Type'd variable into a variable.

[TypeTransfert](#)

transfer a Type'd variable into a String.

[StringToType](#)

copy a String to a Type'd variable.

[TypeToString](#)

copy a Type'd variable to a String.

# Type.X

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[TypesCompare](#) compares two Type'd variable.

[CompareTypeString](#) compares a Type'd to a String.

[CompareStringType](#) compares a String to a Type'd.

[TypeClear](#) clears a Type'd variable.

[TypeMid](#) extracts information from a Type'd variable.

[TypesCopy](#) copies a Type'd variable into a variable.

[TypeTransfert](#) transfers a Type'd variable into a String.

[StringToType](#) copies a String to a Type'd variable.

[TypeToString](#) copies a Type'd variable to a String.

## Declare Syntax :

```
Declare Function cTypesCompare Lib "time2win.dll" (Type1 As Any, Type2 As Any, ByVal lenType1 As Integer) As Integer
```

```
Declare Function cCompareTypeString Lib "time2win.dll" Alias "cTypesCompare" (TypeSrc As Any, ByVal Dst As String, ByVal lenTypeSrc As Integer) As Integer
```

```
Declare Function cCompareStringType Lib "time2win.dll" Alias "cTypesCompare" (ByVal Src As String, TypeDst As Any, ByVal lenTypeSrc As Integer) As Integer
```

```
Declare Sub cTypeClear Lib "time2win.dll" (TypeSrc As Any, ByVal lenTypeSrc As Integer)
```

```
Declare Function cTypeMid Lib "time2win.dll" (TypeSrc As Any, ByVal Offset As Integer, ByVal Length As Integer) As String
```

```
Declare Sub cTypesCopy Lib "time2win.dll" (TypeSrc As Any, TypeDst As Any, ByVal lenTypeSrc As Integer)
```

```
Declare Function cTypeTransfert Lib "time2win.dll" (TypeSrc As Any, ByVal lenTypeSrc As Integer) As String
```

```
Declare Sub cStringToType Lib "time2win.dll" Alias "cTypesCopy" (ByVal Src As String, TypeDst As Any, ByVal lenTypeSrc As Integer)
```

```
Declare Sub cTypeToString Lib "time2win.dll" Alias "cTypesCopy" (TypeSrc As Any, ByVal Dst As String, ByVal lenTypeSrc As Integer)
```

## Call Syntax :

```
test% = cTypesCompare(Type1, Type2, len(Type1))
```

```
test% = cCompareTypeString(TypeSrc, Dst, len(TypeSrc))
```

```
test% = cCompareStringType(Src, TypeDst, len(TypeDst))
```

```
Call cTypeClear(TypeSrc, len(TypeSrc))
```

```
test$ = cTypeMid(TypeSrc, Offset, Length)
```

```
Call cTypesCopy(TypeSrc, TypeDst, len(TypeSrc))
```

```
test$ = cTypeTransfert(TypeSrc, len(TypeSrc))
```

```
Call cStringToType(Src, TypeDst, len(TypeDst))
```

```
Call cTypeToString(TypeSrc, Dst, len(TypeSrc))
```

## Where :

Type1, Type2, TypeSrc, TypeDst

the Type'd variable

Src, Dst,

the String variable

Offset

the offset in the Type'd variable

Length

the length in the Type'd variable

test%	TRUE if the variables to compare are the same
test\$	FALSE if the variables to compare are not the same the result

**Comments :**

Only Type'd variable mixed with INTEGER, LONG, SINGLE, DOUBLE, CURRENCY and FIXED STRING can be used.

When you compare 2 Type'd variables or 1 Type'd variable and 1 string, the size of each variable must be same.  
When you copy 1 Type'd variable into a string or a string into Type'd variable, the size of each variable must be same.

**Examples :**

**See also :** [Type](#)

# BaseConversion

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[BaseConversion](#) convert a number string (long integer) from a radix to another radix.

## Declare Syntax :

Declare Function cBaseConversion Lib "[time2win.dll](#)" (ByVal Num As String, ByVal RadixIn As Integer, ByVal RadixOut As Integer) As String

## Call Syntax :

```
test$ = cBaseConversion(Num$, RadixIn%, RadixOut%)
```

## Where :

Num\$	is the number string to convert
RadixIn%	is the base of the radix
RadixOut%	is the new base of the radix
test\$	is the result

## Comments :

If the number string can be converted, the returned string is an EMPTY string.

## Examples :

```
Convert '1234567' base 10 to base 2 is 100101101011010000111
Convert '1234567' base 10 to base 3 is 2022201111201
Convert '1234567' base 10 to base 4 is 10231122013
Convert '1234567' base 10 to base 5 is 304001232
Convert '1234567' base 10 to base 6 is 42243331
Convert '1234567' base 10 to base 7 is 13331215
Convert '1234567' base 10 to base 8 is 4553207
Convert '1234567' base 10 to base 9 is 2281451
Convert '1234567' base 10 to base 10 is 1234567
Convert '1234567' base 10 to base 11 is 773604
Convert '1234567' base 10 to base 12 is 4b6547
Convert '1234567' base 10 to base 13 is 342c19
Convert '1234567' base 10 to base 14 is 241cb5
Convert '1234567' base 10 to base 15 is 195be7
Convert '1234567' base 10 to base 16 is 12d687
Convert '1234567' base 10 to base 17 is ed4ea
Convert '1234567' base 10 to base 18 is bdc71
Convert '1234567' base 10 to base 19 is 98ig4
Convert '1234567' base 10 to base 20 is 7e687
```

**See also :** [Miscellaneous](#)

# DBFileCopy, PBFileCopy

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**PBFileCopy** copy a file to an another file and display a progress bar a client standard control.

**DBFileCopy** copy a file to an another file and display a dialog box with title, captions, progress bar and cancel button

## Declare Syntax :

Declare Function cPBFileCopy Lib "time2win.dll" (ByVal hWndParent As Long, ByVal FileNameIn As String, ByVal FileNameOut As String) As Integer

Declare Function cDBFileCopy Lib "time2win.dll" (ByVal Title As String, ByVal CaptionFrom As String, ByVal CaptionTo As String, ByVal CaptionButton As String, ByVal FileNameIn As String, ByVal FileNameOut As String) As Integer

## Call Syntax :

intResult% = cPBFileCopy(hWndParent&, FileNameIn\$, FileNameOut\$)

intResult% = cDBFileCopy(Title\$, CaptionFrom\$, CaptionTo\$, CaptionButton\$, FileNameIn\$, FileNameOut\$)

## Where :

hWndParent& is the .hWnd of the standard control or of the form.  
FileNameIn\$ is the file to be copied.  
FileNameOut\$ is the file copied.

Title\$ is the title of the dialog box.  
CaptionFrom\$ is the caption for the file to be copied.  
CaptionTo\$ is the caption for the file copied.  
CaptionButton\$ is the caption for the 'cancel' button.

intResult% = TRUE : no error  
= FALSE : an error has occurred

## Comments :

## Examples :

For cPBFileCopy :

```
Dim intResult As Long
Dim strResult As String
Dim strDisplay As String
```

```
Dim i As Long
```

```
Dim File1 As String
Dim File2 As String
```

```
strResult = ""
strDisplay = ""
```

```
File1 = cGetWindowsDirectory() + "\" + "system.dat"
File2 = "system.pbcopy"
```

```
strDisplay = strDisplay & "PB File Copy " & File1 & " to " & File2 & " is " & cPBFileCopy(Me.hWnd, File1, File2) & vbCrLf & vbCrLf
```

```
Debug.Print strDisplay
```

For cDBFileCopy :

```
Dim intResult      As Long  
Dim strResult      As String  
Dim strDisplay As String
```

```
Dim i              As Long
```

```
Dim File1          As String  
Dim File2          As String
```

```
strResult = ""  
strDisplay = ""
```

```
File1 = cGetWindowsDirectory() + "\system.dat"  
File2 = "system.dbcopy"
```

```
strDisplay = strDisplay & "DB File Copy " & File1 & " to " & File2 & " is " & cDBFileCopy("", "", "", "", File1, File2) &  
vbCrLf & vbCrLf
```

```
File1 = cGetWindowsDirectory() + "\command.com"  
File2 = "command.dbcopy"
```

```
strDisplay = strDisplay & "DB File Copy " & File1 & " to " & File2 & " is " & cDBFileCopy("", "", "", "", File1, File2) &  
vbCrLf & vbCrLf
```

```
Debug.Print strDisplay
```

**See also :** [Windows 95](#)

# Combination

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Combination](#) compute  $C(n,m)$  which is the number of combinations of  $n$  items, taken  $m$  at a time.

## Declare Syntax :

Declare Function cCombination Lib "[time2win.dll](#)" (ByVal nItems As Integer, ByVal mTimes As Integer) As Double

## Call Syntax :

Test# = cCombination(nItems%, mTimes%)

## Where :

nItems	the number of items.
mTimes%	the number taken.
Test#	the result.

## Comments :

If nItems is below 0 or if mTimes is not between 0 and nItems, the result is -1.  
Beware of using to big nItems and/or mTimes, this gives an overflow.

## Examples :

Debug.Print cCombination(42, 0)	' 1
Debug.Print cCombination(42, 1)	' 42
Debug.Print cCombination(42, 2)	' 861
Debug.Print cCombination(42, 42)	' 1
Debug.Print cCombination(42, 41)	' 42
Debug.Print cCombination(42, 40)	' 861

**See also** : [Miscellaneous](#)

# Windows 95/NT 4.0x : Overview

<a href="#">DBFileCopy</a>	copies a file to an another file and display a dialog box with title, captions, progress bar and cancel button.
<a href="#">ExplorerAddToRecentDocs</a>	adds a document to the shell's list of recently used documents.
<a href="#">ExplorerClearAllRecentDocs</a>	clears all documents from the shell's list of recently used documents.
<a href="#">GetFileDisplayName</a>	retrieves the EXPLORER (95/NT4.0x) display name of a file.
<a href="#">GetFileType</a>	retrieves the EXPLORER (95/NT4.0x) type name of a file.
<a href="#">GetOSBuildNumber</a>	identifies the build number of the operating system.
<a href="#">GetOSCSVersion</a>	provides arbitrary additional information about the operating system.
<a href="#">GetOSMajorVersion</a>	identifies the major version number of the operating system.
<a href="#">GetOSMinorVersion</a>	identifies the minor version number of the operating system.
<a href="#">GetOSVersion</a>	identifies the version number of the operating system (major.minor version).
<a href="#">GetOSPlatformId</a>	identifies the operating system platform in a numerical format.
<a href="#">GetOSPlatformName</a>	identifies the operating system platform in a readable format (string).
<a href="#">GetShortPathName</a>	retrieves the short path form of a specified input path.
<a href="#">IsCapsLockOn</a>	verifies if the CAPS LOCK key is On.
<a href="#">IsInsertOn</a>	verifies if the INSERT key is On.
<a href="#">IsNumLockOn</a>	verifies if the NUM LOCK key is On.
<a href="#">IsScrollLockOn</a>	verifies if the SCROLL LOCK key is On.
<a href="#">LockKeyboard</a>	locks/unlocks the keyboard for any applications.
<a href="#">LockMouse</a>	locks/unlocks the mouse (any events : left, middle and right click) for any applications.
<a href="#">MemoryStatus</a>	retrieves the actual state of the memory.
<a href="#">MultitasksKeys</a>	disables/enables CTRL+ALT+DEL and ALT+TAB.
<a href="#">PBFileCopy</a>	copies a file to an another file and display a progress bar a client standard control.
<a href="#">SetCapsLock</a>	sets the CAPS LOCK key On or Off.
<a href="#">SetInsert</a>	sets the INSERT key On or Off.
<a href="#">SetNumLock</a>	sets the NUM LOCK key On or Off.
<a href="#">SetScrollLock</a>	sets the SCROLL LOCK key On or Off.
<a href="#">ShortcutCreate</a>	creates a shortcut from a specified file.
<a href="#">ShortcutFileGetArguments</a>	retrieves the argument list from a shortcut file (*.lnk).
<a href="#">ShortcutFileGetDescription</a>	retrieves the description from a shortcut file (*.lnk).
<a href="#">ShortcutFileGetIconLocation</a>	retrieves the icon location from a shortcut file (*.lnk).
<a href="#">ShortcutFileGetPath</a>	retrieves the path from a shortcut file (*.lnk).
<a href="#">ShortcutFileGetPath83</a>	retrieves the path (in format 8.3) from a shortcut file (*.lnk).
<a href="#">ShortcutFileGetWorkingDir</a>	retrieves the working directory from a shortcut file (*.lnk).
<a href="#">ShortcutFileGetInfo</a>	is a generic function to retrieve the above parameter from a shortcut file (*.lnk).
<a href="#">ShortcutFileSetArguments</a>	modifies the argument list in a shortcut file (*.lnk).
<a href="#">ShortcutFileSetDescription</a>	modifies the description in a shortcut file (*.lnk).
<a href="#">ShortcutFileSetIconLocation</a>	modifies the icon location in a shortcut file (*.lnk).
<a href="#">ShortcutFileSetPath</a>	modifies the path in a shortcut file (*.lnk).
<a href="#">ShortcutFileSetWorkingDir</a>	modifies the working directory in a shortcut file (*.lnk).
<a href="#">ShortcutFileSetInfo</a>	is a generic function to modify the above parameter in a shortcut file (*.lnk).
<a href="#">TaskBarAddIcon</a>	adds an icon for an application in the tray of the task bar.
<a href="#">TaskBarDeleteIcon</a>	deletes the tray icon from an application in the task bar.
<a href="#">TaskBarModifyIcon</a>	modifies an icon for an application in the tray of the task bar.
<a href="#">TrashFile</a>	sends a file to the trash with persistence and confirmation.
<a href="#">TrayBar</a>	hides/shows the tray bar.





```
Debug.Print cGetRegistry("software\The MCR Company\TIME TO WIN for VB 4.0", "", "?1")           ' Code
name
Debug.Print cGetRegistry("software\The MCR Company\TIME TO WIN for VB 4.0", "RegName", "?2")      ' James
Debug.Print cGetRegistry("software\The MCR Company\TIME TO WIN for VB 4.0", "RegId", "?3")       ' Donb
Debug.Print cGetRegistry("software\The MCR Company\TIME TO WIN for VB 4.0", "RegN°", "?4")       ' 007

Debug.Print cKillRegistry("under the fox", "")                                               ' -1
Debug.Print cKillRegistry("software\The MCR Company", "")                                   ' -1
```

**See also :** [Registry key](#)

' structure for windows 95 memory

Type tagMEMORYSTATUS

dwLength	As Long	' sizeof(MEMORYSTATUS)
dwMemoryLoad	As Long	' percent of memory in use
dwTotalPhys	As Long	' bytes of physical memory
dwAvailPhys	As Long	' free physical memory bytes
dwTotalPageFile	As Long	' bytes of paging file
dwAvailPageFile	As Long	' free bytes of paging file
dwTotalVirtual	As Long	' user bytes of address space
dwAvailVirtual	As Long	' free user bytes

End Type

' configuration code

```
Public Const RK_HKEY_CLASSES_ROOT = 1
Public Const RK_HKEY_CURRENT_USER = 2
Public Const RK_HKEY_LOCAL_MACHINE = 3
Public Const RK_HKEY_USERS = 4
Public Const RK_HKEY_PERFORMANCE_DATA = 5
Public Const RK_HKEY_CURRENT_CONFIG = 6
Public Const RK_HKEY_DYN_DATA = 7
```

' error/success code

```
Public Const RK_NO_ERROR = -1
Public Const RK_KEY_IS_EMPTY = 1
Public Const RK_UNABLE_TO_CREATE_KEY = 2
Public Const RK_UNABLE_TO_OPEN_KEY = 3
Public Const RK_UNKNOWN_DISPOSITION = 4
Public Const RK_CANNOT_SET_THE_VALUE = 5
Public Const RK_UNABLE_TO_QUERY_KEY = 6
```

# MemoryStatus

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[MemoryStatus](#) retrieve the actual state of the memory.

## Declare Syntax :

Declare Sub cMemoryStatus Lib "time2win.dll" (MEMORYSTATUS As tagMEMORYSTATUS)

## Call Syntax :

Call cMemoryStatus(MEMORYSTATUS)

## Where :

MEMORYSTATUS is the type'd variable to receive the actual memory status.

## Comments :

MEMORYSTATUS.dwMemoryLoad :

Specifies a number between 0 and 100 that gives a general idea of current memory utilization, in which 0 indicates no memory use and 100 indicates full memory use.

MEMORYSTATUS.dwTotalPhys :

Indicates the total number of bytes of physical memory.

MEMORYSTATUS.dwAvailPhys :

Indicates the number of bytes of physical memory available.

MEMORYSTATUS.dwTotalPageFile :

Indicates the total number of bytes that can be stored in the paging file. Note that this number does not represent the actual physical size of the paging file on disk.

MEMORYSTATUS.dwAvailPageFile :

Indicates the number of bytes available in the paging file.

MEMORYSTATUS.dwTotalVirtual :

Indicates the total number of bytes that can be described in the user mode portion of the virtual address space of the calling process.

MEMORYSTATUS.dwAvailVirtual :

Indicates the number of bytes of unreserved and uncommitted memory in the user mode portion of the virtual address space of the calling process.

## Examples :

Dim strDisplay As String

Dim MSS                      As tagMEMORYSTATUS

strDisplay = ""

Call cMemoryStatus(MSS)

```
strDisplay = strDisplay & "dwMemoryLoad = " & MSS.dwMemoryLoad & vbCrLf
strDisplay = strDisplay & "dwTotalPhys = " & MSS.dwTotalPhys & vbCrLf
strDisplay = strDisplay & "dwAvailPhys = " & MSS.dwAvailPhys & vbCrLf
strDisplay = strDisplay & "dwTotalPageFile = " & MSS.dwTotalPageFile & vbCrLf
strDisplay = strDisplay & "dwAvailPageFile = " & MSS.dwAvailPageFile & vbCrLf
strDisplay = strDisplay & "dwTotalVirtual = " & MSS.dwTotalVirtual & vbCrLf
strDisplay = strDisplay & "dwAvailVirtual = " & MSS.dwAvailVirtual & vbCrLf
```

Debug.Print strDisplay

**See also :** [Windows 95](#)

## Swap : Overview

[SwapD](#)

swap two Double values.

[SwapI](#)

swap two Integer values.

[SwapL](#)

swap two Long values.

[SwapS](#)

swap two Single values.

[SwapStr](#)

swap two strings.

# Matrix : Overview

[MatrixAdd](#)

add two square matrix.

[MatrixCoFactor](#)

calculate the CoFactor of an element in a square matrix.

[MatrixCompare](#)

compare two square matrix.

[MatrixCopy](#)

copy a square matrix.

[MatrixDet](#)

calculate the Determinant of a square matrix.

[MatrixFill](#)

fill a square matrix (matrix zero, matrix unit).

[MatrixInv](#)

invert a square matrix (determinant can't be nul).

[MatrixMinor](#)

calculate the Minor of an element in a square matrix.

[MatrixMul](#)

multiply two square matrix.

[MatrixSub](#)

substract two square matrix.

[MatrixSymToeplitz](#)

create a symmetrical Toeplitz matrix.

[MatrixTranspose](#)

transpose a square matrix.



## Random : Overview

<a href="#">Rnd</a>	return a double random number between 0.0 and 1.0.
<a href="#">RndD</a>	return a double random number.
<a href="#">RndI</a>	return an integer random number.
<a href="#">RndInit</a>	initialize the random generator.
<a href="#">RndL</a>	return a long random number.
<a href="#">RndS</a>	return a single random number.

# Matrix

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**MatrixAdd** add two square matrix.

**MatrixCoFactor** calculate the CoFactor of an element in a square matrix.

**MatrixCompare** compare two square matrix.

**MatrixCopy** copy a square matrix.

**MatrixDet** calculate the Determinant of a square matrix.

**MatrixFill** fill a square matrix (matrix zero, matrix unit).

**MatrixInv** invert a square matrix (determinant can't be nul).

**MatrixMinor** calculate the Minor of an element in a square matrix.

**MatrixMul** multiply two square matrix.

**MatrixSub** subtract two square matrix.

**MatrixSymToeplitz** create a symmetrical Toeplitz matrix.

**MatrixTranspose** transpose a square matrix.

## Declare Syntax :

```
Declare Sub cMatrixAdd Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
```

```
Declare Function cMatrixCoFactor Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal Row As Integer, ByVal Col As Integer) As Double
```

```
Declare Function cMatrixCompare Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
```

```
Declare Sub cMatrixCopy Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double)
```

```
Declare Function cMatrixDet Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double) As Double
```

```
Declare Function cMatrixFill Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal nInit As Integer) As Integer
```

```
Declare Function cMatrixInv Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
```

```
Declare Function cMatrixMinor Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal Row As Integer, ByVal Col As Integer) As Double
```

```
Declare Sub cMatrixMul Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
```

```
Declare Sub cMatrixSub Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
```

```
Declare Function cMatrixSymToeplitz Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
```

```
Declare Sub cMatrixTranspose Lib "time2win.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double)
```

## Call Syntax :

```
Call cMatrixAdd(Size%, ArrayA(), ArrayB(), ArrayC())
```

```
Test# = cMatrixCoFactor(Size%, ArrayA(), Row, Col)
```

```
Test% = cMatrixCompare(Size%, ArrayA(), ArrayC())
```

```
Call cMatrixCopy(Size%, ArrayA(), ArrayC())
```

```
Test# = cMatrixDet(Size%, ArrayA())
```

```
Test% = cMatrixFill(Size%, ArrayA(), nInit%)
```

```
Test% = cMatrixInv(Size%, ArrayA(), ArrayC())
```

```
Test# = cMatrixMinor(Size%, ArrayA(), Row, Col)
```

```
Call cMatrixMul(Size%, ArrayA(), ArrayB(), ArrayC())
```

```
Call cMatrixSub(Size%, ArrayA(), ArrayB(), ArrayC())
```

```
Test% = cMatrixSymToeplitz(Size%, ArrayA(), ArrayC())
```

```
Call cMatrixTranspose(Size%, ArrayA(), ArrayB(), ArrayC())
```

## Where :

Size% is the size for the matrixes.

ArrayA() is the first square matrix (only double value).  
ArrayB() is the second square matrix (only double value).  
ArrayC() is the result square matrix (only double value).  
nInit% MATRIX\_ZERO or MATRIX\_UNIT.  
Test% = True, matrixes are the same,  
= False, matrixes are not the same.

**Comments :**

These matrixes functions doesn't check if the matrix is really square and if the size is ok.  
All matrixes must be the same square (N x N).

**Examples :**

See the demo file.

**See also :**

# File : Overview

<a href="#">AllSubDirectories</a>	retrieve all sub-directories from a specified directory (root or sub-directory).
<a href="#">ChDir</a>	change the directory.
<a href="#">ChDrive</a>	change the drive.
<a href="#">CmpFileAttribute</a>	compare the attribute of two files.
<a href="#">CmpFileContents</a>	compare the contents of two files.
<a href="#">CmpFileSize</a>	compare the size of two files.
<a href="#">CmpFileTime</a>	compare the date and time of two files.
<a href="#">CountDirectories</a>	count the total directory in a specified directory.
<a href="#">CountFiles</a>	count the total files founded in a specified directory.
<a href="#">CutFile</a>	cut a file in two parts (creation of two files).
<a href="#">EnumOpenFiles</a>	enumerate all open files and/or all unmovable open files.
<a href="#">FileChangeChars</a>	replace all chars in a char set by a new char set.
<a href="#">FileCompressTab</a>	compress a number of spaces specified into a TAB char (horizontal tab).
<a href="#">FileExpandTab</a>	expand a TAB char (horizontal tab) into a number of spaces.
<a href="#">FileCopy</a>	copy one file to an another file.
<a href="#">FileCopy2</a>	copy one file to an another file.
<a href="#">FileDateCreated</a>	retrieve the date when the file has been created.
<a href="#">FileForceCopy</a>	copy one file to an another file with the same file attribute.
<a href="#">FileLastDateAccess</a>	retrieve the date when the file has been last accessed.
<a href="#">FileLastDateModified</a>	retrieve the date when the file has been last modified.
<a href="#">FileTimeCreated</a>	retrieve the time when the file has been created.
<a href="#">FileLastTimeAccess</a>	retrieve the time when the file has been last accessed.
<a href="#">FileLastTimeModified</a>	retrieve the time when the file has been last modified.
<a href="#">FileDrive</a>	extract the drive on which the file is present.
<a href="#">FileFilter</a>	copy one file to an another file but filters some chars.
<a href="#">FileFilterNot</a>	copy one file to an another file but filters chars not present in the filter.
<a href="#">FileGetAttrib</a>	retrieves in one call, all attributes of a file.
<a href="#">FileLineCount</a>	count the total number of lines in an ASCII file.
<a href="#">FileMerge</a>	merge two files in one.
<a href="#">FileMergeExt</a>	merge all files in an array in one.
<a href="#">FileMove</a>	move/rename a file in the same or in an another directory.
<a href="#">FilePartAppend</a>	appends part of one file to another file starting at or until Offset.
<a href="#">FilePartCopy</a>	copies part of one file to another file starting at or until Offset.
<a href="#">FilePathExists</a>	verify if the specified file is present.
<a href="#">FileResetAllAttrib</a>	reset all attributes of a file.
<a href="#">FileResetArchive</a>	reset the archive attribute of a file.
<a href="#">FileResetHidden</a>	reset the hidden attribute of a file.
<a href="#">FileResetReadOnly</a>	reset the read-only attribute of a file.
<a href="#">FileResetSystem</a>	reset the system attribute of a file.
<a href="#">FileResetFlag</a>	reset the specified attributes of a file.
<a href="#">FileScanHeader</a>	searchs a string in a given HEADER file starting at a certain line number.
<a href="#">FileScanHeaderForRecipients</a>	searchs a string (recipients like "to: ", "cc: ", "bcc: ") in a given HEADER file starting at a certain line number.
<a href="#">FileSearch</a>	search a string in a given TEXT file.
<a href="#">FileSearchCount</a>	count the occurrence of a string in a given TEXT file.
<a href="#">FileSearchAndReplace</a>	search and replace a string by an another in the specified TEXT file.
<a href="#">FileSearchFromLine</a>	searchs a string in a given TEXT file starting at a certain line number.
<a href="#">FileSearchPatternFromLine</a>	searchs a pattern string in a given TEXT file starting at a certain line number.
<a href="#">FileSetAllAttrib</a>	set all attributes of a file.
<a href="#">FileSetArchive</a>	set the archive attribute of a file.
<a href="#">FileSetHidden</a>	set the hidden attribute of a file.
<a href="#">FileSetReadOnly</a>	set the read-only attribute of a file.
<a href="#">FileSetSystem</a>	set the system attribute of a file.
<a href="#">FileSetFlag</a>	set the specified attributes of a file.
<a href="#">FileSetAttrib</a>	set in one call, attributes of a given file.
<a href="#">FilesCopy</a>	copy files from one directory to an another directory.
<a href="#">FilesInDirectory</a>	retrieve each file in the specified directory.
<a href="#">FilesInDirOnDisk</a>	write all files from a specified directory into a file on disk.

<a href="#">FilesInDirToArray</a>	read all files from a specified directory into an array.
<a href="#">FilesInfolnDir</a>	retrieve each file in the specified directory and returns name, size, Int date, Int time, attribute.
<a href="#">FilesMove</a>	move files from one directory to an another directory/disk.
<a href="#">FileSize</a>	return the size of the specified file.
<a href="#">FileSort</a>	sort an ASCII file or a BINARY file in ascending or descending order with case sensitive or not.
<a href="#">FilesSize</a>	return the logical size of all files specified by file mask.
<a href="#">FilesSizeOnDisk</a>	return the physical size of all files specified by file mask.
<a href="#">FilesSlack</a>	return in one call, the slack from all files specified by file mask, the logical size and the physical size.
<a href="#">FileStatistics</a>	count the lines, words and chars in a specified file.
<a href="#">FileToLower</a>	convert a file to a file with lower case.
<a href="#">FileToUpper</a>	convert a file to a file with upper case.
<a href="#">FileWrapLine</a>	wraps a file in multiple lines with a maximum length by line.
<a href="#">FullPath</a>	convert a partial path stored in path to a fully qualified path.
<a href="#">GetDiskFree</a>	retrieve the free disk space of a disk (hard disk or floppy disk).
<a href="#">GetDiskSpace</a>	retrieve the size of a disk (hard disk or floppy disk).
<a href="#">GetDiskUsed</a>	retrieve the part used of a disk (hard disk or floppy disk).
<a href="#">GetDiskClusterSize</a>	retrieve the size of a cluster on a disk (hard disk or floppy disk).
<a href="#">GetDriveType</a>	determine whether a disk drive is removable, fixed, or remote.
<a href="#">GetFileDateTime</a>	retrieve in one routine all date & time informations (creation, last access, last write) for a file.
<a href="#">KillDir</a>	delete the specified empty directory.
<a href="#">KillDirs</a>	delete the specified directory and its associated directories.
<a href="#">KillDirFilesAll</a>	delete all files specified by a mask in the specified directory and its associated sub-dir.
<a href="#">KillFile</a>	delete the specified filename.
<a href="#">KillFileAll</a>	delete the specified filename with any attribute.
<a href="#">KillFiles</a>	delete all files specified by a file mask.
<a href="#">KillFilesAll</a>	delete all files specified by a file mask even if some files are READ-ONLY files.
<a href="#">MakeDir</a>	create the specified directory.
<a href="#">MakeMultipleDir</a>	create a multiple directory in one call.
<a href="#">MakePath</a>	create a single path, composed of a drive letter, directory path, filename, and filename extension.
<a href="#">RcsCountFileDir</a>	count the total directories or files in a specified directory (with recursivity or not).
<a href="#">RcsFilesSize</a>	return the logical size of files by file mask in a specified directory (with recursivity or not).
<a href="#">RcsFilesSizeOnDisk</a>	return the physical size of files by file mask in a specified directory (with recursivity or not).
<a href="#">RcsFilesSlack</a>	return, in one call, the slack from files, the logical size and the physical size (with recursivity or not).
<a href="#">RenameFile</a>	rename a file or moves a file from one path to an other path.
<a href="#">SearchDir</a>	perform a directory match starting with a specified path.
<a href="#">SearchFile</a>	perform a file match starting with a specified path.
<a href="#">SearchFileAttrib</a>	perform a file (with attribute) match starting with a specified path.
<a href="#">SetFileDateTime</a>	set in one routine all date & time informations (creation, last access, last write) for a file.
<a href="#">SplitFile</a>	split a file into target files based on a part size.
<a href="#">SplitPath</a>	break a full path into its four components.
<a href="#">SubDirectory</a>	retrieve all sub-directories from the specified mask.
<a href="#">TruncatePath</a>	truncate a long path with filename.
<a href="#">UniqueFileName</a>	create a unique filename by modifying a given template argument.

# AllSubDirectories

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[AllSubDirectories](#) retrieve all sub-directories from a specified directory (root or sub-directory).

## Declare Syntax :

Declare Function cAllSubDirectories Lib "time2win.dll" (ByVal lpBaseDirectory As String, nDir As Integer) As String

## Call Syntax :

```
test$ = AllSubDirectories(lpBaseDirectory, nDir)
```

## Where :

lpBaseDirectory\$	is the specified directory
nDir%	< 0 if an error has occurred, > 0 the number of directories founded
test\$	return the directories in one string. Each directory is separated by a CR.

## Comments :

Don't forget that this function can handle a maximum of 700 directories of 70 chars long each. The returned string is always automatically sorted in ascending order.

The returned value in 'nDir' can be negative and have the following value :

-32760	allocation error for memory buffer 1.
-32761	allocation error for memory buffer 2.

## Examples :

```
test = cAllSubDirectories("C:",nDir)
```

**See also :** [File](#)

# ChDir

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[ChDir](#) change the directory.

## **Declare Syntax :**

Declare Function cChDir Lib "[time2win.dll](#)" (ByVal lpDir As String) As Integer

## **Call Syntax :**

status = cChDir(lpDir)

## **Where :**

lpDir	is the new directory
status	TRUE is all is OK
	<> TRUE is an error occurs

## **Comments :**

This fonction is the same that ChDir but doesn't generate an VB Error if a problem occurs.

**See also :** [File](#)

# ChDrive

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[ChDrive](#) change the drive.

## **Declare Syntax :**

Declare Function cChDrive Lib "[time2win.dll](#)" (ByVal lpDrive As String) As Integer

## **Call Syntax :**

status = cChDrive(lpDrive)

## **Where :**

lpDrive	is the new drive
status	TRUE is all is OK
	<> TRUE is an error occurs

## **Comments :**

This fonction is the same that ChDrive but doesn't generate an Error if a problem occurs.

**See also :** [File](#)



# FileCompressTab, FileExpandTab

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileCompressTab](#) compress a number of spaces specified into a TAB char (horizontal tab).

[FileExpandTab](#) expand a TAB char (horizontal tab) into a number of spaces.

## Declare Syntax :

Declare Function cFileCompressTab Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal nTab As Integer) As Long

Declare Function cFileExpandTab Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal nTab As Integer) As Long

## Call Syntax :

```
test& = cFileCompressTab(file1, file2, nTab)
```

```
test& = cFileExpandTab(file1, file2, nTab)
```

## Where :

file1\$	is the source file.
file2\$	is the destination file.
nTab%	is the number of spaces corresponding to a TAB char (horizontal tab).
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

The number of spaces to compress/expand a TAB must be 2 minimum.

Beware of the fact, that if the original file you want to compress spaces contains embedded TAB char, the expanded file is bigger than the original file.

The returned value can be negative and have the following value :

-1	number of spaces is below 2.
-2	overflow error in the expanding buffer for FileExpandTab.
-32720	the number of chars in a block for writing differs from the number of chars for reading.
-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.
-32760	allocation error for memory buffer 1.
-32761	allocation error for memory buffer 2.

## Examples :

```
test& = cFileCompressTab("c:\autoexec.bat", "c:\autoexec.tb1", 3)
```

```
test& = cFileExpandTab("c:\autoexec.tb1", "c:\autoexec.tb2", 3)
```

**See also :** [File](#)

# FileCopy

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileCopy](#) copy one file to an another file.

## Declare Syntax :

Declare Function cFileCopy Lib "[time2win.dll](#)" (ByVal file1 As String, ByVal file2 As String) As Long

## Call Syntax :

```
test& = cFileCopy(file1, file2)
```

## Where :

file1\$	is the source file.
file2\$	is the destination file.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

The returned value can be negative and have the following value :

-32720	the number of chars in a block for writing differs from the number of chars for reading.
-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.
-32760	allocation error for memory buffer.

## Examples :

```
test& = cFileCopy("c:\autoexec.bat", "c:\autoexec.tab")
```

**See also :** [File](#)

# FileMove

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FileMove](#) move/rename a file in the same or in an another directory.

## **Declare Syntax :**

Declare Function cFileMove Lib "[time2win.dll](#)" (ByVal File1 As String, ByVal File2 As String) As Long

## **Call Syntax :**

```
test& = cFileMove(File1, File2)
```

## **Where :**

File1	is the source file
File2	is the destination file
test&	>= 0 : the length of the file < 0 : an error has occurred.

## **Comments :**

## **Examples :**

**See also :** [File](#)

# FileFilter, FileFilterNot

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileFilter](#) copy one file to an another file but filters some chars.

[FileFilterNot](#) copy one file to an another file but filters chars not present in the filter.

## Declare Syntax :

```
Declare Function cFileFilter Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, Filter As String) As Long
Declare Function cFileFilterNot Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, Filter As String) As Long
```

## Call Syntax :

```
test& = cFileFilter(file1, file2, filter)
test& = cFileFilterNot(file1, file2, filternot)
```

## Where :

file1\$	is the source file.
file2\$	is the destination file.
filter\$	is the filter to use to remove chars from the source file.
filternot\$	is the filter to use to remove chars not present in the filter from the source file.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

The returned value can be negative and have the following value :

-1	the filter is an EMPTY string.
-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.
-32760	allocation error for memory buffer 1.
-32761	allocation error for memory buffer 2.

## Examples :

```
test& = cFileFilter("c:\autoexec.bat", "c:\autoexec.tab",
"ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz")
test& = cFileFilterNot("c:\autoexec.bat", "c:\autoexec.tab",
"ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz")
```

**See also :** [File](#)

# FileSize

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FileSize](#) return the size of the specified file.

## **Declare Syntax :**

Declare Function cFileSize Lib "[time2win.dll](#)" (ByVal lpFilename As String) As Long

## **Call Syntax :**

```
test& = cFileSize(lpFilename)
```

## **Where :**

lpFilename	the file to proceed
test&	the size of the file

## **Comments :**

If the file is not present or if an error occurs when accessing the file, the return value is 0

**See also :** [File](#)

# FileLineCount

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileLineCount](#) count the total number of lines in an ASCII file.

## Declare Syntax :

Declare Function cFileLineCount Lib "[time2win.dll](#)" (ByVal lpFilename As String) As Long

## Call Syntax :

```
test& = cFileLineCount(lpFilename$)
```

## Where :

lpFilename\$	is the name of the file.
test&	is the total number of lines.

## Comments :

Each line (*any size*) is determined only if a CR is ending the line.

The returned value can be negative and have the following value :

- 1 error opening file (not exist, not a valid filename).
- 2 error reading file.
- 3 error when allocating memory buffer.

## Examples :

```
test& = cFileLineCount("c:\autoexec.bat")
```

On my system :

```
test& = 31
```

**See also :** [File](#), [FileLineCount2](#)

# FileToLower, FileToUpper

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileToLower](#) convert a file to a file with lower case.

[FileToUpper](#) convert a file to a file with upper case.

## Declare Syntax :

Declare Function cFileToLower Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String) As Long

Declare Function cFileToUpper Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String) As Long

## Call Syntax :

```
test& = cFileToLower(file1, file2)
```

```
test& = cFileToUpper(file1, file2)
```

## Where :

file1\$ is the source file.

file2\$ is the destination file.

test& > 0 if all is OK (the returned value is the total bytes copied),

< 0 if an error has occurred.

## Comments :

The returned value can be negative and have the following value :

-32720 the number of chars in a block for writing differs from the number of chars for reading.

-32730 reading error for file 1.

-32740 writing error for file 2.

-32750 opening error for file 1.

-32751 opening error for file 2.

-32760 allocation error for memory buffer 1.

-32761 allocation error for memory buffer 2.

## Examples :

```
test& = cFileToLower("c:\autoexec.bat","c:\autoexec.lwr")
```

```
test& = cFileToUpper("c:\autoexec.bat","c:\autoexec.upr")
```

**See also :** [File](#)

# FileMerge, FileMergeExt

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FileMerge](#) merge two files in one.

[FileMergeExt](#) merge all files in an array in one.

## Declare Syntax :

Declare Function cFileMerge Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal fileTo As String) As Long

Declare Function cFileMergeExt Lib "time2win.dll" (FileArray() As String, ByVal TargetFile As String) As Long

## Call Syntax :

```
test& = cFileMerge(file1, file2, fileTo)
```

```
test& = cFileMergeExt(FileArray(), TargetFile)
```

## Where :

file1\$	is the first file.
file2\$	is the second file.
FileArray\$()	is the file array.
fileTo\$, TargetFile\$	is the destination file.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred, (for FileMergeExt, see <a href="#">success/error</a> code).

## Comments :

For FileMerge, The returned value can be negative and have the following value :

-32720	the number of chars in a block for writing differs from the number of chars for reading file 1.
-32721	the number of chars in a block for writing differs from the number of chars for reading file 2.
-32730	reading error for file 1.
-32731	reading error for file 2.
-32740	writing error for file To.
-32750	opening error for file 1.
-32751	opening error for file 2.
-32752	opening error for file To.
-32760	allocation error for memory buffer.

## Examples :

```
Dim i As Long
ReDim FileArray(1 To 7) As String

For i = 1 To 7
    FileArray(i) = Space$(256)
Next i

Debug.Print cFilesInDirToArray("C:\*.*", A_ALL, FileArray())

For i = 1 To 7
    FileArray(i) = "c:\\" & FileArray(i)
Next i

test& = cFileMergeExt(FileArray(), "c:\mergeext.byt")

test& = cFileMerge("c:\autoexec.bat", "c:\config.sys", "c:\merge.byt")
```



**See also :** [File](#)

# FileSearchAndReplace

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FileSearchAndReplace](#) search and replace a string by an another in the specified TEXT file.

## Declare Syntax :

Declare Function cFileSearchAndReplace Lib "time2win.dll" (ByVal nFileName As String, ByVal Search As String, ByVal Replace As String, ByVal nFileTemp As String, ByVal Sensitivity As Integer) As Long

## Call Syntax :

test& = cFileSearchAndReplace(nFilename\$, Search\$, Replace\$, nFileTemp\$, Sensitivity%)

## Where :

nFilename\$	the ASCII file.
Search\$	the string to be searched.
Replace\$	the replacement string.
nFileTemp\$	a temporary file.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

cFileSearchAndReplace can handle lines with a maximum of 4096 chars.

If the nFilename string is an EMPTY string, the returned value is FALSE.

If the search string is an EMPTY string, the returned value is FALSE.

The length of the replace string can be > or < of the search string.

The replace string can be an EMPTY string. In this case, the search string is removed from the file.

If the nFileTemp is an EMPTY string, a default temporary file is used.

The returned value can be negative and have the following value :

-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.

## Examples :

```
test& = cFileCopy("c:\autoexec.bat","c:autoexec.tab")
```

```
test& = cFileSearchAndReplace("c:\autoexec.tab", "path", " PATH ", "", False)
```

**See also :** [File](#)

# FileSearch, FileSearchCount

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FileSearch](#) search a string in a given TEXT file.

[FileSearchCount](#) count the occurrence of a string in a given TEXT file.

## Declare Syntax :

Declare Function cFileSearch Lib "time2win.dll" (ByVal nFileName As String, ByVal Search As String, ByVal sensitivity As Integer) As Long

Declare Function cFileSearchCount Lib "time2win.dll" (ByVal nFileName As String, ByVal Search As String, ByVal sensitivity As Integer) As Long

## Call Syntax :

```
test& = cFileSearch(nFilename$, Search$, Sensitivity%)
```

```
test& = cFileSearchCount(nFilename$, Search$, Sensitivity%)
```

## Where :

nFilename\$	the ASCII file.
Search\$	the string to be searched.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

cFileSearch and cFileSearchCount can handle lines with a maximum of 2304 chars.

For cFileSearch, the returned value is TRUE if the string is found and FALSE if not.

For cFileSearchCount, the returned value is the number of occurrence of the specified string.

If the nFilename string is an EMPTY string, the returned value is FALSE.

If the search string is an EMPTY string, the returned value is FALSE.

The returned value can be negative and have the following value :

-32730 reading error for file 1.

-32750 opening error for file 1.

## Examples :

```
test1& = cFileSearch("c:\autoexec.bat", "rEm", False)
```

```
test2& = cFileSearchCount("c:\autoexec.bat", "ReM", False)
```

On my system :

```
test1& = 3
```

```
test2& = 3
```

**See also :** [File](#)

# FileSort

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**FileSort** sort an ASCII file or a BINARY file in ascending or descending order with case sensitive or not.

## Declare Syntax :

Declare Function cFileSort Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, ByVal SortMethod As Integer, ByVal RecordLength As Long, ByVal KeyOffset As Long, ByVal KeyLength As Long, rRecords As Integer) As Long

## Call Syntax :

Test% = cFileSort(FileIn\$, FileOut\$, SortMethod%, RecordLength&, KeyOffset&, KeyLength&, rRecords%)

## Where :

FileIn\$	the input file.
FileOut\$	the output file.
SortMethod%	a combination of <u>sorting</u> constants :
RecordLength&	-1 for an ASCII file, > 0 for a BINARY file.
KeyOffset&	-1 for an ASCII file, >= 0 for a BINARY file.
KeyLength&	-1 for an ASCII file, > 0 for a BINARY file.
rRecords	the number of records treated.
Test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

The returned value can be negative and have the following value :

- 1 file 1 is invalid (empty name).
- 2 file 2 is invalid (empty name).
- 3 KeyOffset must be specified (RecordLength is used).
- 4 KeyOffset must be >= 0 (RecordLength is used).
- 5 KeyLength must be > 0 (RecordLength is used).
- 6 (KeyOffset + KeyLength) must be <= to RecordLength.
- 7 filename 1 must be different of filename 2.
- 8 unable to open file 1.
- 9 unable to open file 2.
- 10 can't allocate memory buffer for no fixed length
- 11 can't allocate memory buffer for pointers.
- 12 can't read first record.
- 13 can't read a record.
- 14 too many records (about > 16384).
- 15 can't expand memory buffer for pointers.
- 16 can't write a record (disk full, disk failure, ...).

FileSort uses memory to perform the sort. You're limited to the memory available and a maximum of about 16384 records.

## Examples :

Dim rRec                      As Integer

```
Debug.Print cFileSort("c:\autoexec.bat", "c:\ae1.bat", SORT_ASCENDING + SORT_CASE_INSENSITIVE, -1, -1, -1, rRec)
```

**See also :** [File](#)

# FileChangeChars

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileChangeChars](#) replace all chars in a char set by a new char set.

## Declare Syntax :

Declare Function cFileChangeChars Lib "time2win.dll" (ByVal nFileName As String, CharSet As String, NewCharSet As String, ByVal nFileTemp As String) As Long

## Call Syntax :

test& = cFileChangeChars(nFilename\$, CharSet\$, NewCharSet\$, nFileTemp\$)

## Where :

nFilename\$	the ASCII file.
CharSet\$	the string to be searched.
NewCharSet\$	the replacement string.
nFileTemp\$	a temporary file.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

If the nFilename string is an EMPTY string, the returned value is FALSE.  
If the char set string is an EMPTY string, the returned value is FALSE.  
If the new char set string is an EMPTY string, the returned value is FALSE.

If the length of char set is different of the length of new char set, the minimum length is used.

If the nFileTemp is an EMPTY string, a default temporary file is used.

The returned value can be negative and have the following value :

-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.

## Examples :

```
test& = cFileCopy("c:\autoexec.bat","c:autoexec.tab")
```

```
test& = cFileChangeChars("c:\autoexec.tab", "path", " PATH ", "", False)
```

**See also :** [File](#)

# KillDir, KillDirs

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**KillDir** delete the specified empty directory.

**KillDirs** delete the specified directory and its associated directories.

## Declare Syntax :

Declare Function cKillDir Lib "time2win.dll" (ByVal lpDir As String) As Integer

Declare Function cKillDirs Lib "time2win.dll" (ByVal lpDir As String, ByVal HeaderDirectory As Integer) As Integer

## Call Syntax :

test% = cKillDir(lpDir\$)

test% = cKillDirs(lpDir\$, HeaderDirectory%)

## Where :

lpDir\$	is the directory to proceed
HeaderDirectory%	specify if lpDir\$ must be delete also
test%	see below

## Comments :

For KillDir :

The directory must be empty, and it must not be the current working directory or the root directory.  
The returned value is TRUE if all is OK, <> TRUE if an error has occurred.

For KillDirs (only in TIME2WIN or T2WIN-16) :

Don't forget that this function can handle a maximum of 700 directories of 70 chars long each.

The returned value can be negative :  
-32760      allocation error for memory buffer.

For KillDirs (only in T2WIN-32) :

The returned value can be :

>= 0	if all is OK (the returned value is the number of dirs killed),
< 0	if an error has occurred (the returned value is the number of dirs that haven't been killed).

This function doesn't generates an VB Error if the specified dir not exists.

## Examples :

Dim Path            As String

Path = "c:\this"                            ' initialize the directory

Debug.Print cMakeDir(Path)                ' create the directory  
Debug.Print cKillDir(Path)                ' remove the directory

Path = "c:\this1\this2\this3\this4"        ' initialize the directories

```
Debug.Print cMakeMultipleDir(Path)' create the directories  
Debug.Print cKillDirs(Path, True)      ' remove the sub-directories and the header
```

```
Debug.Print cMakeMultipleDir(Path)' recreate the directories  
Debug.Print cKillDirs(Path, False)     ' remove the sub-directories only
```

**See also :** [File](#)



# KillDirFilesAll

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[KillDirFilesAll](#) delete all files specified by a mask in the specified directory and its associated sub-dir.

## **Declare Syntax :**

Declare Function cKillDirFilesAll Lib "[time2win.dll](#)" (ByVal lpDir As String, ByVal lpMask As String) As Integer

## **Call Syntax :**

test% = cKillDirFilesAll(lpDir\$, lpMask\$)

## **Where :**

lpDir\$	is the starting directory
lpMask\$	is the file mask to use
test%	>= 0 if all is OK. The returned value specified the total files deleted, < 0 if an error has occurred

## **Comments :**

Don't forget that this function can handle a maximum of 700 directories of 70 chars long each.

This function doesn't generates an VB Error if the specified dir not exists.

The returned value can be negative :

-32760          allocation error for memory buffer.

**See also :** [File](#)

# KillFile, KillFileAll

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[KillFile](#) delete the specified filename.

[KillFileAll](#) delete the specified filename with any attribute.

## **Declare Syntax :**

```
Declare Function cKillFile Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFileAll Lib "time2win.dll" (ByVal lpFilename As String) As Integer
```

## **Call Syntax :**

```
test% = cKillFile(lpFilename)
test% = cKillFileAll(lpFilename)
```

## **Where :**

lpFileName	the filename to proceed
test%	TRUE if all is OK
	<> TRUE if an error has occurred

## **Comments :**

If the file is a combination of READ-ONLY or SYSTEM or HIDDEN attribute, you must use cKillFileAll to remove it.

If the file is an opened file, the returned value is always <> TRUE.

If the file not exist, the returned value is always = TRUE.

This function doesn't generates an VB Error if the specified file not exists.

**See also :** [File](#)

# KillFiles, KillFilesAll

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[KillFiles](#) delete all files specified by a file mask.

[KillFilesAll](#) delete all files specified by a file mask even if some files are READ-ONLY files.

## **Declare Syntax :**

```
Declare Function cKillFiles Lib "time2win.dll" (ByVal lpFilename As String) As Integer  
Declare Function cKillFilesAll Lib "time2win.dll" (ByVal lpFilename As String) As Integer
```

## **Call Syntax :**

```
test% = cKillFiles(lpFilename)  
test% = cKillFilesAll(lpFilename)
```

## **Where :**

lpFilename	the mask file to proceed
test%	> 0 if all is OK. The returned value specified the total files deleted. = 0 if an error has occurred

## **Comments :**

If some files are a combination of READ-ONLY or SYSTEM or HIDDEN attributes, you must use cKillFilesAll to remove it.

If the mask is invalid or if the file not exists or if an error occurs when accessing the files, the return value is 0. This function doesn't generates an VB Error if the specified files not exists.

**See also :** [File](#)

# MakeDir, MakeMultipleDir

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[MakeDir](#) create the specified directory.

[MakeMultipleDir](#) create a multiple directory in one call.

## Declare Syntax :

Declare Function cMakeDir Lib "time2win.dll" (ByVal lpFilename As String) As Integer

Declare Function cMakeMultipleDir Lib "time2win.dll" (ByVal lpFilename As String) As Integer

## Call Syntax :

test% = cMakeDir(lpFilename)

test% = cMakeMultipleDir(lpFilename)

## Where :

lpFilename	the path for the new directory
test%	TRUE if all is OK <> TRUE if an error has occurred

## Comments :

The MakeDir function creates a new directory with the specified dirname. Only one directory can be created at a time, so only the last

component of dirname can name a new directory.

The MakeDir function does not do any translation of path delimiters. All operating systems accept either " or "/" internally as valid delimiters within paths.

This fonction is the same that Mkdir but doesn't generate an VB Error if a problem occurs.

The MakeMultipleDir function creates a new multiple directory with the specified dirname. MakeMultipleDir doesn't return an error if a sub-directory in the multiple directory is already present. The only final test is the existence of the full multiple directory when it was been created.

## Examples :

test% = cMakeDir("C:\")	' 13 (<> TRUE => an error has occurred)
test% = cMakeDir("C:\~~TEST~~")	' TRUE (no error, the directory has been created)
test% = cMakeMultipleDir("C:\~~TEST~~\TEST\TMP")	' TRUE (no error, the directory has been created)

**See also :** [File](#)

# GetDiskFree, GetDiskSpace, GetDiskUsed, GetDiskClusterSize

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 \(Win95/WinNT\)](#), [MSOffice 95](#)

## Purpose :

[GetDiskFree](#) retrieve the free disk space of a disk (hard disk or floppy disk).

[GetDiskSpace](#) retrieve the size of a disk (hard disk or floppy disk).

[GetDiskUsed](#) retrieve the part used of a disk (hard disk or floppy disk).

[GetDiskClusterSize](#) retrieve the size of a cluster on a disk (hard disk or floppy disk).

## Declare Syntax :

```
Declare Function cGetDiskFree Lib "time2win.dll" (ByVal lpDrive As String) As Double
Declare Function cGetDiskSpace Lib "time2win.dll" (ByVal lpDrive As String) As Double
Declare Function cGetDiskUsed Lib "time2win.dll" (ByVal lpDrive As String) As Double
Declare Function cGetDiskClusterSize Lib "time2win.dll" (ByVal lpDrive As String) As Double
```

## Call Syntax :

```
test# = cGetDiskFree(lpDrive)
test# = cGetDiskSpace(lpDrive)
test# = cGetDiskUsed(lpDrive)
test# = cGetDiskClusterSize(lpDrive)
```

## Where :

lpDrive                    is the letter for the disk  
test#                      is the result.

## Comments :

If the disk is not present or if the disk is not available or if an error occurs when accessing the disk, the returned value is always -1.

This function works with local disk (hard, floppy or cd-rom) als well on remote disk (network).

For TIME TO WIN 32-Bit : This function checks if the system uses Win95 OSR2 or not.

## Examples :

```
Dim diskFree     As Double
Dim diskSpace   As Double
Dim diskUsed    As Double
Dim clusterSize As Double

diskFree = cGetDiskFree("C")            ' 268197888
diskSpace = cGetDiskSpace("C")         ' 527654912
diskUsed = cGetDiskUsed("C")           ' 259457024
clusterSize = cGetDiskClusterSize("C") ' 8192
```

**See also :** [File](#)

# RcsCountFileDir

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[RcsCountFileDir](#) count the total directories or files in a specified directory (with recursivity or not).

## **Declare Syntax :**

Declare Function cRcsCountFileDir Lib "time2win.dll" (ByVal FileOrDir As Integer, ByVal FirstFileOrDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Long

## **Call Syntax :**

test& = cRcsCountFileDir(FileOrDir%, FirstFileOrDir\$, MaskDir\$, Recurse%)

## **Where :**

FileOrDir%	FALSE for directories TRUE for files
FirstFileOrDir\$	the starting directory (root or sub-dir) or file
MaskDir\$	the mask for performing the search (If this is an empty string, "**.*" is used)
Recurse%	FALSE for no recursivity TRUE for recursivity
test&	the number of sub-dirs or files founden in the specified directory

## **Comments :**

This function is a superset function of cCountDirectories and cCountFiles

For directory :

The internal '.' and '..' in each directory is not counted.  
The root directory is not counted.

For file :

The mask is the standard search mask (\*, ?, letters, ciphers).

**See also :** [File](#)

# RcsFileSize, RcsFileSizeOnDisk, RcsFilesSlack

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[RcsFileSize](#) return the logical size of files by file mask in a specified directory (with recursivity or not).

[RcsFileSizeOnDisk](#) return the physical size of files by file mask in a specified directory (with recursivity or not).

[RcsFilesSlack](#) return in one call, the slack from files by file mask in a specified directory (with recursivity or not), the logical size and the physical size.

## Declare Syntax :

```
Declare Function cRcsFileSize Lib "time2win.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Double
```

```
Declare Function cRcsFileSizeOnDisk Lib "time2win.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Double
```

```
Declare Function cRcsFilesSlack Lib "time2win.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer, Size1 As Double, Size2 As Double) As Integer
```

## Call Syntax :

```
test# = cRcsFileSize(FirstDir$, MaskDir$, Recurse%)
```

```
test# = cRcsFileSizeOnDisk(FirstDir$, MaskDir$, Recurse%)
```

```
test% = cRcsFilesSlack(FirstDir$, MaskDir$, Recurse%, Size1, Size2)
```

## Where :

FirstDir\$	the starting directory (root or sub-dir).
MaskDir\$	the mask for performing the search (If this is an empty string, "*" is used)
Recurse%	FALSE for no recursivity TRUE for recursivity
test#	is the size of all files fouden with the file mask.
test%	is the slack for all files fouden with the file mask.
Size1	is the logical size of all files fouden with the file mask.
Size2	is the physical size of all files fouden with the file mask.

## Comments :

If the mask is invalid or if the file not exists or if an error occurs when accessing the file, the return value is 0  
The slack of a file or files by file mask is the % of all spaces not used on all last clusters.

## Examples :

```
Dim Size As Double  
Dim Slack As Integer
```

```
Size = cRcsFileSize("C:\", "*.*", True) ' on my system, 437,896,805 bytes  
Size = cRcsFileSize("C:\", "*.*", False) ' on my system, 87,141,863 bytes
```

```
Size = cRcsFileSizeOnDisk("C:\", "*.*", True) ' on my system, 487,784,448 bytes  
Size = cRcsFileSizeOnDisk("C:\", "*.*", False) ' on my system, 87,343,104 bytes
```

```
Slack = cRcsFilesSlack("C:\", "*.*", True, 0, 0) ' on my system, 10 %  
Slack = cRcsFilesSlack("C:\", "*.*", False, 0, 0) ' on my system, 0%
```

**See also :** [File](#)

# FileSize, FileSizeOnDisk, FilesSlack

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**FileSize** return the logical size of all files specified by file mask.

**FileSizeOnDisk** return the physical size of all files specified by file mask.

**FilesSlack** return in one call, the slack from all files specified by file mask, the logical size and the physical size.

## Declare Syntax :

Declare Function cFileSize Lib "time2win.dll" (ByVal lpFilename As String) As Double

Declare Function cFileSizeOnDisk Lib "time2win.dll" (ByVal nFileName As String) As Double

Declare Function cFilesSlack Lib "time2win.dll" (ByVal nFileName As String, Size1 As Double, Size2 As Double) As Integer

## Call Syntax :

test# = cFileSize(nFilename)

test# = cFileSizeOnDisk(nFilename)

test% = cFilesSlack(nFilename, Size1, Size2)

## Where :

nFilename	is the mask file to proceed.
test#	is the size of all files fouden with the file mask.
test%	is the slack for all files fouden with the file mask.
Size1	is the logical size of all files fouden with the file mask.
Size2	is the physical size of all files fouden with the file mask.

## Comments :

If the mask is invalid or if the file not exists or if an error occurs when accessing the file, the return value is 0  
The slack of a file or files by file mask is the % of all spaces not used on all last clusters.

## Examples :

Dim Size           As Double  
Dim Slack           As Integer

Size = cFileSize("**.*")	' on my system, 5607689 bytes
Size = cFileSizeOnDisk("**.*")	' on my system, 5890048 bytes
Slack = cFilesSlack("**.*", 0, 0)	' on my system, 4 %

**See also :** [File](#)



# RUBYencrypt, RUBYdecrypt

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**RUBYencrypt** encode a string with a password using the RUBY algorithm (7 modes).

**RUBYdecrypt** decode a string with a password using the RUBY algorithm (7 modes).

## Declare Syntax :

Declare Function cRUBYencrypt Lib "time2win.dll" (Text As String, Key As String, ByVal Mode As Integer) As String

Declare Function cRUBYdecrypt Lib "time2win.dll" (Text As String, Key As String, ByVal Mode As Integer) As String

## Call Syntax :

testE = cRUBYencrypt(Text, Key)

testD = cRUBYdecrypt(Text, Key)

## Where :

Text	is the string to encrypt/decrypt	
Key	is the key to use for encryption/decryption	
Mode	Public Const RUBY_MODE_MINIMUM = 1	' speed is of the essence, security
secondary.	Public Const RUBY_MODE_DESK_LOCK = 2	' reasonable compromise of speed vs
security.	Public Const RUBY_MODE_DEAD_BOLT = 4	' default = probably good enough for most
things.	Public Const RUBY_MODE_PORTABLE_SAFE = 5	' security is more important than speed.
	Public Const RUBY_MODE_ANCHORED_SAFE = 8	' speed isn't much of a concern.
	Public Const RUBY_MODE_BANK_VAULT = 10	' your pentium has nothing better to do,
anyway.	Public Const RUBY_MODE_FORT_KNOX = 16	' be cool.
test	is the string encrypted/decrypted	

## Comments :

The Key is case sensitive.

The length of Text can be any size.

The length of Key must be greater or equal to 6 characters.

## Examples :

```
Dim Text As String
Dim Key As String
Dim Enc As String
Dim Dec As String
```

```
Text = "Under the blue sky, the sun is yellow"
```

```
Key = "a new encryption"
```

```
Enc = cRUBYencrypt(Text, Key, RUBY_MODE_DESK_LOCK)
```

```
Dec = cRUBYdecrypt(Enc, Key, RUBY_MODE_DESK_LOCK)
```

**See also :** [Encryption](#)

# CountFiles

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[CountFiles](#) count the total files founded in a specified directory.

## **Declare Syntax :**

Declare Function cCountFiles Lib "[time2win.dll](#)" (ByVal lpFilename As String) As Integer

## **Call Syntax :**

test = cCountFiles(lpFilename)

## **Where :**

lpFilename	the directory (root or sub-dir).
test	the number of files in the specified directory.

## **Comments :**

**See also :** [File](#)

# CountDirectories

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[CountDirectories](#) count the total directory in a specified directory.

## **Declare Syntax :**

Declare Function cCountDirectories Lib "[time2win.dll](#)" (ByVal lpFilename As String) As Integer

## **Call Syntax :**

```
test = cCountDirectories(lpFilename)
```

## **Where :**

lpFilename	the directory (root or sub-dir).
test	the number of sub-dir founded in the specified directory.

## **Comments :**

**See also :** [File](#)

# TruncatePath

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[TruncatePath](#) truncate a long path with filename.

## Declare Syntax :

Declare Function cTruncatePath Lib "[time2win.dll](#)" (ByVal nFilename As String, ByVal NewLength As Integer) As String

## Call Syntax :

Test\$ = cTruncatePath(nFilename, NewLength%)

## Where :

nFilename	is the path.
NewLength%	is the new length of the path.
Test\$	is the returned truncated path.

## Comments :

If 'nFilename' is an invalid file, the returned path is always an EMPTY string.  
If 'NewLength' is below to 25, the returned path is always an EMPTY string.  
If the length of 'nFilename' is below 25, the 'nFilename' is returned.

## Examples :

```
Dim Tmp           As String
Dim Test         As String
Dim NewLength    As Integer

NewLength = 25

Tmp = "time2win.bas"
Debug.Print cTruncatePath(Tmp, NewLength)           ' time2win.bas

Tmp = "windows\system\time2win.bas"
Debug.Print cTruncatePath(Tmp, NewLength)           ' windows.....time2win.bas

Tmp = "c:\windows\system\time2win.bas"
Debug.Print cTruncatePath(Tmp, NewLength)           ' c:\windows...time2win.bas

Tmp = "c:\windows\system\vb\time2win\time2win.bas"
Debug.Print cTruncatePath(Tmp, NewLength)           ' c:\windows...time2win.bas

Tmp = "c:\windows\system\vb\source\time2win\time2win.bas"
Debug.Print cTruncatePath(Tmp, NewLength)           ' c:\windows...time2win.bas
```

**See also :** [File](#)

# SplitPath

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SplitPath](#) break a full path into its four components.

## Declare Syntax :

Declare Function cSplitPath Lib "[time2win.dll](#)" (ByVal nFilename As String, SPLITPATH As Any) As Integer

## Call Syntax :

test% = cSplitPath(nFilename, SPLITPATH)

## Where :

nFilename	is the name of a file containing the full path to access it.
SPLITPATH	is the type'd variable ' <a href="#">tagSPLITPATH</a> ' to receive the four components.
test%	TRUE if all is OK, FALSE if an error occurs.

## Comments :

If the file is not available or if an error occurs when accessing the file, the returned value is always 0.

The four components are :

nDrive	Contain the drive letter followed by a colon (:) if a drive is specified in path.
nDir	Contain the path of subdirectories, if any, including the trailing slash.
nName	Contain the base filename without any extensions.
nExt	Contain the filename extension, if any, including the leading period (.).

The return parameters in SPLITPATH will contain empty strings for any path components not found in path.

## Examples :

```
Dim SPLITPATH          As tagSPLITPATH
```

```
Call cSplitPath("C:\AUTOEXEC.BAT", SPLITPATH)
```

On my system :

SPLITPATH.nDrive	is "C"
SPLITPATH.nDir	is "\"
SPLITPATH.nName	is "AUTOEXEC"
SPLITPATH.nExt	is ".BAT"

**See also :** [File](#)

# MakePath

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[MakePath](#) create a single path, composed of a drive letter, directory path, filename, and filename extension.

## Declare Syntax :

Declare Function cMakePath Lib "time2win.dll" (ByVal nDrive As String, ByVal nDir As String, ByVal nFilename As String, ByVal Ext As String) As String

## Call Syntax :

```
test$ = cMakePath(nDrive, nDir, nFilename, Ext)
```

## Where :

### nDrive

The nDrive argument contains a letter (A, B, etc.) corresponding to the desired drive and an optional trailing colon. MakePath routine will insert the colon automatically in the composite path if it is missing. If drive is a null character or an empty string, no drive letter and colon will appear in the composite path string.

### nDir

The nDir argument contains the path of directories, not including the drive designator or the actual filename. The trailing slash is optional, and either forward slashes (/) or backslashes (\) or both may be used in a single dir argument. If a trailing slash (/ or \) is not specified, it will be inserted automatically. If dir is a null character or an empty string, no slash is inserted in the composite path string.

### nFilename

The nFilename argument contains the base filename without any extensions. If nFilename is an EMPTY string, no filename is inserted in the composite path string.

### Ext

The Ext argument contains the actual filename extension, with or without a leading period (.). MakePath routine will insert the period automatically if it does not appear in ext. If ext is a null character or an empty string, no period is inserted in the composite path string.

## Comments :

## Examples :

```
test1$ = cMakePath("c","tmp","test","dat")
test2$ = cMakePath("c","\tmp","test","dat")
test3$ = cMakePath("c","tmp","test","")
test4$ = cMakePath("c","","test","dat")
```

On my system :

```
test1$ = "c:\tmp\test.dat"
test2$ = "c:\tmp\test.dat"
test3$ = "c:\tmp\test"
```

test4\$ = "c:test.dat"

**See also :** [File](#)

# FullPath

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FullPath](#) convert a partial path stored in path to a fully qualified path.

## **Declare Syntax :**

```
Declare Function cFullPath Lib "time2win.dll" (ByVal nFilename As String) As String
```

## **Call Syntax :**

```
test$ = cFullPath(nFilename)
```

## **Where :**

nFilename	is the partial path.
test\$	is the returned full qualified path.

## **Comments :**

If the file is not available or if an error occurs when accessing the file, the returned path is always an EMPTY string.

## **Examples :**

```
tmp$ = cFilesInDirectory(cGetDefaultCurrentDir() + "\*.*", True) 'retrieves the first file in the default current directory  
test$ = cFullPath(tmp$)
```

On my system :

```
tmp$ = "AWARE.BAS"  
test$ = "M:\VB\AWARE.BAS"
```

**See also :** [File](#)



# RenameFile

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[RenameFile](#) rename a file or moves a file from one path to an other path.

## Declare Syntax :

Declare Function cRenameFile Lib "[time2win.dll](#)" (ByVal lpFilename1 As String, ByVal lpFilename2 As String) As Integer

## Call Syntax :

```
test% = cRenameFile(lpFilename1, lpFilename2)
```

## Where :

lpFileName1	the old filename to rename
lpFileName2	the new filename to be used
test%	TRUE if all is OK <> TRUE if an error has occurred

## Comments :

The rename function renames the file or directory specified by lpFilename1 to the name given by lpFilename2. The lpFilename1 must be the path of an existing file or directory. The lpFilename1 must not be the name of an existing file or directory. The rename function can be used to move a file from one directory to another by giving a different path in the lpFilename2 argument. However, files cannot be moved from one device to another (for example, from drive A to drive B). Directories can only be renamed, not moved. This function doesn't generates an VB Error if the specified old filename not exists.

## Examples :

**See also :** [File](#)

# UniqueFileName

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**UniqueFileName** create a unique filename by modifying the given template argument. The template argument must be a string with two chars maximum.

## Declare Syntax :

Declare Function cUniqueFileName Lib "time2win.dll" (Txt As String) As String

## Call Syntax :

test\$ = cUniqueFileName(Txt)

## Where :

Txt                    the filename pattern. If the size is greater than 2, the default pattern is used.  
test\$                 the unique filename in the form of the chars speciefien in Txt plus one char and five digits.

## Comments :

The alphanumeric character is 0 ('0') the first time cUniqueFileName is Called with a given template. In subsequent Calls from the same process with copies of the same template, cUniqueFileName checks to see if previously returned names have been used to create files. If no file exists for a given name, cUniqueFileName returns that name. If files exist for all previously returned names, cUniqueFileName creates a new name by replacing the alphanumeric character in the name with the next available lowercase letter. For example, if the first name returned is t012345 and this name is used to create a file, the next name returned will be ta12345. When creating new names, cUniqueFileName uses, in order, '0' and then the lowercase letters 'a' through 'z'.

Note that the original template is modified by the first Call to cUniqueFileName. If you then Call the cUniqueFileName function again with the same template (i.e., the original one), you will get an error.

The cUniqueFileName function generates unique filenames but does not create or open files. If the filename returned is not created, each subsequent Calls returns the same filename.

If the filename pattern is not specified (by passing an EMPTY string), the default pattern '~~' is used.

## Examples :

```
Dim Tmp      As String

Tmp = cUniqueFileName("MC")                    ' "MC040201"
Debug.Print Tmp
Close #1
Open "c:\" + Tmp For Output Shared As #1
Close #1

Tmp = cUniqueFileName("MC")                    ' "MCa40201"
Debug.Print Tmp
Close #1
Open "c:\" + Tmp For Output Shared As #1
Close #1

Tmp = cUniqueFileName("MC")                    ' "MCb40201"
Debug.Print Tmp
Close #1
Open "c:\" + Tmp For Output Shared As #1
```

Close #1

If you don't create the file, the same filename is returned, see below :

```
Tmp = cUniqueFileName("MC")           ' "MCc40201"  
Tmp = cUniqueFileName("MC")           ' "MCc40201"  
Tmp = cUniqueFileName("MC")           ' "MCc40201"
```

**See also :** [File](#)

# FilesInDirectory

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FilesInDirectory](#) retrieve each file in the specified directory.

## Declare Syntax :

Declare Function cFilesInDirectory Lib "[time2win.dll](#)" (ByVal nFilename As String, ByVal firstnext As Integer) As String

## Call Syntax :

```
test$ = cFilesInDirectory(nFilename, firstnext )
```

## Where :

nFilename	the directory to proceed with the file mask (*. * for all)
firstnext	TRUE for the first file FALSE for each next file
test\$	the returned file

## Comments :

## Examples :

```
Dim i           As Integer
Dim Tmp        As String

i = 0
Tmp = cFilesInDirectory("c:\*.*", True)

Debug.Print "The first 7 files in C:\ are : "

Do While (Len(Tmp) > 0)
    Debug.Print Tmp
    Tmp = cFilesInDirectory("c:\*.*", False)
    i = i + 1
    If (i >= 7) Then Exit Do
Loop
```

On my system:

The first 7 files in C:\ are :

```
863DATA
WINA20.386
AUTOEXEC.BAT
COMMAND.COM
IMAGE.DAT
BOOTSECT.DOS
ACD.IDX
```

**See also :** [File](#)

# SubDirectory

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SubDirectory](#) retrieve all sub-directories from the specified mask.

## Declare Syntax :

Declare Function cSubDirectory Lib "[time2win.dll](#)" (ByVal nFilename As String, ByVal firstnext As Integer) As String

## Call Syntax :

```
test$ = cSubDirectory(nFilename, firstnext)
```

## Where :

nFilename	the specified mask
firstnext	TRUE to retrieve the first directory FALSE to retrieve the next directory
test\$	the retrieved directory

## Comments :

To retrieve all sub-directory is a directory, you must Call first this function with the firstnext argument on TRUE and set it to FALSE for all next directory

## Examples :

```
Dim Test      As String
Test = cSubDirectory("c:\*.*", True)
Do Until (Len(Test) = 0)
    Debug.Print Test
    Test = cSubDirectory("c:\*.*", False)
Loop
```

Directories with "c:\\*.\*" argument are :

```
DOS
TEMP
TMP
BAD.DIR
```

**See also :** [File](#)

# FileSet.X

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileSetAllAttrib](#) set all attributes of a file.  
[FileSetArchive](#) set the archive attribute of a file.  
[FileSetHidden](#) set the hidden attribute of a file.  
[FileSetReadOnly](#) set the read-only attribute of a file.  
[FileSetSystem](#) set the system attribute of a file.  
[FileSetFlag](#) set the specified attributes of a file.  
[FileSetAttrib](#) set in one call, attributes of a given file.

## Declare Syntax :

```
Declare Function cFileSetAllAttrib Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetArchive Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetHidden Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetReadOnly Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetSystem Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetFlag Lib "time2win.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer

Declare Function cFileSetAttrib Lib "time2win.dll" (ByVal nFilename As String, nFileAttribute As Any) As Integer
```

## Call Syntax :

```
status = cFileSetAllAttrib(nFilename)
status = cFileSetArchive(nFilename)
status = cFileSetHidden(nFilename)
status = cFileSetReadOnly(nFilename)
status = cFileSetSystem(nFilename)
status = cFileSetFlag(nFilename, nStatus)

test% = cFileSetAttrib(nFilename, nFileAttribute)
```

## Where :

nFilename	is the filename to change the attributes
nStatus	is a combination of <a href="#">attributes</a>
nFileAttribute	the type variable ' <a href="#">FileAttributeType</a> ' (only for cFileSetAttrib)
status	TRUE if all is OK. FALSE if an error has been detected.

## Comments :

## Examples :

```
nFilename = "tmp.tmp"
nStatus = A_RDONLY or A_SYSTEM or A_HIDDEN

status = cFileSetAllAttrib(nFilename)
status = cFileSetFlag(nFilename, nStatus)
```

**See also :** [File](#)

# FileReset.X

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileResetAllAttrib](#) reset all attributes of a file.  
[FileResetArchive](#) reset the archive attribute of a file.  
[FileResetHidden](#) reset the hidden attribute of a file.  
[FileResetReadOnly](#) reset the read-only attribute of a file.  
[FileResetSystem](#) reset the system attribute of a file.  
[FileResetFlag](#) reset the specified attributes of a file.

## Declare Syntax :

```
Declare Function cFileResetAllAttrib Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetArchive Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetHidden Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetReadOnly Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetSystem Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetFlag Lib "time2win.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer
```

## Call Syntax :

```
status = cFileResetAllAttrib(nFilename)
status = cFileResetArchive(nFilename)
status = cFileResetHidden(nFilename)
status = cFileResetReadOnly(nFilename)
status = cFileResetSystem(nFilename)
status = cFileResetFlag(nFilename, nStatus)
```

## Where :

nFilename	is the filename to change the attributes
nStatus	is a combination of <a href="#">attributes</a>
status	TRUE if all is OK. FALSE if an error has been detected.

## Comments :

## Examples :

```
nFilename = "tmp.tmp"
nStatus = A_RDONLY or A_SYSTEM or A_HIDDEN
```

```
status = cFileResetAllAttrib(nFilename)
status = cFileResetFlag(nFilename, nStatus)
```

**See also :** [File](#)

# FileDrive

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FileDrive](#) extracts the drive on which the file is present.

## **Declare Syntax :**

```
Declare Function cFileDrive Lib "time2win.dll" (ByVal lpFilename As String) As String
```

## **Call Syntax :**

```
test$ = cFileDrive(lpFilename)
```

## **Where :**

lpFilename	the file to proceed
test\$	EMPTY is the file not exist or an error occurs when accessing the file DRIVE LETTER for the file

## **Comments :**

## **Examples :**

**See also :** [File](#)



# FilesInDirOnDisk

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FilesInDirOnDisk](#) write all files from a specified directory into a file on disk.

## **Declare Syntax :**

Declare Function cFilesInDirOnDisk Lib "[time2win.dll](#)" (ByVal nFile As String, ByVal nFilename As String, ByVal nAttribute As Integer) As Integer

## **Call Syntax :**

test% = cFilesInDirOnDisk(nFile\$, nFilename\$, nAttribute)

## **Where :**

nFile\$	the file on disk
nFilename	the directory to proceed with the file mask (if this is an empty string, *.* is used)
nAttribute	the attribute to search (see Constants and Types declaration)
test%	the number of files founded

## **Comments :**

When nAttribute is a positive value, the search is based on an OR test. If one or more attributes of file is founded, the file is taken.

When nAttribute is a negative value, the search is based on an AND test. If all attributes of files are founded, the file is taken.

## **Examples :**

```
Dim i As Integer
```

```
i = cFilesInDirOnDisk("c:\test.tmp", *.* , A_ALL)
```

**See also :** [File](#)

# FilesInDirToArray

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

`FilesInDirToArray` read all files from a specified directory into an array.

## Declare Syntax :

Declare Function `cFilesInDirToArray` Lib "[time2win.dll](#)" (ByVal `nFilename` As String, ByVal `nAttribute` As Integer, `array()` As Any) As Integer

## Call Syntax :

```
test% = cFilesInDirToArray(nFilename$, nAttribute%, Array())
```

## Where :

<code>nFilename</code>	the directory to proceed with the file mask (if this is an empty string, *.* is used)
<code>nAttribute</code>	the attribute to search (see Constants and Types declaration)
<code>Array()</code>	is the variable array string with one dimension.
<code>test%</code>	>=0 is the number of file in the specified directory, < 0 is an error occurs (error n° is the negative value of all DA_x values, see Constants and Types declaration ).

## Comments :

When `nAttribute` is a positive value, the search is based on an OR test. If one or more attributes of file is founded, the file is taken.

When `nAttribute` is a negative value, the search is based on an AND test. If all attributes of files are founded, the file is taken.

This function can handle only a variable type'd string derived from `tagVARSTRING` (see below).

Don't forget that if you use the 'ReDim' statement at the procedure level without have declared the array als Global, you must initialize the array before using this function (see below). You must initialize the array with enough space to handle the size of the file This is due to a VB limitation.

```
Type tagVARSTRING
  Contents          As String
End Type
```

## Examples :

```
ReDim AD(-999 To 1000)          As tagVARSTRING
```

```
For i = -999 To 1000
  AD(i).Contents = Space$(256)
Next i
```

```
Debug.Print cFilesInDirToArray("c:\*.*", A_ALL, AD())
```

```
Debug.Print AD(-999).Contents
Debug.Print AD(-998).Contents
```

**See also :** [File](#)

# FileDate.X, FileTime.X

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileDateCreated](#) retrieve the date when the file has been created.

[FileLastDateAccess](#) retrieve the date when the file has been last accessed.

[FileLastDateModified](#) retrieve the date when the file has been last modified.

[FileTimeCreated](#) retrieve the time when the file has been created.

[FileLastTimeAccess](#) retrieve the time when the file has been last accessed.

[FileLastTimeModified](#) retrieve the time when the file has been last modified.

## Declare Syntax :

```
Declare Function cFileDateCreated Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastDateAccess Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastDateModified Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileTimeCreated Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastTimeAccess Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastTimeModified Lib "time2win.dll" (ByVal lpFilename As String) As String
```

## Call Syntax :

```
test = cFileDateCreated(lpFilename)
test = cFileLastDateAccess(lpFilename)
test = cFileLastDateModified(lpFilename)
test = cFileTimeCreated(lpFilename)
test = cFileLastTimeAccess(lpFilename)
test = cFileLastTimeModified(lpFilename)
```

## Where :

lpFileName	the file to read date and/or time
test	HH:MM for time
	DD/MM/YYYY for date

## Comments :

For TIME2WIN, T2WIN-16 :

The created, access, modified time/date are the same because Win 3.xx don't handle the different date/time information.

## Examples :

**See also** : [File](#)

# GetDriveType

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetDriveType](#) determine whether a disk drive is removable, fixed, or remote.

## Declare Syntax :

Declare Function cGetDriveType Lib "[time2win.dll](#)" (ByVal lpDrive As String) As Integer

## Call Syntax :

```
test% = cGetDriveType(lpDrive$)
```

## Where :

lpDrive\$                    is the letter disk to proceed  
test%                        is the returned drive type

## Comments :

## Examples :

On my system :

```
test% = cGetDriveType("A")        ' DRIVE_REMOVABLE  
test% = cGetDriveType("C")        ' DRIVE_FIXED  
test% = cGetDriveType("X")        ' DRIVE_CDROM  
test% = cGetDriveType("Z")        ' DRIVE_REMOTE
```

**See also :** [File](#)

# FileStatistics

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileStatistics](#) count the lines, words and chars in a specified file.

## Declare Syntax :

Declare Function cFileStatistics Lib "[time2win.dll](#)" (ByVal nFilename As String, nLines As Long, nWords As Long, nChars As Long) As Long

## Call Syntax :

```
test& = cFileStatistics(nFilename$, nLines, nWords, nChars)
```

## Where :

nFilename\$	is the file to proceed
nLines&	is the returned number of lines
nWords&	is the returned number of words
nChars&	is the returned number of chars
test&	> 0 if all is OK (the returned value is the total bytes in the file), < 0 if an error has occurred.

## Comments :

If all is ok, the returned value must be equal to nChars.

The returned value can be negative and have the following value :

-32730	reading error for file.
-32750	opening error for file.
-32760	allocation error for memory buffer.

## Examples :

```
test& = cFileStatistics("c:\autoexec.bat", nLines&, nWords&, nChars&)
```

On my system :

nLines&	is 90
nWords&	is 282
nChars&	is 2212
test&	is 2212

```
test& = cFileStatistics("c:\config.sys", nLines&, nWords&, nChars&)
```

On my system :

nLines&	is 15
nWords&	is 44
nChars&	is 506
test&	is 506

**See also :** [File](#)

# FilePathExists

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FilePathExists](#) verify if the specified file is present.

## **Declare Syntax :**

Declare Function cFilePathExists Lib "[time2win.dll](#)" (ByVal lpFilename As String) As Integer

## **Call Syntax :**

test% = cFilePathExists(lpFilename)

## **Where :**

lpFilename	the file to proceed
test%	TRUE is the file exists <> TRUE if the file not exists or if an error occurs when accessing the file.

## **Comments :**

## **Examples :**

**See also :** [File](#)

# SearchDir, SearchFile, SearchFileAttrib

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[SearchDir](#) perform a directory pattern match starting with a specified path.

[SearchFile](#) perform a file pattern match starting with a specified path.

[SearchFileAttrib](#) perform a file (with attribute) pattern match starting with a specified path.

## Declare Syntax :

Declare Function cSearchDir Lib "time2win.dll" (ByVal lpStartPath As String, ByVal lpFileMask As String, ByVal lpFileResult As String) As Long

Declare Function cSearchFile Lib "time2win.dll" (ByVal lpStartPath As String, ByVal lpFileMask As String, ByVal lpFileResult As String) As Long

Declare Function cSearchFileAttrib Lib "time2win.dll" (ByVal lpStartPath As String, ByVal lpFileMask As String, ByVal lpFileResult As String, ByVal iSearchAttribute As Integer) As Long

## Call Syntax :

lngResult& = cSearchDir(lpStartPath\$, lpFileMask\$, lpFileResult\$, iSearchAttribute%)

lngResult& = cSearchFile(lpStartPath\$, lpFileMask\$, lpFileResult\$, iSearchAttribute%)

## Where :

lpStartPath\$	is the starting path to begin the search.
lpFileMask\$	is the file mask to match.
lpFileResult\$	is the file where the result will be written.
iSearchAttribute%	is the file attribute mask (see <a href="#">attributes</a> )
lngResult&	is the number of dirs/files founded.

## Comments :

For the pattern match, see [PatternMatch](#).

The search for directory/file is case-insensitive.

## Examples :

```
Debug.Print cSearchDir("c:\", "time2win.dll", "c:\tmp\test.srd")
```

```
' 4
' contents of the file c:\tmp\test.srd is :
'     c:\MCR_APPS\CSERVE\T2WIN-32
'     c:\MCR_APPS\REGISTER\T2WIN-32
'     c:\MCR_APPS\SOURCE_B\T2win-32
'     c:\MCR_APPS\SOURCE_C\T2WIN-32
```

```
Debug.Print cSearchFile("c:\", "t2win-32.dll", "c:\tmp\test.srf")
```

```
' 3
' contents of the file c:\tmp\test.srd is :
'     c:\WIN95\SYSTEM\T2win-32.dll
'     c:\MCR_APPS\CSERVE\T2WIN-32\T2win-32.dll
'     c:\MCR_APPS\SOURCE_C\T2WIN-32\Release\T2win-32.dll
```

```
Debug.Print cSearchFile("c:\", "t2win-32.dll", "c:\tmp\test.srf", A_ARCHIVE)
```

```
' 3
' contents of the file c:\tmp\test.srd is :
'     c:\WIN95\SYSTEM\T2win-32.dll
'     c:\MCR_APPS\CSERVE\T2WIN-32\T2win-32.dll
'     c:\MCR_APPS\SOURCE_C\T2WIN-32\Release\T2win-32.dll
```

**See also :** [File](#)



# CmpFile.X

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[CmpFileAttribute](#) compare the attribute of two files.  
[CmpFileContents](#) compare the contents of two files.  
[CmpFileSize](#) compare the size of two files.  
[CmpFileTime](#) compare the date and time of two files.

## Declare Syntax :

```
Declare Function cCmpFileAttribute Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String) As Integer
Declare Function cCmpFileContents Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal sensitivity
As Integer) As Integer
Declare Function cCmpFileSize Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String) As Integer
Declare Function cCmpFileTime Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String) As Integer
```

## Call Syntax :

```
test% = cCmpFileAttribute(file1, file2)
test% = cCmpFileContents(file1, file2, sensitivity)
test% = cCmpFileSize(file1, file2)
test% = cCmpFileTime(file1, file2)
```

## Where :

file1\$	is the first file.
file2\$	is the second file.
sensitivity%	TRUE for case sensitive, FALSE for no case sensitive.
test%	-1 if file1 < file2 for the specified function, 0 if file1 = file2 for the specified function, 1 if file1 > file2 for the specified function.

## Comments :

When using cCmpFileAttribute, only -1 (attribute are the same) or 0 (attribute are different) or -2 (error) is returned.

When using cCmpFileContents

-1	files are the same
0	files are not the same, or file size differs
-32740	reading error for files.
-32750	opening error for file 1.
-32751	opening error for file 2.
-32760	allocation error for memory buffer 1.
-32761	allocation error for memory buffer 2.

## Examples :

```
test% = cCmpFileAttribute("c:\command.com", "c:\dos\command.com")
test% = cCmpFileContents("c:\command.com", "c:\dos\command.com", True)
test% = cCmpFileContents("c:\command.com", "c:\dos\command.com", False)
test% = cCmpFileSize("c:\command.com", "c:\dos\command.com")
test% = cCmpFileTime("c:\command.com", "c:\dos\command.com")
```

**See also :** [File](#)

## Mail & News : Overview

<a href="#"><u>MNInitialize</u></a>	initializes the Mail & News memory to perform encoding/decoding.
<a href="#"><u>MNShutdown</u></a>	shut downs the Mail & News memory (cleanup).
<a href="#"><u>MNGetOption</u></a>	retrieves options.
<a href="#"><u>MNSetOption</u></a>	sets options.
<a href="#"><u>MNEncode</u></a> "XXencode".	encodes a file in single-part or multi-part using "UUencode", "Mime-Base64",
<a href="#"><u>MNLoadInList</u></a>	loads a file to be processed/decoded in the list of files.
<a href="#"><u>MNCountInList</u></a>	counts the number of files in the list of files (not only the file loaded in
<a href="#"><u>MNLoadInList</u></a> but also all sub-parts	on the file like attachment).
<a href="#"><u>MNFirstInList</u></a>	retrieves a pointer to the first file in the list of files.
<a href="#"><u>MNWalkInList</u></a>	walks in the list of files.
<a href="#"><u>MNGetSubjectInList</u></a>	retrieves the subject of a file in the list of files.
<a href="#"><u>MNGetMimeIDInList</u></a>	retrieves the "mime identifier" (if any) of a file in the list of files.
<a href="#"><u>MNGetMimeContentInList</u></a>	retrieves the "mime content-type" (if any) of a file in the list of files.
<a href="#"><u>MNRenameInList</u></a>	renames a file in the list of files.
<a href="#"><u>MNDecodeFromList</u></a> "BinHex".	decodes a file from the list of files using "UUE", "Mime-Base64/PT/QP", "XXE",

' structure for split path

Type tagSPLITPATH

nDrive As String

nDir As String

nName As String

nExt As String

End Type

# Mail & News : List Management

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[MNLoadInList](#) loads a file to be processed/decoded in the list of files.

[MNCountInList](#) counts the number of files in the list of files (not only the file loaded in MNLoadInList but also all sub-parts on the file like attachment).

[MNFirstInList](#) retrieves a pointer to the first file in the list of files.

[MNWalkInList](#) walks in the list of files.

[MNGetSubjectInList](#) retrieves the subject of a file in the list of files.

[MNGetMimeIDInList](#) retrieves the "mime identifier" (if any) of a file in the list of files.

[MNGetMimeContentInList](#) retrieves the "mime content-type" (if any) of a file in the list of files.

[MNRenameInList](#) renames a file in the list of files.

## Declare Syntax :

```
Declare Function cMNLoadInList Lib "time2win.dll" (ByVal FileName As String, ByVal DeleteAfterDecoding As Long) As Long
```

```
Declare Function cMNCountInList Lib "time2win.dll" () As Long
```

```
Declare Function cMNFirstInList Lib "time2win.dll" (Ptr As Long) As Integer
```

```
Declare Function cMNWalkInList Lib "time2win.dll" (ByVal Ptr As Long, FileName As String, TypeOfEncoding As Integer, FileState As Long, NextPtr As Long) As Integer
```

```
Declare Function cMNGetSubjectInList Lib "time2win.dll" (ByVal Ptr As Long) As String
```

```
Declare Function cMNGetMimeIDInList Lib "time2win.dll" (ByVal Ptr As Long) As String
```

```
Declare Function cMNGetMimeContentInList Lib "time2win.dll" (ByVal Ptr As Long) As String
```

```
Declare Function cMNRenameInList Lib "time2win.dll" (ByVal Ptr As Long, ByVal NewFileName As String) As Integer
```

## Call Syntax :

```
IResult1& = cMNLoadInList(FileName$, DeleteAfterDecoding%)
```

```
IResult2& = cMNCountInList()
```

```
iResult1% = cMNFirstInList(Ptr&)
```

```
iResult2% = cMNWalkInList(Ptr&, FileName$, TypeOfEncoding%, FileState&, NextPtr&)
```

```
sResult1$ = cGetSubjectInList(Ptr&)
```

```
sResult2$ = cGetMimeIDInList(Ptr&)
```

```
sResult3$ = cGetMimeContentInList(Ptr&)
```

```
iResult3% = cMNRenameInList(Ptr&, NewFileName$)
```

## Where :

FileName\$ is the name of the file to be loaded in the list of files

DeleteAfterDecoding% True if the file must be deleted after decoding

False if the file is not deleted after decoding

Ptr& is a pointer to a file in the list of files

TypeOfEncoding% is the MN\_???\_ENCODED code

FileState& is an OR'ed combination of MNSTATE\_ code

NextPtr& is a pointer of the next file in the list of files

NewFileName\$ is the new name of the file pointed by Ptr&

IResult1& < 0 : negative values of the MNRET\_ code

>=0 : size of the loaded file

IResult2& number of files in the current list of files

iResult1% True if there is a file in the current list of files

False if no files

iResult2% True if there is a file in the current list of files

False if no files

sResult1\$, sResult2\$, sResult3\$ the asked information

iResult3% True if the rename is successful

False if an error has occurred

**Comments :**

**Examples :**

```
Debug.Print cMNInitialize()
Debug.Print cMNLdInList("c:\temp\message.nws", False)
Debug.Print cMNCntInList()

If (cMNFrstInList(Ptr) = True) Then
    Do
        rc% = cMNWalkInList(Ptr, FileName, TypeOfEncoding, FileState, NextPtr)
        Debug.Print "Filename : "; FileName
        Debug.Print "  TypeOfEncoding : "; TypeOfEncoding
        Debug.Print "  FileState : "; FileState
        Debug.Print "  Subject : "; cGetSubjectInList(Ptr)
        Debug.Print "  MimeID : "; cGetMimeIDInList(Ptr)
        Debug.Print "  MimeContent : "; cGetMimeContentInList(Ptr)
        Ptr = NextPtr
    Loop Until (rc = False)
End If
```

**See also :** [Mail & News](#)

# FileGetAttrib

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FileGetAttrib](#) set in one call, attributes of a given file.

## **Declare Syntax :**

Declare Function cFileGetAttrib Lib "[time2win.dll](#)" (ByVal nFilename As String, nFileAttribute As Any) As Integer

## **Call Syntax :**

status% = cFileGetAttrib(nFilename, nFileAttribute)

## **Where :**

nFilename	is the filename to change the attributes
nFileAttribute	the type'd variable ' <a href="#">FileAttributeType</a> '
status	TRUE if all is OK. FALSE if an error has been detected.

## **Comments :**

## **Examples :**

**See also :** [File](#)

# FileCopy2

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[FileCopy2](#) copy one file to an another file.

## **Declare Syntax :**

Declare Function cFileCopy2 Lib "[time2win.dll](#)" (ByVal file1 As String, ByVal file2 As String) As Long

## **Call Syntax :**

```
test& = cFileCopy2(file1, file2)
```

## **Where :**

file1\$	is the source file.
file2\$	is the destination file.
test&	= True : if all is OK, > 0 : if an error has occurred.

## **Comments :**

This function use the standard 'CopyFile' function from Win32 SDK.  
However, this function is not a speedy function.

## **Examples :**

```
test& = cFileCopy2("c:\autoexec.bat", "c:\autoexec.tab")
```

**See also :** [File](#)

' definition for file attributes

Public Const A_RDONLY = &H1	' Read only file
Public Const A_HIDDEN = &H2	' Hidden file
Public Const A_SYSTEM = &H4	' System file
Public Const A_SUBDIR = &H10	' Subdirectory
Public Const A_ARCHIVE = &H20	' Archive file
Public Const A_NORMAL = &H80	' Normal file - No read/write restrictions
Public Const A_COMPRESSED = &H800	' Compressed file
Public Const A_NORMAL_ARCHIVE = &HFE	' Normal, Archive
Public Const A_ALL = &HFF	' Normal, Archive, Read-Only, Hidden, System



' definition for drive type

```
Public Const DRIVE_UNKNOWN = 0 ' drive type can't be founded, drive not present or unknow.
Public Const DRIVE_NO_ROOT_DIR = 1 ' drive type can't be founded, drive not present or unknow (Win95).
Public Const DRIVE_REMOVABLE = 2 ' disk can be removed from the drive.
Public Const DRIVE_FIXED = 3 ' disk cannot be removed from the drive.
Public Const DRIVE_REMOTE = 4 ' drive is a remote, or network, drive.
Public Const DRIVE_CDROM = 5 ' drive is a cd-rom.
Public Const DRIVE_RAMDISK = 6 ' drive is a ram disk.
```

' definition for file sort

Public Const SORT\_ASCENDING = 1

Public Const SORT\_DESCENDING = 2

Public Const SORT\_CASE\_SENSITIVE = 4

Public Const SORT\_CASE\_INSENSITIVE = 8

' definition for file uuop

Public Const MODE\_UUENCODE = 0

Public Const MODE\_UUDECODE = 1

# FileUUCP

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

FileUUCP uuencode/uudecode a file with header/footer or not (this is can be usefull for Internet).

## Declare Syntax :

Declare Function cFileUUCP Lib "time2win.dll" (ByVal lpFileName1 As String, ByVal lpFileName2 As String, ByVal EncodeDecode As Integer, ByVal HeaderOrNot As Integer) As Long

## Call Syntax :

lngResult& = cFileUUCP(lpFileName1\$, lpFileName2\$, EncodeDecode%, HeaderOrNot%)

## Where :

lpFileName1\$	is the file to be uuencoded/uudecoded
lpFileName2\$	is the file uuencoded/uudecoded
EncodeDecode%	is the <u>mode</u> for encoding/decoding
HeaderOrNot%	TRUE : add header "begin 644 filename" and footer "end" FALSE : no header/footer
lngResult&	< 0 : an error has occurred >= 0 : the size of the file uuencoded/uudecoded

## Comments :

## Examples :

```
Dim lngResult      As Long
Dim strResult      As String
Dim strDisplay     As String
```

```
Dim File1          As String
Dim File2          As String
Dim File3          As String
```

```
strResult = ""
strDisplay = ""
```

```
File1 = "c:\win95\system.dat"
File2 = "system.uuencoded"
File3 = "system.uudecoded"
```

```
strDisplay = strDisplay & "File UUencode '" & File1 & "' to '" & File2 & "' is " & cFileUUCP(File1, File2,
MODE_UUENCODE, True) & vbCrLf
strDisplay = strDisplay & "File UUdecode '" & File2 & "' to '" & File3 & "' is " & cFileUUCP(File2, File3,
MODE_UUDECODE, True) & vbCrLf
strDisplay = strDisplay & "Compare File contents (not sensitive) '" & File1 & "' with '" & File3 & "' is " &
If(cCmpFileContents(File1, File3, False) = -1, "same", "not same") & vbCrLf & vbCrLf
```

```
File1 = "c:\autoexec.bat"
File2 = "autoexec.uuencoded"
File3 = "autoexec.uudecoded"
```

```
strDisplay = strDisplay & "File UUencode '" & File1 & "' to '" & File2 & "' is " & cFileUUCP(File1, File2,
MODE_UUENCODE, True) & vbCrLf
```

```
strDisplay = strDisplay & "File UUdecode " & File2 & " to " & File3 & " is " & cFileUUCP(File2, File3,  
MODE_UUDECODE, True) & vbCrLf  
strDisplay = strDisplay & "Compare File contents (not sensitive) " & File1 & " with " & File3 & " is " &  
If(cCmpFileContents(File1, File3, False) = -1, "same", "not same") & vbCrLf & vbCrLf
```

```
Debug.Print strDisplay
```

**See also :** [UUCP](#)

## File I/O from C : Overview

<a href="#">Fopen</a>	open a file for I/O.
<a href="#">Fclose</a>	close an open stream.
<a href="#">Fgetc</a>	read a single character from a stream.
<a href="#">Fputc</a>	write a single character to a stream.
<a href="#">Fputs</a>	write a line of characters to a stream.
<a href="#">Fgets</a>	read a line of characters from a stream.
<a href="#">Fwrite</a>	write an arbitrary number of characters to a stream.
<a href="#">Fread</a>	read an arbitrary number of characters from a stream.
<a href="#">Fcloseall</a>	close all files opened with fopen.
<a href="#">Fflush</a>	flush buffered I/O to a particular stream to disk.
<a href="#">Fflushall</a>	flush buffered I/O for all open streams to disk.
<a href="#">FEOF</a>	test for end-of-file on a stream.
<a href="#">Ferror</a>	test for an error on a stream.
<a href="#">Fclearerr</a>	reset the error indicator for a stream.
<a href="#">Fseek</a>	move the file pointer to a specified location.
<a href="#">Ftell</a>	get the current position of a file pointer.
<a href="#">Frewind</a>	move the file pointer to the beginning of a file.
<a href="#">FProcessAsciiFile</a>	read the offset of each line from an ASCII file (CR/LF line terminated) into an array.
<a href="#">FGotoRecord</a>	move the file pointer to the beginning of the specified line in an ASCII file (CR/LF line terminated).
<a href="#">FStdIn</a>	returns the stream of the standard input.
<a href="#">FStdOut</a>	returns the stream of the standard output.
<a href="#">FStdErr</a>	returns the stream of the standard error.

# FileIO

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**Fopen** open a file for I/O.  
**Fclose** close an open stream.  
**Fgetc** read a single character from a stream.  
**Fputc** write a single character to a stream.  
**Fputs** write a line of characters to a stream.  
**Fgets** read a line of characters from a stream.  
**Fwrite** write an arbitrary number of characters to a stream.  
**Fread** read an arbitrary number of characters from a stream.  
**Fcloseall** close all files opened with fopen.  
**Flush** flush buffered I/O to a particular stream to disk.  
**Fflushall** flush buffered I/O for all open streams to disk.  
**Feof** test for end-of-file on a stream.  
**Error** test for an error on a stream.  
**Fclearerr** reset the error indicator for a stream.  
**Fseek** move the file pointer to a specified location.  
**Ftell** get the current position of a file pointer.  
**Frewind** move the file pointer to the beginning of a file.  
**FProcessAsciiFile** read the offset of each line from an ASCII file (CR/LF line terminated) into an array.  
**FGotoRecord** move the file pointer to the beginning of the specified line in an ASCII file (CR/LF line terminated).  
**FStdIn** returns the stream of the standard input.  
**FStdOut** returns the stream of the standard output.  
**FStdErr** returns the stream of the standard error.

## Declare Syntax :

```
Declare Function cFopen Lib "time2win.dll" (ByVal File As String, ByVal Mode As String) As Long
Declare Function cFclose Lib "time2win.dll" (ByVal IOstream As Long) As Integer
Declare Function cFgetc Lib "time2win.dll" (ByVal IOstream As Long) As Integer
Declare Function cFputc Lib "time2win.dll" (ByVal char As Integer, ByVal IOstream As Long) As Integer
Declare Function cFputs Lib "time2win.dll" (ByVal Txt As String, ByVal IOstream As Long) As Integer
Declare Function cFgets Lib "time2win.dll" (Txt As String, ByVal Length As Integer, ByVal IOstream As Long) As Integer
Declare Function cFwrite Lib "time2win.dll" (Txt As String, ByVal IOstream As Long) As Integer
Declare Function cFread Lib "time2win.dll" (Txt As String, ByVal Length As Integer, ByVal IOstream As Long) As Integer
Declare Function cFcloseall Lib "time2win.dll" () As Integer
Declare Function cFlush Lib "time2win.dll" (ByVal IOstream As Long) As Integer
Declare Function cFflushall Lib "time2win.dll" () As Integer
Declare Function cFeof Lib "time2win.dll" (ByVal IOstream As Long) As Integer
Declare Function cFerror Lib "time2win.dll" (ByVal IOstream As Long) As Integer
Declare Sub cFclearerr Lib "time2win.dll" (ByVal IOstream As Long)
Declare Function cFseek Lib "time2win.dll" (ByVal IOstream As Long, ByVal offset As Long, ByVal Origin As Integer) As Integer
Declare Function cFtell Lib "time2win.dll" (ByVal IOstream As Long) As Long
Declare Sub cFrewind Lib "time2win.dll" (ByVal IOstream As Long)
Declare Function cFProcessAsciiFile Lib "time2win.dll" (ByVal IOstream As Long, AsciiOffset() As Long) As Long
Declare Function cFGotoRecord Lib "time2win.dll" (ByVal IOstream As Long, AsciiOffset() As Long, ByVal Record As Long) As Integer
Declare Function cFStdIn Lib "time2win.dll" () As Long
Declare Function cFStdOut Lib "time2win.dll" () As Long
Declare Function cFStdErr Lib "time2win.dll" () As Long
```

## Call Syntax :

see above

**Where :**

File\$	the name to use for streaming.
Mode\$	the open mode for the file (see comments).
IStream&	the returned stream or the stream to use to perform file management.
Char%	the char to write/read in decimal.
Txt\$	the string to write/read.
Length%	the length to read a string.
Offset&	the new seek position in the stream.
Origin% declaration)	the seeking method (see definition for file I/O in Constants and Types

**Comments :**

Code returned by these routines :

Fopen	>= 0	: I/O stream in a long integer.
Fclose	= 0 < 0	: all is OK, : error.
Fgetc	>= 0 < 0	: the char readed, : error.
Fputc	>= 0 < 0	: the char writed, : error.
Fputs	>= 0 < 0	: all is OK, : error.
Fgets	= 0 < 0	: all is OK, : error.
Fwrite	>= 0 < 0	: all is OK, : error.
Fread	>= 0 < 0	: all is OK, : error.
Fcloseall	= 0 < 0	: all is OK, : error.
Fflush	= 0 < 0	: all is OK, : error.
Fflushall	= 0 < 0	: all is OK : error.
Feof	= 0 = -1	: not EOF, : EOF.
Ferror	= 0 <>0	: no error, : error number.
Fseek	= 0 < 0	: all is OK, : error.
Ftell	>= 0 < 0	: the pointer position, : error.
FProcessAsciiFile	> 0	: the number of lines in the ASCII file (CR/LF terminated),



16384 characters),	= 0	: error : can't allocate memory buffer (each line can't be longer than
	< 0	: error.
FGotoRecord	= -1	: all is ok,
	= 0	: record is outside of the limits of the array,
	< 0	: error.
FStdIn, FStdOut, FStdErr	>= 0	: I/O stream in a long integer. <span style="color: red;">(only available in TIME TO WIN 32-Bit)</span>

The character string mode specifies the type of access requested for the file, as follows:

"r"	Opens for reading. If the file does not exist or cannot be found, the fopen call will fail.
"w"	Opens an empty file for writing. If the given file exists, its contents are destroyed.
"a"	Opens for writing at the end of the file (appending); creates the file first if it doesn't exist.
"r+"	Opens for both reading and writing. (The file must exist.)
"w+"	Opens an empty file for both reading and writing. If the given file exists, its contents are destroyed.
"a+"	Opens for reading and appending; creates the file first if it doesn't exist.

When a file is opened with the "a" or "a+" access type, all write operations occur at the end of the file. Although the file pointer can be repositioned using *cFseek* or *cFrewind*, the file pointer is always moved back to the end of the file before any write operation is carried out. Thus, existing data cannot be overwritten.

When the "r+", "w+", or "a+" access type is specified, both reading and writing are allowed (the file is said to be open for "update"). However, when you switch between reading and writing, there must be an intervening *cFlush*, *cFseek*, or *cFrewind* operation. The current position can be specified for the *cFseek* operation, if desired. In addition to the values listed above, the following characters can be included in mode to specify the translation mode for newline characters:

"t"

Open in text (translated) mode. In this mode, carriage-return-line-feed (CR-LF) combinations are translated into single line feeds (LF) on input and LF characters are translated to CR-LF combinations on output. Also, CTRL+Z is interpreted as an end-of-file character on input. In files opened for reading or for reading/writing, *cFopen* checks for a CTRL+Z at the end of the file and removes it, if possible. This is done because using the *cFseek* and *cFtell* functions to move within a file that ends with a CTRL+Z may cause *cFseek* to behave improperly near the end of the file.

"b"

Open in binary (untranslated) mode; the above translations are suppressed.

#### Examples :

see FileIO.MAK

#### See also :

# HugeStrAdd

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[HugeStrAdd](#) add a VB string into a Huge String.

## Declare Syntax :

Declare Function cHugeStrAdd Lib "[time2win.dll](#)" (ByVal hsHandle As Long, hsText As String) As Integer

## Call Syntax :

hsReturn% = cHugeStrAdd(hsHandle%, hsText\$)

## Where :

hsHandle% is the Handle for all functions for Huge String.  
hsText\$ is the VB string to add into the Huge String.  
hsReturn% TRUE : if all is ok  
FALSE : if length of the VB string is 0, or if the VB string can't be fitted into the Huge String.

## Comments :

The length of hsText must be between 1 and 64,000 chars.  
The position of hsText into the Huge String is depending of the Write Pointer.  
If you don't set manually the Write Pointer, the VB String is always appended to previous chars.

## Examples :

```
Dim hsHandle As Integer
Dim hsSize As Long
Dim hsReturn As Integer
Dim hsLength As Long
```

```
hsSize = 512 & * 1024
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

## See also :

# Interest rate

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**AtoF** : annuity to future value.  
**AtoFC** : annuity to future value continuous compounding.  
**AtoP** : annuity to present value.  
**AtoPC** : annuity to present value continuous compounding.  
**FtoA** : future value to annuity.  
**FtoAC** : future value to annuity continuous compounding.  
**FtoP** : future value to present value.  
**FtoPC** : future value to present value continuous compounding.  
**PtoA** : present value to annuity.  
**PtoAC** : present value to annuity continuous compounding.  
**PtoF** : present value to future value.  
**PtoFC** : present value to future value continuous compounding.

## Declare Syntax :

Declare Function cAtoF Lib "time2win.dll" (ByVal Interest As Double, ByVal N As Integer) As Double  
Declare Function cAtoFC Lib "time2win.dll" (ByVal Rates As Double, ByVal N As Integer) As Double  
Declare Function cAtoP Lib "time2win.dll" (ByVal Interest As Double, ByVal N As Integer) As Double  
Declare Function cAtoPC Lib "time2win.dll" (ByVal Rates As Double, ByVal N As Integer) As Double  
Declare Function cFtoA Lib "time2win.dll" (ByVal Interest As Double, ByVal N As Integer) As Double  
Declare Function cFtoAC Lib "time2win.dll" (ByVal Rates As Double, ByVal N As Integer) As Double  
Declare Function cFtoP Lib "time2win.dll" (ByVal Interest As Double, ByVal N As Integer) As Double  
Declare Function cFtoPC Lib "time2win.dll" (ByVal Rates As Double, ByVal N As Integer) As Double  
Declare Function cPtoA Lib "time2win.dll" (ByVal Interest As Double, ByVal N As Integer) As Double  
Declare Function cPtoAC Lib "time2win.dll" (ByVal Rates As Double, ByVal N As Integer) As Double  
Declare Function cPtoF Lib "time2win.dll" (ByVal Interest As Double, ByVal N As Integer) As Double  
Declare Function cPtoFC Lib "time2win.dll" (ByVal Rates As Double, ByVal N As Integer) As Double

## Call Syntax :

## Where :

In all functions, N is the number of periods.

**AtoF** : Interest is the effective interest rate per period.  
**AtoFC** : Interest is the nominal interest rate per period.  
**AtoP** : Interest is effective interest rate per period.  
**AtoPC** : Interest is the nominal interest rate per period.  
**FtoA** : Interest is the effective interest rate per period.  
**FtoAC** : Interest is the nominal interest rate per period.  
**FtoP** : Interest is the effective interest rate per period.  
**FtoPC** : Interest is the nominal interest rate per period.  
**PtoA** : Interest is the effective interest rate per period.  
**PtoAC** : Interest is the nominal interest rate per period.  
**PtoF** : Interest is the effective interest rate per period.  
**PtoFC** : Interest is the nominal interest rate per period.

## Comments :

If Interest is 0 or N is below or equal to 0, the returned value is -1.

## Examples :

**See also :**

## Interest rate : Overview

- [AtoF](#) : annuity to future value.
- [AtoFC](#) : annuity to future value continuous compounding.
- [AtoP](#) : annuity to present value.
- [AtoPC](#) : annuity to present value continuous compounding.
- [FtoA](#) : future value to annuity.
- [FtoAC](#) : future value to annuity continuous compounding.
- [FtoP](#) : future value to present value.
- [FtoPC](#) : future value to present value continuous compounding.
- [PtoA](#) : present value to annuity.
- [PtoAC](#) : present value to annuity continuous compounding.
- [PtoF](#) : present value to future value.
- [PtoFC](#) : present value to future value continuous compounding.

# GetIni

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetIni](#) retrieve an item in a section of an INI file.

## Declare Syntax :

Declare Function cGetIni Lib "time2win.dll" (ByVal AppName As String, ByVal szItem As String, ByVal szDefault As String, ByVal InitFile As String) As String

## Call Syntax :

```
test$ = cGetIni(AppName, szItem, szDefault, InitFile)
```

## Where :

AppName	a string that specifies the section containing the entry.
szItem	a string containing the entry whose associated string is to be retrieved.
szDefault	a string that specifies the default value for the given entry if the entry cannot be found in the initialization file.
InitFile	a filename. If this parameter does not contain a full path, Windows searches for the file in the Windows directory.

## Comments :

The function searches the file for an entry that matches the name specified by the szItem parameter under the section heading specified by the AppName parameter. If the entry is found, its corresponding string is returned. If the entry does not exist, the default character string specified by the szDefault parameter is copied. A string entry in the initialization file must have the following form:

```
[section]  
entry=string
```

## Examples :

```
test$ = cGetIni("Desktop","IconTitleFaceName","MS Sans Serif","WIN.INI")
```

**See also :** [Windows](#)

# HugeStrAddress

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[HugeStrAddress](#) return the memory address of a Huge String.

## Declare Syntax :

Declare Function cHugeStrAddress Lib "time2win.dll" (ByVal hsHandle As Long) As Long

## Call Syntax :

hsAddress& = cHugeStrLength(hsHandle%)

## Where :

hsHandle%        is the Handle for all functions for Huge String.  
hsAddress&      is the memory address of the Huge String.

## Comments :

## Examples :

```
Dim hsHandle        As Integer
Dim hsSize          As Long
Dim hsReturn        As Integer
Dim hsAddress       As Long
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsAddress = cHugeStrAddress(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had an address of " & hsAddress
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

## See also :

# HugeStrAppend

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 \(Win95/WinNT\)](#), [MSOffice 95](#)

## Purpose :

[HugeStrAppend](#) append a VB string into a Huge String.

## Declare Syntax :

Declare Function cHugeStrAppend Lib "[time2win.dll](#)" (ByVal hsHandle As Long, hsText As String) As Integer

## Call Syntax :

hsReturn% = cHugeStrAppend(hsHandle%, hsText\$)

## Where :

hsHandle% is the Handle for all functions for Huge String.  
hsText\$ is the VB string to append into the Huge String.  
hsReturn% TRUE : if all is ok  
FALSE : if length of the VB string is 0, or if the VB string can't be fitted into the Huge String.

## Comments :

The length of hsText must be between 1 and 64,000 chars.  
The position of hsText into the Huge String is NOT depending of the Write Pointer. The VB string is appended without regards and whitout change of the Write Pointer.

## Examples :

```
Dim hsHandle As Integer
Dim hsSize As Long
Dim hsReturn As Integer
Dim hsLength As Long
```

```
hsSize = 512 & * 1024
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
hsReturn = cHugeStrSetWP(hsHandle, 10)
hsReturn = cHugeStrAppend(hsHandle, ", No price change.")
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```



**See also :**

# HugeStrBlocks

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**HugeStrBlocks** return the number of blocks of 64,000 chars into a Huge String.

## Declare Syntax :

Declare Function cHugeStrBlocks Lib "time2win.dll" (ByVal hsHandle As Long) As Long

## Call Syntax :

hsBlocks& = cHugeStrBlocks(hsHandle%)

## Where :

hsHandle%        is the Handle for all functions for Huge String.  
hsBlocks&        is the number of blocks of 64,000 chars.

## Comments :

If the size of a Huge String is a multiple of 64,000, the returned blocks will be always the quotient of the division.  
If the size of a Huge String is not a multiple of 64,000, the returned blocks will be the quotient of the division plus one.

## Examples :

```
Dim hsHandle        As Integer
Dim hsSize         As Long
Dim hsReturn       As Integer
Dim hsBlocks       As Long
```

```
hsSize = 512 & * 1024
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
  MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
  MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, String$(64000, "A"))
hsReturn = cHugeStrAdd(hsHandle, String$(64000, "B"))
hsReturn = cHugeStrAdd(hsHandle, String$(32000, "C"))
```

```
hsBlocks = cHugeStrBlocks(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had " & hsBlocks & " blocks"
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
  MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
  MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

## See also :

# HugeStrClear

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[HugeStrClear](#) clear the contents of a Huge String.

## Declare Syntax :

Declare Function cHugeStrClear Lib "time2win.dll" (ByVal hsHandle As Long) As Integer

## Call Syntax :

hsReturn% = cHugeStrClear(hsHandle%)

## Where :

hsHandle%      is the Handle for all functions for Huge String.  
hsReturn%      is the returned code,  
                 TRUE    : the Huge String has been cleared.  
                 FALSE : the Huge String can't be cleared.

## Comments :

## Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
  MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
  MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If
```

```
hsReturn = cHugeStrClear(hsHandle)
```

```
If (hsReturn = TRUE) Then
  MsgBox "Huge String (" & hsHandle & ") has been cleared."
Else
  MsgBox "Huge String (" & hsHandle & ") can't be cleared."
End If
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
  MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
  MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

## See also :

# HugeStrCreate

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**HugeStrCreate** create and reserve enough memory space for the required Huge String.

## Declare Syntax :

Declare Function cHugeStrCreate Lib "time2win.dll" (ByVal hsSize As Long) As Integer

## Call Syntax :

hsHandle% = cHugeStrCreate(hsSize&)

## Where :

hsSize&            is the size for the Huge String (TIME2WIN add 12 bytes for header).  
hsHandle%        is the Handle for all functions for Huge String.

## Comments :

The Handle can be '0' if the Huge String can't be created. In this case, you can't use any functions for Huge String.

## Examples :

```
Dim hsHandle            As Integer  
Dim hsSize             As Long
```

```
hsSize = 512& * 1024  
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then  
  MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"  
Else  
  MsgBox "Huge String of " & hsSize & " bytes can't be created."  
End If
```

## See also :

# HugeStrFree

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[HugeStrFree](#) free a Huge String created with [cHugeStrCreate](#).

## Declare Syntax :

Declare Function cHugeStrFree Lib "time2win.dll" (ByVal hsHandle As Long) As Integer

## Call Syntax :

hsReturn% = cHugeStrFree(hsHandle%)

## Where :

hsHandle%      is a handle returned by the cHugeStrCreate function.  
hsReturn%      is the returned code,  
                 TRUE    : the Huge String has been destroyed.  
                 FALSE  : the Huge String can't be destroyed.

## Comments :

In the case of the Huge String can't be destroyed, the memory will be restablish when 'TIME TO WIN (32-Bit)' will be unloaded.

## Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

## See also :

# HugeStrGetNP

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[HugeStrGetNP](#) return the Next Pointer of a Huge String.

## Declare Syntax :

Declare Function cHugeStrGetNP Lib "[time2win.dll](#)" (ByVal hsHandle As Long) As Long

## Call Syntax :

hsPtr& = cHugeStrGetNP(hsHandle%)

## Where :

hsHandle%        is the Handle for all functions for Huge String.  
hsPtr&            is the readed Next Pointer.

## Comments :

## Examples :

```
Dim hsHandle        As Integer
Dim hsSize         As Long
Dim hsReturn       As Integer
Dim hsLength       As Long
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsReturn = cHugeStrSetWP(hsHandle, 9)
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
MsgBox "The contents of the next 11 chars is " & cHugeStrNext(hsHandle, 11)
```

```
MsgBox "The Next Pointer is " & cHugeStrGetNP(hsHandle)
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

## See also :



# HugeStrGetWP

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[HugeStrGetWP](#) return the Write Pointer of a Huge String.

## Declare Syntax :

Declare Function cHugeStrGetWP Lib "[time2win.dll](#)" (ByVal hsHandle As Long) As Long

## Call Syntax :

```
hsPtr& = cHugeStrGetWP(hsHandle%)
```

## Where :

hsHandle%        is the Handle for all functions for Huge String.  
hsPtr&            is the readed Write Pointer.

## Comments :

## Examples :

```
Dim hsHandle        As Integer  
Dim hsSize         As Long  
Dim hsReturn       As Integer  
Dim hsLength       As Long
```

```
hsSize = 512& * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsReturn = cHugeStrSetWP(hsHandle, 9)
```

```
hsReturn = cHugeStrAdd(hsHandle, "time to win")
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
MsgBox "The contents of the first block is " & cHugeStrRead(hsHandle, 1)
```

```
MsgBox "The Write Pointer is " & cHugeStrGetWP(hsHandle)
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```



**See also :**

# HugeStrLength

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**HugeStrLength** return the length of used chars in a Huge String.

## Declare Syntax :

Declare Function cHugeStrLength Lib "time2win.dll" (ByVal hsHandle As Long) As Long

## Call Syntax :

hsLength% = cHugeStrLength(hsHandle%)

## Where :

hsHandle%        is the Handle for all functions for Huge String.  
hsLength%        is the length of used chars.

## Comments :

## Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

## See also :

# HugeStrMid

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[HugeStrMid](#) return the X chars from a position from a Huge String.

## Declare Syntax :

Declare Function cHugeStrMid Lib "[time2win.dll](#)" (ByVal hsHandle As Long, ByVal hsStart As Long, ByVal hsLength As Long) As String

## Call Syntax :

```
hsText$ = cHugeStrMid(hsHandle%, hsStart&, hsLength&)
```

## Where :

hsHandle% is the Handle for all functions for Huge String.  
hsStart& is the starting position (1 to Length of the Huge String).  
hsLength& is the length of the desired string (1 to 64,000).  
hsText\$ is the readed string.

## Comments :

## Examples :

```
Dim hsHandle As Integer
Dim hsSize As Long
Dim hsReturn As Integer
Dim hsLength As Long
```

```
hsSize = 512& * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
MsgBox "The contents of the 11 chars from the position 9 is " & cHugeStrMid(hsHandle, 9, 11)
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

## See also :



# HugeStrNext

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[HugeStrNext](#) return the X next chars from the Next Pointer in a Huge String.

## Declare Syntax :

Declare Function cHugeStrNext Lib "time2win.dll" (ByVal hsHandle As Long, ByVal hsNext As Long) As String

## Call Syntax :

```
hsText$ = cHugeStrNext(hsHandle%, hsNext&)
```

## Where :

hsHandle%	is the Handle for all functions for Huge String.
hsNext&	is the number of next chars to read (1 to 64,000).
hsText\$	is the readed string.

## Comments :

## Examples :

```
Dim hsHandle      As Integer
Dim hsSize        As Long
Dim hsReturn      As Integer
Dim hsLength      As Long
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsReturn = cHugeStrSetWP(hsHandle, 9)
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
MsgBox "The contents of the next 11 chars is " & cHugeStrNext(hsHandle, 11)
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

## See also :

# Object : Overview

<a href="#"><u>DisableCtlRedraw</u></a>	disable the redraw of a object (by Control).
<a href="#"><u>DisableFl</u></a>	disable mouse and keyboard input to the given form (by Object).
<a href="#"><u>DisableForm</u></a>	disable mouse and keyboard input to the given form (by hWnd).
<a href="#"><u>DisableRedraw</u></a>	disable the redraw of a object (by hWnd).
<a href="#"><u>EnableCtlRedraw</u></a>	enable the redraw of a object (by Control).
<a href="#"><u>EnableFl</u></a>	enable mouse and keyboard input to the given form (by Object).
<a href="#"><u>EnableForm</u></a>	enable mouse and keyboard input to the given form (by hWnd).
<a href="#"><u>EnableRedraw</u></a>	enable the redraw of a object (by hWnd).
<a href="#"><u>GetCtlCaption</u></a>	return the .Caption property.
<a href="#"><u>GetCtlClass</u></a>	return the class name defined in the properties windows in the design-mode of VB.
<a href="#"><u>GetCtlContainer</u></a>	return the name of the container did contains the control.
<a href="#"><u>GetCtlDataField</u></a>	return the .DataField property.
<a href="#"><u>GetCtlForm</u></a>	return the name of the form did contains the control.
<a href="#"><u>GetCtlIndex</u></a>	return the .Index property.
<a href="#"><u>GetCtlName</u></a>	return the .Name of the control.
<a href="#"><u>GetCtlNameIndex</u></a>	return the name and the of the control.
<a href="#"><u>GetCtlPropCaption</u></a>	return the position of the .Caption property in the definition table of the control.
<a href="#"><u>GetCtlPropDataField</u></a>	return the position of the .DataField property in the definition table of the control.
<a href="#"><u>GetCtlPropText</u></a>	return the position of the .Text property in the definition table of the control.
<a href="#"><u>GetCtlTag</u></a>	return the .Tag property of the control.
<a href="#"><u>GetCtlTagSized</u></a>	return the full .Tag property of the control.
<a href="#"><u>GetCtlText</u></a>	return the .Text property of the control.
<a href="#"><u>GetHwnd</u></a>	return the .hwnd of the control.
<a href="#"><u>GetObjCaption</u></a>	
<a href="#"><u>GetObjClassName</u></a>	
<a href="#"><u>GetObjContainer</u></a>	
<a href="#"><u>GetObjDataField</u></a>	
<a href="#"><u>GetObjDataSource</u></a>	
<a href="#"><u>GetObjIndex</u></a>	
<a href="#"><u>GetObjName</u></a>	
<a href="#"><u>GetObjNameIndex</u></a>	
<a href="#"><u>GetObjParent</u></a>	
<a href="#"><u>GetObjTag</u></a>	
<a href="#"><u>GetObjText</u></a>	
<a href="#"><u>ObjDisableRedraw</u></a>	disable the redraw of a object (by Object).
<a href="#"><u>ObjEnableRedraw</u></a>	enable the redraw of a object (by Object).
<a href="#"><u>ObjectGetBoolean</u></a>	
<a href="#"><u>ObjectGetByte</u></a>	
<a href="#"><u>ObjectGetIndex</u></a>	
<a href="#"><u>ObjectGetInteger</u></a>	
<a href="#"><u>ObjectGetLong</u></a>	
<a href="#"><u>ObjectGetPropertyByName</u></a>	read data in properties (by name) from OCX custom controls.
<a href="#"><u>ObjectGetPropertyByPos</u></a>	read data in properties (by position) from OCX custom controls.
<a href="#"><u>ObjectGetString</u></a>	
<a href="#"><u>ObjectGetStringW</u></a>	
<a href="#"><u>ObjectGetVariant</u></a>	
<a href="#"><u>ObjectMethod</u></a>	
<a href="#"><u>ObjectMethodByName</u></a>	give the access of method (by name) of OCX custom controls.
<a href="#"><u>ObjectMethodByPos</u></a>	give the access of method (by position) of OCX custom controls.
<a href="#"><u>ObjectPutBoolean</u></a>	
<a href="#"><u>ObjectPutByte</u></a>	
<a href="#"><u>ObjectPutInteger</u></a>	
<a href="#"><u>ObjectPutLong</u></a>	
<a href="#"><u>ObjectPutPropertyByName</u></a>	write data in properties (by name) from OCX custom controls.
<a href="#"><u>ObjectPutPropertyByPos</u></a>	write data in properties (by position) in OCX custom controls.
<a href="#"><u>ObjectPutString</u></a>	
<a href="#"><u>ObjectPutVariant</u></a>	
<a href="#"><u>PutObjCaption</u></a>	
<a href="#"><u>PutObjDataField</u></a>	

PutObjDataSource

PutObjTag

PutObjText

SetCtlCaption

SetCtlDataField

SetCtlFocus

SetCtlPropString

control.

SetCtlTag

SetCtlText

set the .Caption property of the control.

set the .DataField property of the control.

give the Focus to a control.

set the specified property (founded with cGetCtlPropString function) of the

set the .Tag property of the control.

set the .Text property of the control.

# HugeStrOnDisk

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

`HugeStrOnDisk` read/write a Huge String from/to a file.

## Declare Syntax :

Declare Function `cHugeStrOnDisk` Lib "time2win.dll" (ByVal `hsHandle` As Long, ByVal `hsFile` As String, ByVal `hsGetPut` As Integer) As Long

## Call Syntax :

`hsFileLength& = cHugeStrOnDisk(hsHandle%, hsFile$, hsGetPut%)`

## Where :

`hsHandle%` is the Handle for all functions for Huge String.  
`hsFile$` is the name of the file to read/write the Huge String.  
`hsGetPut%` PUT\_ARRAY\_ON\_DISK to put the array on disk,  
GET\_ARRAY\_ON\_DISK to get the array from disk.  
`hsFileLength&` >=0 is the returned length of the file,  
< 0 is an error occurs (error n° is the negative value of all DA\_x values, see Constants and Types declaration ).

## Comments :

The file length is the size of the Huge String plus the 12 bytes header.

## Examples :

```
Dim hsHandle          As Integer
Dim hsSize            As Long
Dim hsReturn          As Integer
Dim hsLength          As Long
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
MsgBox "The length of the saved file is " & cHugeStrOnDisk(hsHandle, "c:\hugestr.tmp", PUT_ARRAY_ON_DISK)
```

```
hsReturn = cHugeStrClear(hsHandle)
```

```
MsgBox "The length of the readed file is " & cHugeStrOnDisk(hsHandle, "c:\hugestr.tmp", GET_ARRAY_ON_DISK)
```

```
hsReturn = cHugeStrFree(hsHandle)
```



```
If (hsReturn = TRUE) Then
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

**See also :**

# HugeStrRead

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[HugeStrRead](#) read a block of 64,000 chars or a part of block in a Huge String.

## Declare Syntax :

Declare Function cHugeStrRead Lib "time2win.dll" (ByVal hsHandle As Long, ByVal hsBlock As Long) As String

## Call Syntax :

```
hsText$ = cHugeStrRead(hsHandle%, hsBlock&)
```

## Where :

hsHandle% is the Handle for all functions for Huge String.  
hsBlock& is a block number for reading into Huge String (must be between 1 and cHugeStrBlocks).  
hsText\$ is the returned string (maximum 64,000 chars).

## Comments :

The length of hsText will be between 0 and 64,000 chars.

## Examples :

```
Dim hsHandle As Integer
Dim hsSize As Long
Dim hsReturn As Integer
Dim hsLength As Long
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
MsgBox "The contents of the first block is " & cHugeStrRead(hsHandle, 1)
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

## See also :

# HugeStrSetNP

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[HugeStrSetNP](#) set the Next Pointer of a Huge String.

## Declare Syntax :

Declare Function cHugeStrSetNP Lib "time2win.dll" (ByVal hsHandle As Long, ByVal hsPtr As Long) As Integer

## Call Syntax :

hsReturn% = cHugeStrSetNP(hsHandle% , hsPtr&)

## Where :

hsHandle%	is the Handle for all functions for Huge String.
hsPtr&	is the new Next Pointer.
hsReturn%	TRUE : if all is ok FALSE : if hsPtr is <=0 or > Length of the Huge String.

## Comments :

## Examples :

```
Dim hsHandle      As Integer
Dim hsSize        As Long
Dim hsReturn      As Integer
Dim hsLength      As Long
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsReturn = cHugeStrSetNP(hsHandle, 9)
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
MsgBox "The contents of the next 11 chars is " & cHugeStrNext(hsHandle, 11)
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

## See also :



# HugeStrSetWP

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**HugeStrSetWP** set the Write Pointer into a Huge String.

## Declare Syntax :

Declare Function cHugeStrSetWP Lib "time2win.dll" (ByVal hsHandle As Long, ByVal hsPtr As Long) As Integer

## Call Syntax :

hsReturn% = cHugeStrSetWP(hsHandle%, hsPtr&)

## Where :

hsHandle% is the Handle for all functions for Huge String.  
hsPtr& is the new Write Pointer.  
hsReturn% TRUE : if all is ok  
FALSE : if hsPtr is <=0 or > Length of the Huge String.

## Comments :

## Examples :

```
Dim hsHandle As Integer
Dim hsSize As Long
Dim hsReturn As Integer
Dim hsLength As Long
```

```
hsSize = 512 & * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
```

```
hsReturn = cHugeStrSetWP(hsHandle, 9)
```

```
hsReturn = cHugeStrAdd(hsHandle, "time to win")
```

```
hsLength = cHugeStrLength(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength
```

```
MsgBox "The contents of the first block is " & cHugeStrRead(hsHandle, 1)
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

**See also :**

# HugeStrSize

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[HugeStrSize](#) return the size of a Huge String.

## Declare Syntax :

Declare Function cHugeStrSize Lib "[time2win.dll](#)" (ByVal hsHandle As Long) As Long

## Call Syntax :

hsReadSize& = cHugeStrSize(hsHandle%)

## Where :

hsHandle% is a handle returned by the cHugeStrCreate function.  
hsReadSize& is the size of the Huge String.

## Comments :

The returned size is the size specified in the cHugeStrCreate function.

## Examples :

```
Dim hsHandle      As Integer
Dim hsSize        As Long
Dim hsReturn      As Integer
Dim hsReadSize    As Long
```

```
hsSize = 512& * 1024
```

```
hsHandle = cHugeStrCreate(hsSize)
```

```
If (hsHandle <> 0) Then
```

```
    MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
```

```
Else
```

```
    MsgBox "Huge String of " & hsSize & " bytes can't be created."
```

```
End If
```

```
hsReadSize = cHugeStrSize(hsHandle)
```

```
MsgBox "Huge String (" & hsHandle & ") had a size of " & hsReadSize
```

```
hsReturn = cHugeStrFree(hsHandle)
```

```
If (hsReturn = TRUE) Then
```

```
    MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
```

```
    MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
```

```
End If
```

## See also :

# Huge string : Overview

The functions/subs used in the Huge String routines handle Huge String. Huge String is a string from 1 to 16,711,680 chars.

An bigger advantage of Huge String is the speed.

The functions for adding or appending chars in a Huge String is faster than VB equivalent (20 times faster).

The maximum number of Huge String is 8192.

This number is a theoretical maximum and is depending of any application loaded in memory.

The following functions/subs are used to handle big sized arrays on disk :

<u>HugeStrAdd</u>	add a VB string into a Huge String.
<u>HugeStrAddress</u>	return a pointer for the first char of a Huge String.
<u>HugeStrAppend</u>	append a VB string into a Huge String.
<u>HugeStrBlocks</u>	return the number of block of 64,000 chars from a Huge String.
<u>HugeStrClear</u>	clear a Huge String.
<u>HugeStrCreate</u>	create a Huge String.
<u>HugeStrFree</u>	free a Huge String (destroy it).
<u>HugeStrGetNP</u>	get the Next Pointer of a Huge String.
<u>HugeStrGetWP</u>	get the Write Pointer of a Huge String.
<u>HugeStrLength</u>	return the length of data in a Huge String.
<u>HugeStrMid</u>	extract a VB sub-string from a Huge String.
<u>HugeStrNext</u>	read the next part of a Huge String.
<u>HugeStrOnDisk</u>	get/put a Huge String from/to a file on disk.
<u>HugeStrRead</u>	read a block of 64,000 chars or minder from a Huge String.
<u>HugeStrSetNP</u>	set the Next Pointer of a Huge String.
<u>HugeStrSetWP</u>	set the Write Pointer of a Huge String.
<u>HugeStrSize</u>	return the full size of a Huge String.

Don't forget that any Huge String must be destroyed before quitting the application.

If you not destroy all Huge String that you've created, the memory used will be only released when [TIME2WIN.DLL](#) will be unloaded from memory.



# SetCtl.X

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

The functions below applies to a custom control.

[SetCtlCaption](#) set the .Caption property of the control.

[SetCtlDataField](#) set the .DataField property of the control.

[SetCtlFocus](#) give the Focus to a control.

[SetCtlPropString](#) set the specified property (founded with cGetCtlPropString function) of the control.

[SetCtlTag](#) set the .Tag property of the control.

[SetCtlText](#) set the .Text property of the control.

## Declare Syntax :

```
Declare Sub cSetCtlCaption Lib "time2win.dll" (Obj As Object, ByVal Text As String)
```

```
Declare Sub cSetCtlDataField Lib "time2win.dll" (Obj As Object, ByVal Text As String)
```

```
Declare Sub cSetCtlFocus Lib "time2win.dll" (Obj As Object)
```

```
Declare Sub cSetCtlPropString Lib "time2win.dll" (Obj As Object, ByVal PropIndex As Integer, ByVal Text As String)
```

```
Declare Sub cSetCtlTag Lib "time2win.dll" (Obj As Object, ByVal Text As String)
```

```
Declare Sub cSetCtlText Lib "time2win.dll" (Obj As Object, ByVal Text As String)
```

## Call Syntax :

The purpose and the declare syntax are very explicite.

## Where :

Obj                    the name of the object to proceed

## Comments :

The advantage to use these routines is that these routines doesn't generates an error if the property not exists.

## Examples :

**See also :** [Object](#)

# GetCtl.X

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

The functions below applies to a custom control.

[GetCtlCaption](#) return the .Caption property.

[GetCtlClass](#) return the class name defined in the properties windows in the design-mode of VB.

[GetCtlContainer](#) return the name of the container did contains the control. The container can be the form or an another control.

[GetCtlDataField](#) return the .DataField property.

[GetCtlForm](#) return the name of the form did contains the control.

[GetCtlIndex](#) return the .Index property. If the control has no index, -1 is returned.

[GetCtlName](#) return the .Name of the control.

[GetCtlNameIndex](#) return the name and the of the control. The format is Name(x), if no index => Name is used.

[GetCtlPropCaption](#) return the position of the .Caption property in the definition table of the control.

[GetCtlPropDataField](#) return the position of the .DataField property in the definition table of the control.

[GetCtlPropText](#) return the position of the .Text property in the definition table of the control.

[GetCtlTag](#) return the .Tag property of the control. The returned string is limited to the first chr\$(0) founded.

[GetCtlTagSized](#) return the full .Tag property of the control.

[GetCtlText](#) return the .Text property of the control.

[GetHwnd](#) return the .hwnd of the control. If the control has no .hwnd, the returned value is 0.

## Declare Syntax :

```
Declare Function cGetCtlCaption Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlClass Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlContainer Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlDataField Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlForm Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlIndex Lib "time2win.dll" (Obj As Object) As Integer
Declare Function cGetCtlName Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlNameIndex Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlPropCaption Lib "time2win.dll" (Obj As Object) As Integer
Declare Function cGetCtlPropDataField Lib "time2win.dll" (Obj As Object) As Integer
Declare Function cGetCtlPropText Lib "time2win.dll" (Obj As Object) As Integer
Declare Function cGetCtlTag Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlTagSized Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetCtlText Lib "time2win.dll" (Obj As Object) As String
Declare Function cGetHwnd Lib "time2win.dll" (Obj As Object) As Integer
```

## Call Syntax :

The purpose and the declare syntax are very explicite.

## Where :

Ctl                    the name of the control to proceed

## Comments :

The advantage to use these routines is that these routines doesn't generates an error if the property not exists.

## Examples :

**See also :** [Object](#)

# ObjectMethod, ObjectGetProperty, ObjectPutProperty

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[ObjectMethodByPos](#) give the access of method (by position) of OCX custom controls.

[ObjectMethodByName](#) give the access of method (by name) of OCX custom controls.

[ObjectGetPropertyByPos](#) read data in properties (by position) from OCX custom controls.

[ObjectGetPropertyByName](#) read data in properties (by name) from OCX custom controls.

[ObjectPutPropertyByPos](#) write data in properties (by position) in OCX custom controls.

[ObjectPutPropertyByName](#) write data in properties (by name) from OCX custom controls.

## Declare Syntax :

```
Declare Sub cObjectMethodByPos Lib "time2win.dll" (Obj As Object, ByVal Property As Integer, lpPut As Variant)
Declare Function cObjectGetPropertyByPos Lib "time2win.dll" (Obj As Object, ByVal Property As Integer) As Variant
Declare Sub cObjectPutPropertyByPos Lib "time2win.dll" (Obj As Object, ByVal Property As Integer, lpPut As Variant)
Declare Sub cObjectMethodByName Lib "time2win.dll" (Obj As Object, ByVal Property As String, lpPut As Variant)
Declare Function cObjectGetPropertyByName Lib "time2win.dll" (Obj As Object, ByVal Property As String) As Variant
Declare Sub cObjectPutPropertyByName Lib "time2win.dll" (Obj As Object, ByVal Property As String, lpPut As Variant)
```

## Call Syntax :

```
Call cObjectMethodByPos(Obj, Property%, varPut)
Call cObjectMethodByName(Obj, Property$, varPut)
varGet = cObjectGetPropertyByPos(Obj, Property%)
varGet = cObjectGetPropertyByName(Obj, Property$)
Call cObjectPutPropertyByPos(Obj, Property%, varPut)
Call cObjectPutPropertyByName(Obj, Property$, varPut)
```

## Where :

Obj	is a valid object (Form, OCX custom control, VBX custom control);
Property%	is a constant for accessing the data (see Constants and Types declaration);
Property\$	is a valid property;
varPut	is a data in a type variant;
varGet	is the returned data in a type variant.

## Comments :

For cObjectGetProperty?, if the property don't exist the returned variant is EMPTY

## Examples :

```
Dim varGet As Variant
```

```
Call cObjectPutPropertyByPos(Frame1, OBJ_CAPTION, "this is a test")
varGet = cObjectGetPropertyByPos(Frame1, OBJ_CAPTION) '---> this is a test
```

```
Call cObjectPutPropertyByName(Frame1, "caption", "this is an another test")
varGet = cObjectGetPropertyByName(Frame1, "caption") '---> this is an another test
```

```
Call cObjectMethodByName(List1, "clear", Empty)
```

## See also :

# ObjectGet.X

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

```
Declare Function cObjectGetBoolean Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String) As Boolean
Declare Function cObjectGetByte Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String) As Byte
Declare Function cObjectGetInteger Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String) As Integer
Declare Function cObjectGetLong Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String) As Long
Declare Function cObjectGetString Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String) As String
Declare Function cObjectGetStringW Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String) As String
Declare Function cObjectGetVariant Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String) As Variant
Declare Function cObjectGetIndex Lib "time2win.dll" (ByVal Obj As Object) As Integer
```

# GetObj.X

```
Declare Function cGetObjCaption Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjContainer Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjParent Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjTag Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjText Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjDataField Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjDataSource Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjName Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjIndex Lib "time2win.dll" (ByVal Obj As Object) As Integer
Declare Function cGetObjNameIndex Lib "time2win.dll" (ByVal Obj As Object) As String
Declare Function cGetObjClassName Lib "time2win.dll" (ByVal Obj As Object) As String
```

# ObjectPut.X

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

Declare Sub cObjectPutBoolean Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String, ByVal Value As Boolean)

Declare Sub cObjectPutByte Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String, ByVal Value As Byte)

Declare Sub cObjectPutInteger Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String, ByVal Value As Integer)

Declare Sub cObjectPutLong Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String, ByVal Value As Long)

Declare Sub cObjectPutString Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String, ByVal Value As String)

Declare Sub cObjectPutVariant Lib "time2win.dll" (ByVal Obj As Object, ByVal Property As String, ByVal Value As Variant)

# PutObj.X

```
Declare Sub cPutObjCaption Lib "time2win.dll" (ByVal Obj As Object, ByVal Value As String)
Declare Sub cPutObjDataField Lib "time2win.dll" (ByVal Obj As Object, ByVal Value As String)
Declare Sub cPutObjDataSource Lib "time2win.dll" (ByVal Obj As Object, ByVal Value As String)
Declare Sub cPutObjTag Lib "time2win.dll" (ByVal Obj As Object, ByVal Value As String)
Declare Sub cPutObjText Lib "time2win.dll" (ByVal Obj As Object, ByVal Value As String)
```

# ObjectMethod

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

Declare Sub cObjectMethod Lib "time2win.dll" (ByVal Obj As Object, ByVal Method As String, ByVal Value As Variant)



# EnableFI, DisableFI

## Purpose :

EnableFI and DisableFI enable or disable mouse and keyboard input to the given form by sending a WM\_ENABLE message and displaying an invisible control such a picture or an image.

When input is disabled, the form ignore input such as mouse clicks and key presses.

When input is enabled, the form process all input.

## Declare Syntax :

```
Declare Sub cEnableFI Lib "time2win.dll" (Obj As Object)
```

```
Declare Sub cDisableFI Lib "time2win.dll" (Obj As Object)
```

## Call Syntax :

```
Call cEnableFI(Ctl)
```

```
Call cDisableFI(Ctl)
```

## Where :

Ctl                    the invisible control that you want become visible (cDisableFI) or invisible (cEnableFI).

## Comments :

I use this function with a picture control which contains a timer BMP.

If the enabled state of the form is changing, a WM\_ENABLE message is sent before this function returns. If a form is already disabled, all its child forms are implicitly disabled, although they are not sent a WM\_ENABLE message.

After some experience, I've noted that some custom controls doesn't answers correctly to this function. In fact, all controls can't receive the input when you Call cDisableFI.

Use this with caution.

**See also :** cEnableForm, cDisableForm

# EnableForm, DisableForm

## Purpose :

EnableForm and DisableForm enable or disable mouse and keyboard input to the given form by sending a WM\_ENABLE message.

When input is disabled, the form ignore input such as mouse click and key press.

When input is enabled, the form process all inputs.

## Declare Syntax :

```
Declare Sub cEnableForm Lib "time2win.dll" (ByVal hWnd As Long)
```

```
Declare Sub cDisableForm Lib "time2win.dll" (ByVal hWnd As Long)
```

## Call Syntax :

```
Call cEnableForm(Form.hWnd)
```

```
Call cDisableForm(Form.hWnd)
```

## Where :

Form.hWnd                    the .hWnd of the specified form

## Comments :

If the enabled state of the form is changing, a WM\_ENABLE message is sent before this function returns. If a form is already disabled, all its child forms are implicitly disabled, although they are not sent a WM\_ENABLE message.

Use this with caution.

**See also :** cEnableFI, cDisableFI

# EnableRedraw, DisableRedraw, EnableCtlRedraw, DisableCtlRedraw, ObjEnableRedraw, ObjDisableRedraw

## Purpose :

[EnableRedraw](#) and [DisableRedraw](#) send a WM\_SETREDRAW message from a hWnd of a control to allow change in that window to be redrawn or to prevent change in that window from being redrawn.

[EnableCtlRedraw](#) and [DisableCtlRedraw](#) send a WM\_SETREDRAW message to a control to allow change in that window to be redrawn or to prevent change in that window from being redrawn.

## Declare Syntax :

```
Declare Sub cEnableRedraw Lib "time2win.dll" (ByVal hWnd As Long)
Declare Sub cDisableRedraw Lib "time2win.dll" (ByVal hWnd As Long)
```

```
Declare Sub cEnableCtlRedraw Lib "time2win.dll" (Obj As Object)
Declare Sub cDisableCtlRedraw Lib "time2win.dll" (Obj As Object)
```

```
Declare Sub cObjEnableRedraw Lib "time2win.dll" (ByVal Obj As Object)
Declare Sub cObjDisableRedraw Lib "time2win.dll" (ByVal Obj As Object)
```

## Call Syntax :

```
Call cEnableRedraw(Ctl.hWnd)
Call cDisableRedraw(Ctl.hWnd)
```

```
Call cEnableCtlRedraw(Ctl)
Call cDisableCtlRedraw(Ctl)
```

## Where :

## Comments :

The WM\_SETREDRAW message can be used to set and clear the redraw flag for a window. This message is very useful for preventing a list box from being updated when many items are being added to it, and then allowing the list box to be redrawn when all of the changes have been made to its contents. Using this technique prevents a list box that is currently visible from flashing constantly as its contents are being updated.

This message sets or clears the redraw flag. If the redraw flag is cleared, the contents of the specified window will not be updated after each change, and the window will not be repainted until the redraw flag is set. For example, an application that needs to add several items to a list box can clear the redraw flag, add the items, and then set the redraw flag. Finally, the application can Call the InvalidateRect function to cause the list box to be repainted.

If the custom control doesn't have a .hWnd (Label control b.e.), you must use the XCtlRedraw routine.

# Printer : Overview

<a href="#"><u>EnumPrinterJobs</u></a>	enumerates all pending jobs on a printer.
<a href="#"><u>EnumPrinters1</u></a>	enumerates all printers (data returned are description, printer name, comment).
<a href="#"><u>EnumPrinters2</u></a>	enumerates all printers (data returned are for remote printers).
<a href="#"><u>EnumPrinters5</u></a>	enumerates all printers (data returned are printer name, port name).
<a href="#"><u>GetPrinterCopies</u></a>	gets the number of copies of a printer.
<a href="#"><u>GetPrinterDefaultSource</u></a>	gets the default source of a printer.
<a href="#"><u>GetPrinterDitherType</u></a>	gets the dither type of pictures of a printer.
<a href="#"><u>GetPrinterOrientation</u></a>	gets the paper orientation of a printer.
<a href="#"><u>GetPrinterPaper</u></a>	gets the paper size of a printer.
<a href="#"><u>GetPrinterQuality</u></a>	gets the quality for a printer.
<a href="#"><u>PrinterHeight</u></a>	returns the height of the printer in inch.
<a href="#"><u>PrinterOffsetLeft</u></a>	returns the left offset of the printer in inch (begin of the printable area).
<a href="#"><u>PrinterOffsetTop</u></a>	returns the top offset of the printer in inch (begin of the printable area).
<a href="#"><u>PrinterWidth</u></a>	returns the height of the printer in inch.
<a href="#"><u>SetPrinterCopies</u></a>	sets the number of copies for a printer.
<a href="#"><u>SetPrinterDefault</u></a>	sets the default printer.
<a href="#"><u>SetPrinterDefaultSource</u></a>	sets the default source for a printer.
<a href="#"><u>SetPrinterDitherType</u></a>	sets the dither type of pictures for a printer.
<a href="#"><u>SetPrinterOrientation</u></a>	sets the paper orientation for a printer.
<a href="#"><u>SetPrinterQuality</u></a>	sets the quality for a printer.

# Date and time : Overview

<a href="#">AddTime</a>	retrieves only the part for hours on one day.
<a href="#">CheckTime</a>	verifies if an hour (in minutes) is between two others hours (in minutes).
<a href="#">CurrentTime</a>	returns the minutes elapsed since midnight.
<a href="#">DateHourToLong</a>	computes a Long from all date-hour parts.
<a href="#">DateToInt</a>	computes an Integer from all date parts.
<a href="#">DateToScalar</a>	computes a scalar (long) from all date parts.
<a href="#">DayOfWeek</a>	calculates the day of the week.
<a href="#">DayOfYear</a>	calculates the day of the year.
<a href="#">DaysInMonth</a>	returns the total days in a month.
<a href="#">ExtCurrentTime</a>	returns the seconds elapsed since midnight.
<a href="#">HourTo</a>	converts a time string to a VARIANT value in minutes (INTEGER or LONG).
<a href="#">IntoBalance</a>	converts a VARIANT value (INTEGER or LONG) in a time string.
<a href="#">IntoBalance</a>	converts a VARIANT value (INTEGER or LONG) in a time string with leading zero.
<a href="#">IntoDate</a>	converts a date value into a date string specified the short date format order in the Control Panel.
<a href="#">IntoDateFill</a>	converts a date value into a date string specified the short date format order in the Control Panel.
<a href="#">IntoDateNull</a>	converts a date value into a date string specified the short date format order in the Control Panel.
<a href="#">IntoFixHour</a>	is super-set for converting a VARIANT (INTEGER or LONG) into a fixed time string.
<a href="#">IntoHour</a>	converts a VARIANT (INTEGER or LONG) into a hour string.
<a href="#">IntoVarHour</a>	converts a VARIANT (INTEGER or LONG) into a hour string (variable length following the value).
<a href="#">IntToDate</a>	decomposes an Integer date into these components.
<a href="#">LongToDateHour</a>	decomposes a Long date-hour into these components.
<a href="#">ScalarToDate</a>	decomposes a scalar date into these components.
<a href="#">ScalarToTime</a>	decomposes a scalar time into these components.
<a href="#">TimeBetween</a>	calculates the time (in minutes) between two hours (in minutes).
<a href="#">TimeToScalar</a>	computes a Int from all time parts.
<a href="#">WeekOfYear</a>	calculates the week of the year.

# GetVersion

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetVersion](#) return the version number of 'TIME TO WIN'

## **Declare Syntax :**

Declare Function cGetVersion Lib "[time2win.dll](#)" () As Single

## **Call Syntax :**

version% = cGetVersion()

## **Where :**

## **Comments :**

This is usefull to avoid version conflict with old version.

## **Examples :**

```
version% = cGetVersion()           ' 3.00
```

**See also :** [TIME2WIN](#)

# TIME2WIN : Overview

[GetVersion](#) return the version number of 'TIME TO WIN'.

# GetNetConnection

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetNetConnection](#) return the name of the network resource associated with the specified redirected local device.

## Declare Syntax :

Declare Function cGetNetConnection Lib "[time2win.dll](#)" (ByVal lpDrive As String, ErrCode As Integer) As String

## Call Syntax :

```
test$ = cGetNetConnection(lpDrive, ErrCode)
```

## Where :

lpDrive	a string specifying the name of the redirected local device.
ErrCode	TRUE is all is ok <> TRUE if an error has occurred
test\$	the returned name of the remote network resource.

## Comments :

## Examples :

**See also :** [Network](#)



## Network : Overview

[GetNetConnection](#)

local device.

[GetNetUser](#)

network connection.

[GetNetAdapterMacAddress](#)

[GetNetAdapterNumber](#)

[GetNetNumberOfAdapter](#)

return the name of the network resource associated with the specified redirected

retrieves the current default user name or the user name used to establish a

retrieves the encoded address of the network adapter (MAC address).

retrieves the logical address of a physical LAN adapter.

retrieves the number of LAN adapter (number of network card).

# LngInpBox

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[LngInpBox](#) is a fully replacement of the standard function `InputBox$`. It supports Multi-Language.

## **Declare Syntax :**

Declare Function `cLngInpBox` Lib "[time2win.dll](#)" (ByVal `nLanguage` As Integer, ByVal `Message` As String, ByVal `Title` As String, ByVal `Default` As String) As String

## **Call Syntax :**

```
test$ = cLngInpBox(nLanguage, Message, Title, Default)
```

## **Where :**

<code>nLanguage</code>	is the language number.
<code>Message</code>	is the message to display.
<code>Title</code>	is the title of the message box.
<code>Default</code>	is the default string to display in the input part.
<code>Test\$</code>	is the returned data in the input part.

## **Comments :**

`nLanguage` must be a language number defined in Constants and Types declaration. If the language number is not correct, the french language is always returned.

The returned data can be an EMPTY string if the 'Cancel' button is pushed. If the 'OK' button is pushed the contents of the input part is returned.

## **Examples :**

```
test$ = cLngInpBox(LNG_FRENCH, "This a new InputBox in French", "TIME TO WIN ", " INPUT BOX IN FRENCH")
```

**See also :** [Multi language message box - input box](#)

# LngBoxMsg, LngMsgBox

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[LngBoxMsg](#) is a fully replacement of the standard sub MsgBox. It supports Multi-Language and add some new parameters.

[LngMsgBox](#) is a fully replacement of the standard function MsgBox. It supports Multi-Language and add some new parameters.

## Declare Syntax :

Declare Sub cLngBoxMsg Lib "time2win.dll" Alias "cLngMsgBox" (ByVal nLanguage As Integer, ByVal Message As String, ByVal Button As Long, ByVal Title As String)

Declare Function cLngMsgBox Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal Message As String, ByVal Button As Long, ByVal Title As String) As Integer

## Call Syntax :

Call cLngBoxMsg(nLanguage, Message, Button, Title)

test% = cLngMsgBox(nLanguage, Message, Button, Title)

## Where :

nLanguage	is the language number.
Message	is the message to display.
Button	specifies the contents and behavior of the message box. This parameter is a combination of the standard MsgBox parameters
Title	is the title of the message box.
test%	is the button Id pushed (see VB MsgBox).

## Comments :

nLanguage must be a language number defined in Constants and Types declaration. If the language number is not correct, the french language is always returned.

Button adds two new parameters : MB\_MESSAGE\_CENTER (centering the message), MB\_MESSAGE\_RIGHT (right-justify the message).

Button adds four mixing timeout : 2, 4, 8, 16 seconds (The timeout can be : 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30 seconds).

Button adds display of the message box without buttons : MB\_NO\_BUTTONS.

If a timeout occurs after no actions from the operator, cLngMsgBox returns the default button.

A timeout occurs even if the system menu of the message box is activated.

The default justification is MB\_MESSAGE\_LEFT.

The icons used a little different from the standard message box.

Beware when using TimeOut fonctionnality in the new message box, use only to display some low warning messages.

## Examples :

Call cLngBoxMsg(LNG\_FRENCH, "This is new.", MB\_ICONSTOP or MB\_MESSAGE\_CENTER or MB\_YESNOCANCEL or MB\_TIMEOUT\_8, "TIME TO WIN")

test% = cLngMsgBox(LNG\_FRENCH, "This is new.", MB\_ICONSTOP or MB\_MESSAGE\_CENTER or MB\_YESNOCANCEL or MB\_TIMEOUT\_12 or MB\_DISPLAY\_TIMEOUT, "TIME TO WIN")

**See also :** [Multi language message box - input box](#)

## Multi language message box - input box : Overview

[LngInpBox](#) is a fully replacement of the standard function InputBox\$. It supports Multi-Language.

[LngBoxMsg](#) is a fully replacement of the standard sub MsgBox. It supports Multi-Language and add some new parameters.

[LngMsgBox](#) is a fully replacement of the standard function MsgBox. It supports Multi-Language and add some new parameters.

# ReadCtlLanguage, SaveCtlLanguage

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[ReadCtlLanguage](#) read a file which contains the text for supporting a language.

[SaveCtlLanguage](#) create or update a file which contains the text for supporting a language.

## Declare Syntax :

```
Declare Function cSaveCtlLanguage Lib "time2win.dll" (Obj As Object, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
```

```
Declare Function cReadCtlLanguage Lib "time2win.dll" (Obj As Object, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
```

## Call Syntax :

```
test% = cSaveCtlLanguage(Obj, Property, FileLanguage)
```

```
test% = cReadCtlLanguage(Obj, Property, FileLanguage)
```

## Where :

Obj	is any object on the form to use the text language.
Property	is an association of constants (RS_CAPTION, RS_TEXT, RS_DATAFIELD, RS_DATASOURCE, RS_TAG)
FileLanguage	is the file name to perform the language management.
test%	TRUE if all is ok FALSE is an error has occurred

## Comments :

These functions are very, VERY simple to use and your application can support multi-language very fast.

If a problem occurs when accessing the controls or if the filename is an EMPTY string, the returned value is FALSE. These functions doesn't test the validity of the file name.

Ctl can be any control on the form (also Label1).

Property can be RS\_CAPTION to use only controls did have a .Caption property.  
can be RS\_TEXT to use only controls did have a .Text property.  
can be RS\_DATAFIELD to use only controls did have a .DataField property.  
can be RS\_DATASOURCE to use only controls did have a .DataSource property.  
can be RS\_TAG to use only controls did have a .Tag property.  
can be any 'OR' association of the four following constants :  
RS\_CAPTION Or RS\_TEXT Or RS\_DATAFIELD Or RS\_DATASOURCE Or RS\_TAG

If you want to use all properties, you can pass the value 255.

If you use of RS\_DATAFIELD and/or RS\_DATASOURCE, you don't need to set the .DataField and/or .DataSource in the Properties Window is design mode. This is can be useful and is not memory hungry, and the EXE size of your application is minder.

FileLanguage is the name of the file to use to store or retrieve the Property. After the first saving, you translate the file (with NOTEPAD, b.e.) into an another language and save it to an other name. You can use the extension as follows .T?? with ?? is FR (for FRENch), UK (for United Kingdom), GE (for GERmany), IT (for ITaly), SP (for SPain), ...

## Examples :

```
test% = cSaveCtlLanguage(Command1, RS_CAPTION Or RS_TEXT, "D:\TIME2WIN\DEMO\TIME2WIN.TUK")  
' translate it to French and save it in the file "D:\TIME2WIN\DEMO\TIME2WIN.TFR"
```

test% = cReadCtlLanguage(Command1, RS\_CAPTION Or RS\_TEXT, "D:\TIME2WIN\DEMO\TIME2WIN.TFR")

**See also :** [Language control](#)

# Language control : Overview

[LngSysMenu](#)

change all text items in a system menu to one of six available language.

[ReadCtlLanguage](#)

read a file which contains the text for supporting a language.

[ReadCtlLanguageExt](#)

language.

read a generic file (one file per language) which contains the text for supporting a

[ReadMnuLanguage](#)

read a file which contains the text (menu) for supporting a language.

[SaveCtlLanguage](#) create or update a file which contains the text for supporting a language.

[SaveCtlLanguageExt](#)

supporting a language.

create or update a generic file (one file par language) which contains the text for

[SaveMnuLanguage](#)

create or update a file which contains the text (menu) for supporting a language.

# LngSysMenu

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[LngSysMenu](#) change all text items in a system menu to one of six available language.

## **Declare Syntax :**

Declare Sub cLngSysMenu Lib "[time2win.dll](#)" (ByVal nLanguage As Integer, ByVal hWnd As Long)

## **Call Syntax :**

Call cLngSysMenu(nLanguage%, hWnd%)

## **Where :**

nLanguage%                    is the language number.  
hWnd%                         is the .hWnd of the form.

## **Comments :**

This sub only changes the item text not the fonctionnality.  
This sub take care of the menu 'grayed'.

nLanguage must be a language number defined in Constants and Types declaration. If the language number is not correct, the french language is always returned.

## **Examples :**

Call cLngSysMenu(LNG\_FRENCH, Me.hWnd)

**See also :** [Language control](#)



# ReadMnuLanguage, SaveMnuLanguage

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SaveMnuLanguage](#) create or update a file which contains the text (menu) for supporting a language.

[ReadMnuLanguage](#) read a file which contains the text (menu) for supporting a language.

## Declare Syntax :

```
Declare Function cReadMnuLanguage Lib "time2win.dll" (hCtlFirstMenu As Control, ByVal FileLanguage As String) As Integer
```

```
Declare Function cSaveMnuLanguage Lib "time2win.dll" (hCtlFirstMenu As Control, ByVal FileLanguage As String) As Integer
```

## Call Syntax :

```
test% = cSaveMnuLanguage(hCtlFirstMenu, FileLanguage)
```

```
test% = cReadMnuLanguage(hCtlFirstMenu, FileLanguage)
```

## Where :

hCtlFirstMenu	is the first menu control on the form.
FileLanguage\$	is the file name to perform the language management.
test%	TRUE if all is ok FALSE is an error has ocured

## Comments :

These functions are very, VERY simple to use and your application can support multi-language very fast.

If a problem occurs when accessing the menus or if the form has no menu or if the filename is an EMPTY string, the returned value is FALSE. These fonctions doesn't test the validity of the file name.

FileLanguage is the name of the file to use to store or retrieve the Property. After the first saving, you translate the file (with NOTEPAD, b.e.) into an another language and save it to an other name. You can use the extension als follows .T?? with ?? is FR (for FRench), UK (for United Kingdom, GE (for GERmany), IT (for ITaly), SP (for SPain), ... .

## Examples :

```
test% = cSaveMnuLanguage(mnu_File, "D:\TIME2WIN\DEMO\TIME2WIN.TUK")  
    ' translate it to French and save it in the file "D:\TIME2WIN\DEMO\TIME2WIN.TFR"  
test% = cReadMnuLanguage(mnu_File, "D:\TIME2WIN\DEMO\TIME2WIN.TFR")
```

**See also :** [Language control](#)

# FileToComboBox, FileToListBox

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileToComboBox](#) read a file and append it to a Combo Box.

[FileToListBox](#) read a file and append it to a List Box.

## Declare Syntax :

Declare Function cFileToComboBox Lib "time2win.dll" (ByVal hWnd As Long, ByVal nFile As String) As Integer

Declare Function cFileToListBox Lib "time2win.dll" (ByVal hWnd As Long, ByVal nFile As String) As Integer

## Call Syntax :

Test% = cFileToComboBox(Combo1.hWnd, nFile\$)

Test% = cFileToListBox(List1.hWnd, nFile\$)

## Where :

Combo1.hWnd	the .hWnd of a Combo Box.
List1.hWnd	the .hWnd of a List Box.
nFile\$	the filename to read.
Test%	= True, if all is ok, <> True, if an error has occurred.

## Comments :

## Examples :

```
Debug.Print cFileToComboBox(Combo1.hWnd, "c:\tmp\cmb_001.txt")
```

```
Debug.Print cFileToListBox(List1.hWnd, "c:\tmp\lst_001.txt")
```

**See also :** [List box - combo box](#)

## List box - combo box : Overview

<a href="#">ArrayToComboBox</a>	read an string array and append it to a Combo Box.
<a href="#">ArrayToListBox</a>	read an string array and append it to a List Box.
<a href="#">ComboFiles</a>	fill a Combo Box with files with the specified attribute and mask.
<a href="#">ComboSearchDir</a>	perform a directory pattern match starting with a specified path and fill a standard combo box.
<a href="#">ComboSearchFile</a>	perform a file pattern match starting with a specified path and fill a standard combo box.
<a href="#">ComboSearchFileAttrib</a>	Search for file(s) with attribute and show the result in a standard combo box.
<a href="#">FileToComboBox</a>	read a file and append it to a Combo Box.
<a href="#">FileToListBox</a>	read a file and append it to a List Box.
<a href="#">ListFiles</a>	fill a List Box with files with the specified attribute and mask.
<a href="#">ListSearchDir</a>	perform a directory pattern match starting with a specified path and fill a standard list box.
<a href="#">ListSearchFile</a>	perform a file pattern match starting with a specified path and fill a standard list box.
<a href="#">ListSearchFileAttrib</a>	Search for file(s) with attribute and show the result in a standard list box.
<a href="#">ListSetTabs</a>	set tabulation in a List Box.

# ArrayToComboBox, ArrayToListBox

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ArrayToComboBox](#) read an string array and append it to a Combo Box.

[ArrayToListBox](#) read an string array and append it to a List Box.

## Declare Syntax :

```
Declare Function cArrayToComboBox Lib "time2win.dll" (ByVal hWnd As Long, Array() As Any) As Integer
Declare Function cArrayToListBox Lib "time2win.dll" (ByVal hWnd As Long, Array() As Any) As Integer
```

## Call Syntax :

```
Test% = cArrayToComboBox(Combo1.hWnd, Array())
```

```
Test% = cArrayToListBox(List1.hWnd, Array())
```

## Where :

Combo1.hWnd	the .hWnd of a Combo Box.
List1.hWnd	the .hWnd of a List Box.
nFile\$	the filename to read.
Test%	= True, if all is ok, <> True, if an error has occurred.

## Comments :

This function can handle only a variable type'd string derived from tagVARSTRING (see below).

Don't forget that if you use the 'ReDim' statement at the procedure level without have declared the array als Global, you must initialize the array before using this function (see below). You must initialize the array with enough space to handle the List/Combo boxes This is due to a VB limitation.

This function can handle huge array (greater than 65535 bytes) (see the example below).

```
Type tagVARSTRING
  Contents          As String
End Type
```

## Examples :

```
ReDim AD(-999 To 999)          As tagVARSTRING
Dim i                          As Long
Dim r                          As Long
```

```
For i = -999 To 999
  AD(i).Contents = Space$(256)
Next i
```

```
Debug.Print cArrayToListBox(List1.hWnd, AD())
Debug.Print cArrayToComboBox(Combo1.hWnd, AD())
```

**See also :** [List box - combo box](#)

## Media ID - Volume : Overview

[DOSGetMediaID](#)

read the media ID (serial number, volume label, ...) from a disk.

[DOSGetVolumeLabel](#)

read the volume label of any disk.

[DOSSetMediaID](#)

change the existing media ID (serial number, volume label, ...) from a disk.

[DOSSetVolumeLabel](#)

create/change/delete the volume label of any disk.

# DOSGetMediaID, DOSSetMediaID

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**DOSGetMediaID** read the media ID (serial number, volume label, ...) from a disk.

**DOSSetMediaID** change the existing media ID (serial number, volume label, ...) from a disk.

## Declare Syntax :

Declare Function cDOSGetMediaID Lib "time2win.dll" (ByVal nDrive As String, DOSMEDIAID As tagDOSMEDIAID) As Integer

Declare Function cDOSSetMediaID Lib "time2win.dll" (ByVal nDrive As String, DOSMEDIAID As tagDOSMEDIAID) As Integer

## Call Syntax :

Test% = cDOSGetMediaID(nDrive\$, DOSMEDIAID)

Test% = cDOSSetMediaID(nDrive\$, DOSMEDIAID)

## Where :

nDrive\$	is the drive letter.
DOSMEDIAID	is the <u>type'd variable</u> to access the drive.
Test%	TRUE, all is ok FALSE, no media ID or an error has occurred.

## Comments :

If nDrive is empty, the default drive is used.

The informations returned by these routines are different from the GetMediaID and SetMediaID.

For T2WIN-32.DLL and T2WOFFIC.DLL :

To decode the 'InfoLevel', you must use cCVI function.

To decode the 'SerialNumber', you must use the cCVL function.

## Examples :

```
Dim DOSMEDIAID As tagMEDIAID
```

```
test% = cDOSGetMediaID("A", DOSMEDIAID)
```

```
' Drive A : no media id
```

```
test% = cDOSGetMediaID("B", DOSMEDIAID)
```

```
' Drive B : no media id
```

```
test% = cDOSGetMediaID("C", DOSMEDIAID)
```

```
' Drive C :
```

```
' InfoLevel : '0' (Hex$(cCVI(DOSMEDIAID.InfoLevel))
```

```
' SerialNumber : '43361ECF' (Hex$(cCVL(DOSMEDIAID.SerialNumber))
```

```
' VolLabel : 'UNICORN_7'
```

```
' FileSysType : 'FAT16'
```

**See also :** [Media ID - Volume](#)



' structure for get/set DOS Media ID

Type tagDOSMEDIAID32

InfoLevel	As String * 2	'use cCVI for integer conversion
SerialNumber	As String * 4	'use cCVL for long conversion
VolLabel	As String * 11	
FileSysType	As String * 8	

End Type

' structure for get/set Media ID

Type tagMEDIAID16

InfoLevel	As Integer
SerialNumber	As Long
VolLabel	As String * 11
FileSysType	As String * 8

End Type



' structure for get/set Media ID

Type tagMEDIAID

VolumeName As String

VolumeSerialNumber As Long

SystemName As String

MaxNameLength As Long

FileSystemFlags As Long

End Type

# DOSGetVolumeLabel, DOSSetVolumeLabel

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DOSGetVolumeLabel](#) read the volume label of any disk.

[DOSSetVolumeLabel](#) create/change/delete the volume label of any disk.

## Declare Syntax :

```
Declare Function cDOSGetVolumeLabel Lib "time2win.dll" (ByVal nDrive As String) As String
```

```
Declare Function cDOSSetVolumeLabel Lib "time2win.dll" (ByVal nDrive As String, ByVal nVolumeLabel As String) As Integer
```

## Call Syntax :

```
VolLbl$ = cDOSGetVolumeLabel(nDrive$)
```

```
Test% = cDOSSetVolumeLabel(nDrive$, NewVolLbl$)
```

## Where :

nDrive\$	is the drive to use.
VolLbl\$	is the readed volume label.
NewVolLbl\$	is the new volume label.
Test%	= True, if all is ok <> True, if an error has occurred.

## Comments :

The length of a volume label can be 11 chars maximum.

The description of a volume label must respect the DOS filename convention.

## Examples :

```
Dim VolLbl As String
```

```
Dim TestAs Integer
```

```
VolLbl = cDOSGetVolumeLabel("A")
```

```
' VolLbl -> "TIME_TO_WIN"
```

```
Test = cDOSSetVolumeLabel("A", "NEW_VOLUME")
```

```
' Test -> -1 (True)
```

**See also :** [Media ID - Volume](#)

# IntoDate, IntoDateFill, IntoDateNull

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**IntoDate** convert a date value into a date string specified the short date format order in the Control Panel.

**IntoDateFill** convert a date value into a date string specified the short date format order in the Control Panel. But if the date is 0, the returned string is 10 spaces according to the maximum chars in the short date format ("dd/mm/yyyy" or "mm/dd/yyyy" or "yyyy/mm/dd").

**IntoDateNull** convert a date value into a date string specified the short date format order in the Control Panel. But if the date is 0, the returned string is an EMPTY string.

## Declare Syntax :

```
Declare Function clntoDate Lib "time2win.dll" (ByVal nDate As Long) As String
Declare Function clntoDateFill Lib "time2win.dll" (ByVal nDate As Long) As String
Declare Function clntoDateNull Lib "time2win.dll" (ByVal nDate As Long) As String
```

## Call Syntax :

```
test$ = clntoDate(nDate)
test$ = clntoDateFill(nDate)
test$ = clntoDateNull(nDate)
```

## Where :

nDate            the date to proceed  
test\$            the date string returned

## Comments :

The date to be proceed is always a LONG.  
This fonction take care of the date separator specified in the Control Panel.

## Examples :

```
test$ = clntoDate(Int(Now))            ' "09/12/1994"
test$ = clntoDateFill(Int(Now))        ' "09/12/1994"
test$ = clntoDateNull(Int(Now))        ' "09/12/1994"

test$ = clntoDate(-1)                  ' "29/12/1899"
test$ = clntoDateFill(-1)              ' "29/12/1899"
test$ = clntoDateNull(-1)              ' "29/12/1899"

test$ = clntoDate(0)                    ' "30/12/1899"
test$ = clntoDateFill(0)                ' "            "
test$ = clntoDateNull(0)                ' ""

test$ = clntoDate(1)                    ' "31/12/1899"
test$ = clntoDateFill(1)                ' "31/12/1899"
test$ = clntoDateNul(1)                 ' "31/12/1899"
```

**See also :** [Date and time](#)

# DayOfYear

## Purpose :

[DayOfYear](#) calculate the day of the year.

## Declare Syntax :

Declare Function cDayOfYear Lib "[time2win.dll](#)" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Integer

## Call Syntax :

Test% = cDayOfYear(nYear%, nMonth%, nDay%)

## Where :

nYear%	is the year.
nMonth%	is the month.
nDay%	is the day.
Test%	is the returned day of the year.

## Comments :

The returned value is 365 or 366 (for a leap year).

If the parameters are incorrect, the returned value is -1.

## Examples :

Dim TestAs Integer

Test = cDayOfYear(1995, 1, 1)	' 1
Test = cDayOfYear(1995, 3, 25)	' 84
Test = cDayOfYear(1995, 12, 31)	' 365
Test = cDayOfYear(1996, 12, 31)	' 366

**See also :** [Date and time](#)

# DayOfWeek

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DayOfWeek](#) calculate the day of the week.

## Declare Syntax :

Declare Function cDayOfWeek Lib "time2win.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer, ByVal nISO As Integer) As Integer

## Call Syntax :

Test% = cDayOfWeek(nYear%, nMonth%, nDay%, nISO%)

## Where :

nYear%	is the year.
nMonth%	is the month.
nDay%	is the day.
nISO%	= True, for ISO specification, = False, for non-ISO specification.
Test%	is the returned day of the week.

## Comments :

Following the ISO specification, the returned day of the week will be 0 (Monday) to 6 (Sunday).  
Following the non-ISO specification, the returned day of the week will be 0 (Sunday) to 6 (Saturday).

If the parameters are incorrect, the returned value is -1.

## Examples :

Dim Test As Integer

' For ISO specification

Test = cDayOfWeek(1995, 3, 25, True)	' 5 (Saturday)
Test = cDayOfWeek(1995, 3, 26, True)	' 6 (Sunday)
Test = cDayOfWeek(1995, 3, 27, True)	' 0 (Monday)

' For non-ISO specification

Test = cDayOfWeek(1995, 3, 25, False)	' 6 (Saturday)
Test = cDayOfWeek(1995, 3, 26, False)	' 0 (Sunday)
Test = cDayOfWeek(1995, 3, 27, False)	' 1 (Monday)

**See also :** [Date and time](#)

# WeekOfYear

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

`WeekOfYear` calculate the week of the year.

## Declare Syntax :

Declare Function `cWeekOfYear` Lib "time2win.dll" (ByVal `nYear` As Integer, ByVal `nMonth` As Integer, ByVal `nDay` As Integer, ByVal `nISO` As Integer) As Integer

## Call Syntax :

`Test%` = `cWeekOfYear`(`nYear%`, `nMonth%`, `nDay%`)

## Where :

`nYear%` is the year.  
`nMonth%` is the month.  
`nDay%` is the day.  
`nISO%` = True, for ISO specification,  
= False, for non-ISO specification.  
`Test%` is the returned week of the year.

## Comments :

ISO defines the first week with 4 or more days in it to be week #1

Following the ISO specification, the returned week of the year will be 0 to 52.  
Following the non-ISO specification, the returned week of the year will be 1 to 53.

If the parameters are incorrect, the returned value is -1.

## Examples :

Dim `Test` As Integer

' Following the ISO specification

```
Test = cWeekOfYear(1995, 12, 31, True) ' 52
Test = cWeekOfYear(1995, 1, 1, True) ' 0
Test = cWeekOfYear(1995, 1, 2, True) ' 1
Test = cWeekOfYear(1995, 3, 25, True) ' 12
Test = cWeekOfYear(1995, 3, 26, True) ' 12
Test = cWeekOfYear(1995, 12, 31, True) ' 52
Test = cWeekOfYear(1996, 1, 1, True) ' 1
```

' Following the non-ISO specification

```
Test = cWeekOfYear(1995, 12, 31, False) ' 53
Test = cWeekOfYear(1995, 1, 1, False) ' 1
Test = cWeekOfYear(1995, 1, 2, False) ' 1
Test = cWeekOfYear(1995, 3, 25, False) " 12
Test = cWeekOfYear(1995, 3, 26, True) ' 13
Test = cWeekOfYear(1995, 12, 31, False) ' 53
Test = cWeekOfYear(1996, 1, 1, False) ' 1
```

**See also :** [Date and time](#)

# DateToScalar, ScalarToDate

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DateToScalar](#) compute a Int from all date parts.

[ScalarToDate](#) decompose a Int date into these components.

## Declare Syntax :

Declare Function cDateToScalar Lib "[time2win.dll](#)" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Long

Declare Sub cScalarToDate Lib "[time2win.dll](#)" (ByVal Scalar As Long, nYear As Integer, nMonth As Integer, nDay As Integer)

## Call Syntax :

Int& = cDateToScalar(nYear%, nMonth%, nDay%)

Call cScalarToDate(Int&, nYear%, nMonth%, nDay%)

## Where :

nYear%	is the year.
nMonth%	is the month.
nDay%	is the day.
Scalar&	is the returned computed Scalar.

## Comments :

The date is computed from 0001.01.01, so 1 is 0001.01.01

For DateToScalar :

If the parameters are not correct, the returned value is -1.

## Examples :

```
Dim Scalar      As Long
Dim nYear      As Integer
Dim nMonth     As Integer
Dim nDay      As Integer
```

```
Scalar = cDateToScalar(1995, 3, 25)      '-> 728377
```

```
Call cScalarToDate(Scalar, nYear%, nMonth%, nDay%)
```

```
' nYear%      ' 1995
' nMonth%     ' 3
' nDay%       ' 25
```

**See also :** [Date and time](#)

# DaysInMonth

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DaysInMonth](#) return the total days in a month.

## Declare Syntax :

Declare Function cDaysInMonth Lib "[time2win.dll](#)" (ByVal nYear As Integer, ByVal nMonth As Integer) As Integer

## Call Syntax :

```
test = cDaysInMonth(nYear, nMonth)
```

## Where :

nYear            is the year with the century  
nMonth          is the month

## Comments :

## Examples :

```
nYear = 1994  
nMonth = 12  
test = cDaysInMonth(nYear, nMonth)        ' 31  
  
nYear = 1995  
nMonth = 2  
test = cDaysInMonth(nYear, nMonth)        ' 28
```

**See also :** [Date and time](#)



# ScalarToTime, TimeToScalar

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ScalarToTime](#) decompose a scalar time into these components.

[TimeToScalar](#) compute a scalar from all time parts.

## Declare Syntax :

Declare Sub cScalarToTime Lib "[time2win.dll](#)" (ByVal Scalar As Long, nHour As Integer, nMin As Integer, nSec As Integer)

Declare Function cTimeToScalar Lib "[time2win.dll](#)" (ByVal nHour As Integer, ByVal nMin As Integer, ByVal nSec As Integer) As Long

## Call Syntax :

Call cScalarToTime(Scalar&, nHour%, nMin%, nSec%)

Scalar& = cTimeToScalar(nHour%, nMin%, nSec%)

## Where :

Scalar&	is a Scalar time.
nHour%	is the returned hour.
nMin%	is the returned minute.
nSec%	is the returned second.

## Comments :

For TimeToScalar :

The parameter Hour can be between 0 to 32767.

If the parameters are not correct, the returned value is -1.

## Examples :

```
Dim Scalar      As Long
Dim nHour      As Integer
Dim nMin       As Integer
Dim nSec       As Integer
```

```
Scalar = cTimeToScalar(16, 50, 30) ' 60630
```

```
Call cScalarToTime(60630, nHour%, nMin%, nSec%)
```

```
' nHour%      16
' nMin%       50
' nSec%       30
```

**See also :** [Date and time](#)

# IntoFixHour, IntoHour, IntoVarHour

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[IntoFixHour](#) is super-set for converting a VARIANT (INTEGER or LONG) into a fixed time string.

[IntoHour](#) convert a VARIANT (INTEGER or LONG) into a hour string.

[IntoVarHour](#) convert a VARIANT (INTEGER or LONG) into a hour string (variable length following the value).

## Declare Syntax :

Declare Function clntoFixHour Lib "time2win.dll" (Var As Variant, ByVal Length As Integer, ByVal fillZero As Integer, ByVal Hundreds As Integer) As String

Declare Function clntoHour Lib "time2win.dll" (Var As Variant) As String

Declare Function clntoVarHour Lib "time2win.dll" (Var As Variant) As String

## Call Syntax :

```
test$ = clntoFixHour(Var, Length, fillZero, Hundreds)
```

```
test$ = clntoHour(Var)
```

```
test$ = clntoVarHour(Var)
```

## Where :

Var	the VARIANT value (LONG or INTEGER) to proceed
Length	the length of the returned time string
fillZero	TRUE if the time string must be filled with zero 0, FALSE if it not
Hundreds	TRUE if the minutes must be converted in Hundreds, FALSE if it not. (This is useful for making calculation)
test\$	the returned time string

## Comments :

For the clntoFixHour function, if the value can be fitted in the length specified, the return string is filled with '?'  
The maximum format for the returned time string is HHHHHHHH:MM

## Examples :

Convert 12345 minutes into fixed hour :

Length	fillZero = TRUE	fillZero = FALSE
0	""	""
1	"?"	"?"
2	"??"	"??"
3	"???"	"???"
4	"????"	"????"
5	"?????"	"?????"
6	"205:45"	"205:45"
7	"0205:45"	" 205:45"
8	"00205:45"	" 205:45"
9	"000205:45"	" 205:45"
10	"0000205:45"	" 205:45"
11	"00000205:45"	" 205:45"

**See also** : [Date and time](#)

# IntoBalance, IntoBalanceFill

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[IntoBalance](#) convert a VARIANT value (INTEGER or LONG) in a time string.

[IntoBalance](#) convert a VARIANT value (INTEGER or LONG) in a time string with leading zero.

## Declare Syntax :

```
Declare Function clntoBalance Lib "time2win.dll" (Var As Variant) As String
Declare Function clntoBalanceFill Lib "time2win.dll" (Var As Variant) As String
```

## Call Syntax :

```
test$ = clntoBalance(Var)
test$ = clntoBalanceFill(Var)
```

## Where :

Var                    the value to convert  
test\$                  the time string

## Comments :

For a positive value :

The format returned for the time string is "HHHHHH:MM"

For a negative value :

The maximum format and the minimum format returned for the time string is "-HHHHH:MM"

## Examples :

IntoBalanceFill	IntoBalance
1234 is "00020:34"	" 20:34"
1235 is "00020:35"	" 20:35"
1236 is "00020:36"	" 20:36"
1237 is "00020:37"	" 20:37"
1238 is "00020:38"	" 20:38"
1239 is "00020:39"	" 20:39"
1240 is "00020:40"	" 20:40"
1241 is "00020:41"	" 20:41"
1242 is "00020:42"	" 20:42"
1243 is "00020:43"	" 20:43"
1244 is "00020:44"	" 20:44"
1245 is "00020:45"	" 20:45"

**See also :** [Date and time](#)

# CurrentTime

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CurrentTime](#) returns the minutes elapsed since midnight.

[ExtCurrentTime](#) returns the seconds elapsed since midnight.

## Declare Syntax :

```
Declare Function cCurrentTime Lib "time2win.dll" () As Integer
Declare Function cExtCurrentTime Lib "time2win.dll" () As Long
```

## Call Syntax :

```
test% = cCurrentTime()
test& = cExtCurrentTime()
```

## Where :

test%	the minutes
test&	the seconds

## Comments :

## Examples :

```
test% = cCurrentTime()           ' 1234
test& = cExtCurrentTime()        ' 86399
```

**See also :** [Date and time](#)

## Bitmap : Overview

<a href="#"><u>DIBSaveScreen</u></a>	save the screen (entire desktop) in a file (DIB format).
<a href="#"><u>DIBSaveWindow</u></a>	save a window in a file (DIB format).
<a href="#"><u>DisplaySplash</u></a>	display a bitmap as splash screen.
<a href="#"><u>DestroySplash</u></a>	destroy a bitmap displayed by DisplaySplash.
<a href="#"><u>TileBitmapOnWindow</u></a>	tile a bitmap (DDB or DIB format) on a window.

# TimeBetween

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[TimeBetween](#) calculate the time (in minutes) between two hours (in minutes).

## Declare Syntax :

Declare Function cTimeBetween Lib "[time2win.dll](#)" (ByVal Hr1 As Integer, ByVal Hr2 As Integer) As Integer

## Call Syntax :

```
test% = cTimeBetween(Hr1, Hr2)
```

## Where :

Hr1                    the first time (0 to 1439)  
Hr2                    the second time (0 to 1439)

## Comments :

## Examples :

```
test% = cTimeBetween(600, 721)            ' 121  
test% = cTimeBetween(1438, 62)           ' 64
```

**See also** : [Date and time](#)

# AddTime

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[AddTime](#) retrieve only the part for hours on one day.

## Declare Syntax :

Declare Function cAddTime Lib "[time2win.dll](#)" (ByVal Hr As Integer) As Integer

## Call Syntax :

test = cAddTime(Hr)

## Where :

Hr                    is the total minutes  
test                  is the result value.

## Comments :

## Examples :

test = cAddTime(1439+2)            ' 1  
test = cAddTime(2-4)                ' 1438

**See also :** [Date and time](#)

# CheckTime

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[CheckTime](#) verify if an hour (in minutes) is between two others hours (in minutes).

## Declare Syntax :

Declare Function cCheckTime Lib "[time2win.dll](#)" (ByVal Hr As Integer, ByVal Hr1 As Integer, ByVal Hr2 As Integer) As Integer

## Call Syntax :

```
test = cCheckTime(Hr, Hr1, Hr2)
```

## Where :

Hr	the hour (in minutes) to test
Hr1	the first hour
Hr2	the second value
test	TRUE if Hr is between Hr1 and Hr2

## Comments :

## Examples :

```
Hr = 1439           ' (23:59)
Hr1 = 1400         ' (23:20)
Hr2 = 10           ' (00:10)
```

```
test = cCheckTime(Hr, Hr1, Hr2) ' TRUE
```

```
Hr = 120           ' (02:00)
```

```
test = cCheckTime(Hr, Hr1, Hr2) ' FALSE
```

**See also :** [Date and time](#)



# AddTwoTimes

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[AddTwoTimes](#) add two time string to form a third time string.

## Declare Syntax :

Declare Function cAddTwoTimes Lib "[time2win.dll](#)" (ByVal Time1 As String, ByVal Time2 As String) As String

## Call Syntax :

Test\$ = cAddTwoTimes(Time1\$, Time2\$)

## Where :

Time1\$	is the first time string (format is HH:MM:SS).
Time2\$	is the second time string (format is HH:MM:SS).
Test\$	is the result (format is HH:MM:SS).

## Comments :

The length of each time string must be absolutely 8 characters.  
The format of each time string must be absolutely HH:MM:SS.  
If the sum of the two time string exceed 24:00:00, the returned string is calculated from 00:00:00.

## Examples :

```
Dim Time1 As String
Dim Time2 As String
Dim Time3 As String
```

```
Time1 = "23:58:58"
Time2 = "01:02:01"
```

```
Time3 = cAddTwoTimes(Time1$, Time2$) ' "01:00:59"
```

**See also** : [Date and time](#)

# HourTo

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

**HourTo** convert a time string to a VARIANT value in minutes (INTEGER or LONG).

## Declare Syntax :

Declare Function cHourTo Lib "[time2win.dll](#)" (Txt As String) As Variant

## Call Syntax :

test = cHourTo(Txt)

## Where :

Txt	the time to convert
test	the time in minutes

## Comments :

The maximum format is for positive time "HHHHHHH:MM" and for negative time "-HHHHHH:MM"  
The returned value is a VARIANT (INTEGER or LONG).

## Examples :

The time "123:45"	is 7425 minutes
The time "23:58"	is 1438 minutes
The time "7:36"	is 456 minutes
The time " :24"	is 24 minutes
The time " :4"	is 4 minutes
The time " :"	is 0 minutes

The time "-123:45"	is -7425 minutes
The time "-23:58"	is -1438 minutes
The time "-7:36"	is -456 minutes
The time " -:24"	is -24 minutes
The time " -:4"	is -4 minutes
The time " -:"	is 0 minutes

**See also :** [Date and time](#)

# DIBSaveScreen, DIBSaveWindow

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DIBSaveScreen](#) save the screen (entire desktop) in a file.

[DIBSaveWindow](#) save a window in a file.

## Declare Syntax :

```
Public Const DIB_SAVE_WINDOW = True
Public Const DIB_SAVE_CLIENT = False
```

```
Declare Function cDIBSaveScreen Lib "time2win.dll" (ByVal lpFileName As String) As Integer
```

```
Declare Function cDIBSaveWindow Lib "time2win.dll" (ByVal hWnd As Long, ByVal SaveArea As Integer, ByVal lpFileName As String) As Integer
```

## Call Syntax :

```
intResult% = cDIBSaveScreen(lpFileName$)
intResult% = cDIBSaveWindow(hWnd&, SaveArea%, lpFileName$)
```

## Where :

lpFileName\$	is the name of the file to save the DIB (Device-Independent Bitmap)
hWnd&	is the .hWnd property of a form or a control
SaveArea%	DIB_SAVE_WINDOW : save the client area and the non-client area DIB_SAVE_CLIENT : save only the client area
intResul%	True : all is OK False : an error has occurred

## Comments :

All files saven with these functions can be used with the .LoadPicture property.

## Examples :

```
debug.print cDIBSaveScreen("c:\test\save_scr.bmp")
debug.print cDIBSaveWindow(Me.hWnd, DIB_SAVE_WINDOW, "c:\test\save_win.bmp")
debug.print cDIBSaveWindow(Me.hWnd, DIB_SAVE_CLIENT, "c:\test\save_cli.bmp")
```

**See also** : [Bitmap](#)

# InstallHookKeyboard

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[InstallHookKeyboard](#) install a hook of the keyboard to handle special keys for special tasks.

## Declare Syntax :

Declare Function cInstallHookKeyboard Lib "[time2win.dll](#)" (ByVal InstallRemove As Integer) As Integer

## Call Syntax :

intResult% = cInstallHookKeyboard(InstallRemove%)

## Where :

InstallRemove%	TRUE to add the hook FALSE to remove the hook
intResult%	TRUE : the hook has been successfully installed FALSE : an error has occurred or the hook has been already installed

## Comments :

Press ALT+CTRL+SHIFT+F11 to open a dialog box for save the screen in a file to be selected.  
Press ALT+CTRL+SHIFT+F12 to open a dialog box for save the window in a file to be selected.

There is no need to call this function with the FALSE parameter when you stop your program. The hook of the keyboard will be automatically removed when T2WIN-32.DLL will be removed from the memory

## Examples :

```
debug.print cInstallHookKeyboard(TRUE)
```

' Press ALT+CTRL+SHIFT+F11 : for save the screen in a file to be selected.  
' Press ALT+CTRL+SHIFT+F12 : for save the active window in a file to be selected.

**See also :** [Hook keyboard](#)

# TileBitmapOnWindow

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[TileBitmapOnWindow](#) tile a bitmap (DDB or DIB format) on a window.

## Declare Syntax :

Declare Function cTileBitmapOnWindow Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal lpFileName As String) As Integer

## Call Syntax :

```
intResult% = cTileBitmapOnWindow(hWnd&, lpFileName$)
```

## Where :

hWnd& is the .hWnd property of a form or a control  
lpFileName\$ is the name of the file to read the DDB (Device-Dependent Bitmap) or DIB (Device-Independent Bitmap)

intResult% TRUE : all is OK  
FALSE : lpFileName\$ not exist

## Comments :

The function take care of the state of the form.

You must set the .AutoRedraw property to False.

To perform an autoredraw, you must do this :

```
Private Sub Form_Paint()  
  
    Dim intResult As Integer  
  
    intResult = cTileBitmapOnWindow(Me.hWnd, App.Path + "\time2win.dib")  
  
End Sub
```

## Examples :

```
debug.print cTileBitmapOnWindow(Me.hWnd, "c:\test\time2win.dib")
```

**See also** : [Bitmap](#)

# Hook keyboard : Overview

[InstallHookKeyboard](#) install a hook keyboard to save the screen or the active window in a file (DIB format).

# RegistryKeyInfo

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[RegistryKeyInfo](#) retrieves informations about the specified key in the Windows registry.

## Declare Syntax :

Declare Function cRegistryKeyInfo Lib "time2win.dll" (ByVal lpSection As String, REGISTRYKEYINFO As tagREGISTRYKEYINFO) As Integer

## Call Syntax :

retCode% = cRegistryKeyInfo(lpSection\$, REGISTRYKEYINFO)

## Where :

lpSection\$	string expression containing the name of the section where the key setting is being saved.
REGISTRYKEYINFO	is a type'd structure
	Type tagREGISTRYKEYINFO
	ISubKeys                  As Long                  'number of subkeys
	IMaxSubKeyLen          As Long                  'maximum length of subkeys
	IValues                 As Long                  'number of keys
	IMaxValueNameLen      As Long                  'maximum length of the name of keys
	IMaxValueLen           As Long                  'maximum length of the value of
	keys
	sInfolnStr              As String                 'above informations in a string
	End Type
retCode%	<u>error/success code.</u>

## Comments :

## Examples :

```
strDisplay = strDisplay & "Key information is " & cRegistryKeyInfo(Section1, RKI) & "" & vbCrLf
strDisplay = strDisplay & "    SubKeys = " & RKI.ISubKeys & vbCrLf
strDisplay = strDisplay & "    MaxSubKeyLen = " & RKI.IMaxSubKeyLen & vbCrLf
strDisplay = strDisplay & "    Values = " & RKI.IValues & vbCrLf
strDisplay = strDisplay & "    MaxValueNameLen = " & RKI.IMaxValueNameLen & vbCrLf
strDisplay = strDisplay & "    MaxValueLen = " & RKI.IMaxValueLen & vbCrLf
strDisplay = strDisplay & "    InfolnStr = " & RKI.sInfolnStr & vbCrLf & vbCrLf
```

## Result :

Getting default value is 'no key'  
Getting value of key 'key2' is 'test key 2'  
Getting value of key 'key1' is 'test key 1'

```
Key information is '-1'
    SubKeys = 0
    MaxSubKeyLen = 0
    Values = 3
    MaxValueNameLen = 4
    MaxValueLen = 11
    InfolnStr = 0;0;3;4;11
```

**See also :** [Registry key](#)





## Registry key : Overview

<a href="#">GetAllSettings</a>	returns a list of key settings and their respective values from an application's entry in the registry.
<a href="#">GetRegistry</a>	returns a key setting value from an application's Windows registry entry.
<a href="#">KillRegistry</a>	deletes a section or key setting from the Windows registry entry.
<a href="#">PutRegistry</a>	saves or creates an application entry in the Windows registry entry.
<a href="#">GetRegistryExt</a>	returns a key setting value from an application's Windows registry entry for any key handle.
<a href="#">KillRegistryExt</a>	deletes a section or key setting from the Windows registry entry for any key handle.
<a href="#">PutRegistryExt</a>	saves or creates an application entry in the Windows registry entry for any key handle.
<a href="#">RegistryKeyInfo</a>	retrieves informations about the specified key in the Windows registry.

# RegistrationKey, RegistrationKey2, RegistrationKey3

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[RegistrationKey](#) perform the calculation of a key from a name and one code.

[RegistrationKey2](#) perform the calculation of a key from a name and two code.

[RegistrationKey3](#) perform the calculation of a key from a name and three code.

## Declare Syntax :

Declare Function cRegistrationKey Lib "time2win.dll" (ByVal RegText As String, ByVal RegKey1 As Long) As Long

Declare Function cRegistrationKey2 Lib "time2win.dll" (ByVal RegText As String, ByVal RegKey1 As Long, ByVal RegKey2 As Long) As Long

Declare Function cRegistrationKey3 Lib "time2win.dll" (ByVal RegText As String, ByVal RegKey1 As Long, ByVal RegKey2 As Long, ByVal RegKey3 As Long) As Long

## Call Syntax :

Key& = cRegistrationKey(RegString\$, RegCode&)

## Where :

RegText\$	the name for the registration.
RegKey1&	the basis code for generating the registration
RegKey2&	the first extended code for generating the registration
RegKey3&	the second extended code for generating the registration
Key&	= 0, if length of RegText is < 10 or if RegKey1 is 0, <>0, the key calculated from RegText and RegKey1.

## Comments :

Using this registration key system, you can easily and quickly generate and verify the validity of numerical registration keys that correspond to a person who has purchased your program. Thus, when someone who already has a shareware or demo version of your program wishes to purchase the program, you need only send them a simple registration key number, instead of sending an entire registered version. You can simply use this package to generate a unique registration key number which corresponds to the user's name (or any other string you wish to use). The user will then be able to enter this number into your software's configuration file / configuration program. When your program begins, it will be able to read this number from the configuration file, and again using this package, determine whether it is a valid registration key corresponding to the user's name. If the registration key is valid, your program can switch into "registered mode", and if not, can run in its unregistered "unregistered mode". (Source from Brian Pirie).

## Examples :

```
Dim Key           As Long
Dim RegText       As String
```

```
RegText = "this is a testthis is a test"
```

```
Key = cRegistrationKey(Tmp, 123456789)           ' 590573797
```

```
Key = cRegistrationKey3(Tmp, 123456789, 864297531, 12344321) ' 132616468
```

**See also :** [Protection](#)

## Protection : Overview

[HashMD5](#)

perform the hash algorithm (MD5) to a specified string.

[Mod10](#)

calculate the mod-10 of the given string.

[Mod10R](#)

calculate the reverse mod-10 of the given string.

[Mod11](#)

calculate the mod-11 of the given string.

[Mod11R](#)

calculate the reverse mod-11 of the given string.

[Mod137](#)

calculate the mod-1.3.7 of the given string.

[Mod137R](#)

calculate the reverse mod-1.3.7 of the given string.

[RegistrationKey](#)

perform the calculation of a key from a name and one code.

[RegistrationKey2](#)

perform the calculation of a key from a name and two code.

[RegistrationKey3](#)

perform the calculation of a key from a name and three code.

# ReadCtlLanguageExt, SaveCtlLanguageExt

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[SaveCtlLanguageExt](#) create or update a generic file (one file par language) which contains the text for supporting a language.

[ReadCtlLanguageExt](#) read a generic file (one file per language) which contains the text for supporting a language.

## Declare Syntax :

```
Declare Function cSaveCtlLanguageExt Lib "time2win.dll" (Obj As Object, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
```

```
Declare Function cReadCtlLanguageExt Lib "time2win.dll" (Obj As Object, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
```

## Call Syntax :

```
test% = cSaveCtlLanguage(Obj, Property, FileLanguage)
```

```
test% = cReadCtlLanguage(Obj, Property, FileLanguage)
```

## Where :

Obj	is any object on the form to use the text language.
Property	is an association of constants (RS_CAPTION, RS_TEXT, RS_DATAFIELD, RS_DATASOURCE, RS_TAG)
FileLanguage	is the file name to perform the language management.
test%	TRUE if all is ok FALSE is an error has occurred

## Comments :

These functions are very, VERY simple to use and your application can support multi-language very fast.

If a problem occurs when accessing the controls or if the filename is an EMPTY string, the returned value is FALSE. These fonctions doesn't test the validity of the file name.

Ctl can be any control on the form (also Label1).

Property can be RS\_CAPTION to use only controls did have a .Caption property.  
can be RS\_TEXT to use only controls did have a .Text property.  
can be RS\_DATAFIELD to use only controls did have a .DataField property.  
can be RS\_DATASOURCE to use only controls did have a .DataSource property.  
can be RS\_TAG to use only controls did have a .Tag property.  
can be any 'OR' association of the four following constants :  
RS\_CAPTION Or RS\_TEXT Or RS\_DATAFIELD Or RS\_DATASOURCE Or RS\_TAG

If ypu want to use all properties, you can pass the value 255.

If you use of RS\_DATAFIELD and/or RS\_DATASOURCE, you don't need to set the .DataField and/or .DataSource in the Properties Window is design mode. This is can be useful and is not memory hungry, and the EXE size of your application is minder.

FileLanguage is the name of the file to use to store or retrieve the Property. After the first saving, you translate the file (with NOTEPAD, b.e.) into an another language and save it to an other name. You can use the extension als follows .T?? with ?? is FR (for FRench), UK (for United Kingdom, GE (for GERmany), IT (for ITaly), SP (for SPain), ... .

## Examples :

```
test% = cSaveCtlLanguageExt(Command1, RS_CAPTION Or RS_TEXT, "D:\TIME2WIN\DEMO\TIME2WIN.TUK")  
      ' translate it to French and save it in the file "D:\TIME2WIN\DEMO\TIME2WIN.TFR"  
test% = cReadCtlLanguageExt(Command1, RS_CAPTION Or RS_TEXT, "D:\TIME2WIN\DEMO\TIME2WIN.TFR")
```

**See also :** [Language control](#)

# HashMD5

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[HashMD5](#) perform the hash algorithm (MD5) to a specified string.

## Declare Syntax :

Declare Function cHashMD5 Lib "[time2win.dll](#)" (Text As String) As String

## Call Syntax :

Hash\$ = cHashMD5(Text\$)

## Where :

Text\$                               the specified string (length between 1 to 32767).  
Hash\$                                the returned hashed string.

## Comments :

A hash algorithm such as MD5 is often used in cryptosystems to "reduce" a user-supplied passphrase into a sufficient number of bits to use as a key to the system. The following is taken from the Executive Summary section of the Internet RFC that proposes MD5 as a standard.

The [MD5] algorithm takes as input an input message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. (Source from Andy Brown).

HashMD5 is derived from the RSA   \*\* \*\* Data Security, Inc. MD5 Message-Digest Algorithm.

## Examples :

Dim Hash                As String

Hash = cToHexa(cHashMD5("TIME TO WIN"))                               '24456922E9A382257E22338DEC584191

**See also :** [Protection](#)

# Windows : Overview

<a href="#">ArrangeDesktopIcons</a>	arranges all desktop icons.
<a href="#">CenterWindow</a>	centers a window in the screen.
<a href="#">EXENameActiveWindow</a>	retrieves the full filename (path and file) of the active window.
<a href="#">EXENameTask</a>	retrieves the full path and filename of the executable file from which the specified module was loaded.
<a href="#">EXENameWindow</a>	retrieves the full filename (path and file) of the specified window.
<a href="#">ExitWindowsAndExecute</a>	terminates Windows, runs a specified MS-DOS application, and then restarts Windows.
<a href="#">GetClassName</a>	retrieves the full class name of a window.
<a href="#">GetCountry</a>	returns the country name.
<a href="#">GetCountryCode</a>	returns the country code.
<a href="#">GetCurrency</a>	returns the currency.
<a href="#">GetCurrentDrive</a>	returns the current default drive.
<a href="#">GetDateFormat</a>	returns the format for the date.
<a href="#">GetDateSeparator</a>	returns the separator for the date.
<a href="#">GetDefaultPrinter</a>	returns the default printer in the [windows] section of Win.INI
<a href="#">GetDevices</a>	returns all devices founden in the [devices] section in the Win.INI
<a href="#">GetDriveCurrentDir</a>	retrieves the current dir on the specified drive.
<a href="#">GetHourFormat</a>	returns the format for the hour.
<a href="#">GetIni</a>	retrieves an item in a section of an INI file.
<a href="#">GetLanguage</a>	returns the letters for the language.
<a href="#">GetListSeparator</a>	returns the separator for list.
<a href="#">GetPrinterPorts</a>	returns all printers set in the [printerports] section in the Win.INI
<a href="#">GetSectionItems</a>	retrieves all items founden in a section of a specified INI file.
<a href="#">GetSystemDirectory</a>	retrieves the full path of the System directory for Windows.
<a href="#">GetTimeSeparator</a>	returns the separator for the date.
<a href="#">GetWindowsDirectory</a>	retrieves the full path for the Windows directory.
<a href="#">GetWinINI</a>	returns the information for a given item.
<a href="#">Gradient</a>	creates a gradient (8 effects) on a control/form which can accept .hDC
<a href="#">PutIni</a>	saves an item in a section of an INI file.
<a href="#">RebootSystem</a>	reboots your system.
<a href="#">RestartWindows</a>	restarts your Windows.
<a href="#">RunFile</a>	opens a specified executable or document file.
<a href="#">ShowWindow</a>	shows a window after an exploded/imploded focus rectangle has been displayed.
<a href="#">WalkThruWindow</a>	walks in the window's list of all windows at a given moment.

# PutIni

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[PutIni](#) save an item in a section of an INI file.

## **Declare Syntax :**

Declare Sub cPutIni Lib "[time2win.dll](#)" (ByVal appName As String, ByVal szItem As String, ByVal szDefault As String, ByVal InitFile As String)

## **Call Syntax :**

Call cPutIni(appName, szItem, szDefault, InitFile)

## **Where :**

appName	a string that specifies the section to which the string will be copied. If the section does not exist, it is created.
szItem	a string containing the entry to be associated with the string. If the entry does not exist in the specified section, it is created.
szDefault	If this parameter is NULL, the entire section, including all entries within the section, is deleted. a string to be written to the file. If this parameter is NULL, the entry specified by the szItem parameter is deleted.
InitFile	a filename that names the initialization file.

## **Comments :**

To improve performance, Windows keeps a cached version of the most-recently accessed initialization file. If that filename is specified and the other three parameters are NULL, Windows flushes the cache.

Sections in the initialization file have the following form:

```
[section]  
entry=string
```

## **Examples :**

Call cPutIni("Desktop","IconTitleFaceName","MS Sans Serif","WIN.INI")

**See also :** [Windows](#)



# GetSeparator.X

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

All values returned are readed from the Win.INI file.

[GetCountry](#) return the country name.  
[GetCountryCode](#) return the country code.  
[GetCurrency](#) return the currency.  
[GetDateFormat](#) return the format for the date.  
[GetDateSeparator](#) return the separator for the date.  
[GetHourFormat](#) return the format for the hour.  
[GetLanguage](#) return the letters for the language.  
[GetListSeparator](#) return the separator for list.  
[GetTimeSeparator](#) return the separator for the date.  
[GetWinINI](#) return the information for a given item.

## Declare Syntax :

```
Declare Function cGetCountry Lib "time2win.dll" () As String
Declare Function cGetCountryCode Lib "time2win.dll" () As String
Declare Function cGetCurrency Lib "time2win.dll" () As String
Declare Function cGetDateFormat Lib "time2win.dll" () As String
Declare Function cGetDateSeparator Lib "time2win.dll" () As String
Declare Function cGetHourFormat Lib "time2win.dll" () As String
Declare Function cGetLanguage Lib "time2win.dll" () As String
Declare Function cGetListSeparator Lib "time2win.dll" () As String
Declare Function cGetTimeSeparator Lib "time2win.dll" () As String
Declare Function cGetWinINI Lib "time2win.dll" (ByVal Info As Integer) As String
```

## Call Syntax :

The purpose and the declare syntax are very explicite.

## Where :

Info                    the number of the desired item

## Comments :

The advantage to use these routines is that these routines is very fast and doesn't use the WINDOWS API in VB.

## Examples :

GetDateSeparator	is '/'
GetTimeSeparator	is ':'
GetListSeparator	is ';'
GetDateFormat	is 'dd/mm/yyyy'
GetHourFormat	is 'hh:nn'
GetCurrency	is 'FB'
GetLanguage	is 'fra'
GetCountry	is 'Belgium (French)'
GetCountryCode	is '32'

**See also :** [Windows](#)

' definition for win.ini section

```
Public Const GET_TIME_SEPARATOR = 1
Public Const GET_DATE_SEPARATOR = 2
Public Const GET_TIME_FORMAT = 3
Public Const GET_DATE_FORMAT = 4
Public Const GET_CURRENCY = 5
Public Const GET_LANGUAGE = 6
Public Const GET_COUNTRY = 7
Public Const GET_COUNTRY_CODE = 8
Public Const GET_LIST_SEPARATOR = 9
Public Const GET_DEFAULT_PRINTER = 10
```

# GetWindowsDirectory

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetWindowsDirectory](#) retrieve the full path for the Windows directory.

## **Declare Syntax :**

Declare Function cGetWindowsDirectory Lib "[time2win.dll](#)" () As String

## **Call Syntax :**

```
test$ = cGetWindowsDirectory()
```

## **Where :**

test\$            is the full path

## **Comments :**

## **Examples :**

```
test$ = cGetWindowsDirectory()            ' "K:\WIN95"
```

**See also :** [Windows](#)

# GetSystemDirectory

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetSystemDirectory](#) retrieve the full path of the System directory for Windows.

## **Declare Syntax :**

Declare Function cGetSystemDirectory Lib "[time2win.dll](#)" () As String

## **Call Syntax :**

```
test$ = cGetSystemDirectory()
```

## **Where :**

test\$                      the full path of the System directory

## **Comments :**

## **Examples :**

```
test$ = cGetSystemDirectory()                      ' "K:\WIN95\SYSTEM"
```

**See also :** [Windows](#)

# GetTaskName

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetTaskName](#) read the name of the task. You see the name in the Task Manager by pressing the CTRL + ESC keys.

## **Declare Syntax :**

```
Declare Function cGetTaskName Lib "time2win.dll" (ByVal hWnd As Long) As String
```

## **Call Syntax :**

```
test$ = cGetTaskName(Form.hWnd)
```

## **Where :**

Form.hWnd	is the hWnd of your application
test\$	is the old task name of the application

## **Comments :**

This is useful to retrieve the task name.

## **Examples :**

```
Dim TaskName As String  
  
TaskName = cGetTaskName(Me.hWnd)  
MsgBox TaskName ' "Microsoft Visual Basic"
```

**See also :** [Task - File version](#)

# GetSectionItems

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[GetSectionItems](#) retrieve all items founden in a section of a specified INI file.

## Declare Syntax :

Declare Function cGetSectionItems Lib "time2win.dll" (ByVal Section As String, ByVal InitFile As String, nItems As Integer) As String

## Call Syntax :

```
test$ = cGetSectionItems(Section, InitFile, nItems)
```

## Where :

Section	the section to proceed
InitFile	the INI file to proceed.
nItems	the total items founden in the section
test\$	the items in the specified section

## Comments :

If the section don't exists, the returned file is an EMPTY string and nItems is 0.  
The InitFile is any file which have a INI structure.  
Each item in the section is separated by a chr\$(13).

## Examples :

```
Dim n As Integer
```

```
Debug.Print cGetSectionItems("desktop", "win.ini", n)
```

```
Debug.Print "Total Items founded in this section is " & n
```

On my system :

```
Pattern=(None)
GridGranularity=0
IconSpacing=77
TileWallPaper=1
IconTitleFaceName=MS Sans Serif
IconTitleSize=-11
IconTitleStyle=0
IconVerticalSpacing=72
wallpaper=(None)
```

```
Total Items founded in this section is = 9
```

```
Debug.Print cGetSectionItems("intl", "win.ini", n)
```

```
Debug.Print "Total Items founded in this section is " & n
```

```
sLanguage=fra
sCountry=Belgium (French)
iCountry=32
iDate=1
iTime=1
```

iTLZero=0  
iCurrency=3  
iCurrDigits=2  
iNegCurr=8  
iLzero=0  
iDigits=2  
iMeasure=0  
s1159=  
s2359=  
sCurrency=FB  
sThousand=  
sDecimal=,  
sDate=/  
sTime=:  
sList=;  
sShortDate=d/MM/yy  
sLongDate=ddd d MMMM yyyy  
sFrameNum=#mmjk`sdnm

Total Items founded in this section is = 23

**See also :** [Windows](#)

# GetPrinterPorts

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetPrinterPorts](#) return all printers set in the [printerports] section in the Win.INI

## **Declare Syntax :**

Declare Function cGetPrinterPorts Lib "[time2win.dll](#)" () As String

## **Call Syntax :**

```
test$ = cGetPrinterPorts()
```

## **Where :**

test\$                      all printer founded separated by a chr\$(13).

## **Comments :**

Use the cGetIn function to extract each printer.

**See also :** [Windows](#)



# ChangeTaskName

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ChangeTaskName](#) change the name of the task. You see change in the Task Manager by pressing the CTRL + ESC keys.

## Declare Syntax :

Declare Sub cChangeTaskName Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal Text As String)

## Call Syntax :

Call cChangeTaskName(Form.hWnd, Text)

## Where :

Form.hWnd                    is the hWnd of your application  
Text                            is the new task name to given at your application

## Comments :

This is useful to set a particular task name at your application.

## Examples :

Call cChangeTaskName(Me.hWnd, "Hello world")  
    ' press the CTRL + ESC keys to see the change in the Task Manager

**See also :** [Task - File version](#)

# ShowWindow

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ShowWindow](#) show a window after an exploded/imploded focus rectangle has been displayed.

## Declare Syntax :

Declare Sub cShowWindow Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal method As Integer, ByVal interval As Integer)

## Call Syntax :

Call cShowWindow(hWnd%, method%, interval%)

## Where :

hWnd%	is the handle of a form.
method%	0 : explode the form starting at center of the form. 1 : implode the form starting at external.
interval%	0 : faster 699 : lower

## Comments :

The interval is a modulo of 700 and is calculated in millisecond.

## Examples :

Call cShowWindow(Form1.hWnd, 0, 250)

**See also :** [Windows](#)

# GetChangeTaskName

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetChangeTaskName](#) get and change the name of the task. You see change in the Task Manager by pressing the CTRL + ESC keys.

## Declare Syntax :

Declare Function cGetChangeTaskName Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal Text As String) As String

## Call Syntax :

```
test$ = cGetChangeTaskName(Form.hWnd, Text)
```

## Where :

Form.hWnd	is the hWnd of your application
Text	is the new task name to given at your application
test\$	is the old task name of the application

## Comments :

This is useful to set a particular task name at your application and backups the old task name. This function is a mix of [cGetTaskName](#) and [cChangeTaskName](#).

## Examples :

```
Dim OldTaskName As String
```

```
OldTaskName = cGetChangeTaskName(Me.hWnd, "Hello world")
```

```
MsgBox OldTaskName
```

```
    ' press the CTRL + ESC keys to see the change in the Task Manager
```

```
    ' OldTaskName is "Microsoft Visual Basic"
```

```
' if you repeat the test
```

```
    ' OldTaskName is "Hello world"
```

**See also :** [Task - File version](#)

# TaskBarAddIcon, TaskBarDeletelcon, TaskBarModifylcon

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[TaskBarAddIcon](#) add an icon for an application in the tray of the task bar.

[TaskBarDeletelcon](#) delete the tray icon from an application in the task bar.

[TaskBarModifylcon](#) modify an icon for an application in the tray of the task bar.

## Declare Syntax :

Declare Function cTaskBarAddIcon Lib "time2win.dll" (ByVal hWnd As Long, ByVal hIcon As Long, ByVal lpszTip As String) As Integer

Declare Function cTaskBarDeletelcon Lib "time2win.dll" (ByVal hWnd As Long) As Integer

Declare Function cTaskBarModifylcon Lib "time2win.dll" (ByVal hWnd As Long, ByVal hIcon As Long, ByVal lpszTip As String) As Integer

## Call Syntax :

```
intResult% = cTaskBarAddIcon(hWnd&, hIcon&, lpszTip$)
```

```
intResult% = cTaskBarDeletelcon(hWnd&)
```

```
intResult% = cTaskBarModifylcon(hWnd&, hIcon&, lpszTip$)
```

## Where :

hWnd& is the .hWnd property of the form used to performe operation in the tray on task bar.

hIcon& is the .Icon property of the form used to performe operation in the tray on task bar.

lpszTip\$ is the tooltip message to display when the mouse moves over the icon in the tray

## Comments :

Don't forget to call cTaskBarDeletelcon when your application end.

Beware when you use CTRL+BREAK to stop your application.

Beware when you use END statement to stop your application.

## Examples :

in the Form\_Load event :

```
debug.print cTaskBarAddIcon(Me.hWnd, Me.Icon., "Form1 loaded")
```

in the Form\_Resize event :

```
debug.print cTaskBarModifylcon(Me.hWnd, Me.Icon., "Form1 minimized")
```

in the Form\_QueryUnload event :

```
debug.print cTaskBarDeletelcon(Me.hWnd)
```

**See also** : [Windows 95](#)

# GetClassName

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetClassName](#) retrieve the full class name of a window.

## Declare Syntax :

Declare Function cGetClassName Lib "[time2win.dll](#)" (ByVal hWnd As Long) As String

## Call Syntax :

```
test$ = cGetClassName(hWnd)
```

## Where :

hWnd is the .hWnd of a control.  
test\$ is the returned class name.

## Comments :

if the .hWnd is not exist, the returned string is an EMPTY string.

## Examples :

```
test$ = cGetClassName(Me.hWnd)           -> "ThunderForm"  
test$ = cGetClassName(Command1.hWnd)    -> "ThunderCommandButton"  
test$ = cGetClassName(List1.hWnd)       -> "ThunderListBox"  
test$ = cGetClassName(Text1.hWnd)       -> "ThunderTextBox"
```

**See also :** [Windows](#)

# EXEnameActiveWindow

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[EXEnameActiveWindow](#) retrieve the full filename (path and file) of the active window.

## **Declare Syntax :**

Declare Function cEXEnameActiveWindow Lib "[time2win.dll](#)" () As String

## **Call Syntax :**

```
test$ = cEXEnameActiveWindow()
```

## **Where :**

test\$                      is the name of the active window

## **Comments :**

## **Examples :**

```
test$ = cEXEnameActiveWindow()
```

On my system : test\$ = "K:\WIN95\VB\VB.EXE"

**See also :** [Windows](#)

# EXEnameWindow

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[EXEnameWindow](#) retrieve the full filename (path and file) of the specified window.

## **Declare Syntax :**

```
Declare Function cEXEnameWindow Lib "time2win.dll" (ByVal hModule As Integer) As String
```

## **Call Syntax :**

```
test$ = cEXEnameWindow(Form.Hwnd)
```

## **Where :**

hModule            is the hWnd of the window  
test\$                is the name of the specified window

## **Comments :**

## **Examples :**

```
test$ = cEXEnameWindow(Me.hWnd)
```

On my system : test\$ = "K:\WIN95\VB\VB.EXE"

**See also :** [Windows](#)

# EXEnameTask

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[EXEnameTask](#) retrieve the full path and filename of the executable file from which the specified module was loaded.

## **Declare Syntax :**

Declare Function cEXEnameTask Lib "[time2win.dll](#)" (ByVal nFileName As String) As String

## **Call Syntax :**

```
test$ = cEXEnameTask(nFileName)
```

## **Where :**

nFileName	is the task name as you fin when pressing CTRL + ESC keys
test\$	is the returned full path and filename

## **Comments :**

## **Examples :**

```
test$ = cEXEnameTask("PROGMAN")
```

On my system : test\$ = "K:\WIN95\PROGMAN.EXE"

**See also :** [Windows](#)



# ExitWindowsAndExecute, RebootSystem, RestartWindows

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 \(Win95/WinNT\)](#), [MSOffice 95](#)

## Purpose :

[ExitWindowsAndExecute](#) terminate Windows, runs a specified MS-DOS application, and then restarts Windows.  
[RebootSystem](#) reboot your system.  
[RestartWindows](#) restart your Windows.

## Declare Syntax :

```
Declare Function cExitWindowsAndExecute Lib "time2win.dll" (ByVal IpszExe As String, ByVal IpszParams As String) As Integer
Declare Function cRebootSystem Lib "time2win.dll" () As Integer
Declare Function cRestartWindows Lib "time2win.dll" () As Integer
```

## Call Syntax :

```
test% = cExitWindowsAndExecute(IpszExe, IpszParams)
test% = cRebootSystem()
test% = cRestartWindows()
```

## Where :

IpszExe	is the program to launch after exiting Windows.
IpszParams	are the associated parameter to pass to the program.
test%	= 0 if one or more applications refuse to terminate.

## Comments :

The ExitWindowsAndExecute function is typically used by installation programs to replace components of Windows which are active when Windows is running.

## Examples :

```
test% = cExitWindowsAndExecute("MENU.EXE", "/Z/V/C")
test% = cRebootSystem()
test% = cRestartWindows()
```

**See also** : [Windows](#)

# GetDefaultCurrentDir

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetDefaultCurrentDir](#) retrieve the current dir on the current drive.

## **Declare Syntax :**

```
Declare Function cGetDefaultCurrentDir Lib "time2win.dll" () As String
```

## **Call Syntax :**

```
test$ = cGetDefaultCurrentDir()
```

## **Where :**

```
test$            the dir
```

## **Comments :**

The GetDefaultCurrentDir function gets the full path of the current working directory for the default drive . The integer  
The GetDefaultCurrentDir function returns a string that represents the path of the current working directory. If the current working directory is set to the root, the string will end with a backslash ( \ ). If the current working directory is set to a directory other than the root, the string will end with the name of the directory and not with a backslash.

## **Examples :**

**See also :** [Windows](#)

# GetDefaultPrinter

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetDefaultPrinter](#) return the default printer in the [windows] section of Win.INI

## **Declare Syntax :**

Declare Function cGetDefaultPrinter Lib "[time2win.dll](#)" () As String

## **Call Syntax :**

```
test$ = cGetDefaultPrinter()
```

## **Where :**

test\$                    is the default printer

## **Comments :**

## **Examples :**

```
test$ = cGetDefaultPrinter()                    -> "HP LASERJET III, HPPCL5MS, LPT1:"
```

**See also :** [Windows](#)

# GetDevices

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetDevices](#) return all devices founden in the [devices] section in the Win.INI

## **Declare Syntax :**

Declare Function cGetDevices Lib "[time2win.dll](#)" () As String

## **Call Syntax :**

test\$ = cGetDevices()

## **Where :**

test\$                      all devices separated by a chr\$(13).

## **Comments :**

Use the cGetIn function to extract each device.

## **Examples :**

test\$ = cGetDevices()                      -> "HP LaserJet III=HPPCL5MS,LPT1:"

**See also :** [Windows](#)

# GetDriveCurrentDir

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetDriveCurrentDir](#) retrieve the current dir on the specified drive.

## Declare Syntax :

Declare Function cGetDriveCurrentDir Lib "[time2win.dll](#)" (ByVal lpDrive As String) As String

## Call Syntax :

```
test$ = cGetDefaultCurrentDir(lpDrive)
```

## Where :

lpDrive	the letter for the drive
test\$	the dir

## Comments :

The GetDriveCurrentDir function gets the full path of the current working directory on the specified drive  
The GetDriveCurrentDir function returns a string that represents the path of the current working directory on the specified drive. If the current working directory is set to the root, the string will end with a backslash (\). If the current working directory is set to a directory other than the root, the string will end with the name of the directory and not with a backslash.

If the disk is not present or if the disk is not available or if an error occurs when accessing the disk, the returned value is always an EMPTY string.

This function works with local disk (hard, floppy or cd-rom) als well on remote disk (network).

## Examples :

**See also :** [Windows](#)

# ComboSearchDir, ComboSearchFile, ComboSearchFileAttrib ListSearchDir, ListSearchFile, ListSearchFileAttrib

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[ComboSearchDir](#) perform a directory pattern match starting with a specified path and fill a standard combo box.

[ComboSearchFile](#) perform a file pattern match starting with a specified path and fill a standard combo box.

[ComboSearchFileAttrib](#) perform a file (with attribute) pattern match starting with a specified path and fill a standard combo box.

[ListSearchDir](#) perform a directory pattern match starting with a specified path and fill a standard list box.

[ListSearchFile](#) perform a file pattern match starting with a specified path and fill a standard list box.

[ListSearchFileAttrib](#) perform a file (with attribute) pattern match starting with a specified path and fill a standard list box.

## **Declare Syntax :**

Declare Function cComboSearchDir Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal lpStartPath As String, ByVal lpFileMask As String) As Long

Declare Function cComboSearchFile Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal lpStartPath As String, ByVal lpFileMask As String) As Long

Declare Function cComboSearchFileAttrib Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal lpStartPath As String, ByVal lpFileMask As String, ByVal iSearchAttribute As Integer) As Long

Declare Function cListSearchDir Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal lpStartPath As String, ByVal lpFileMask As String) As Long

Declare Function cListSearchFile Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal lpStartPath As String, ByVal lpFileMask As String) As Long

Declare Function cListSearchFileAttrib Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal lpStartPath As String, ByVal lpFileMask As String, ByVal iSearchAttribute As Integer) As Long

## **Call Syntax :**

lngResult& = cComboSearchDir(hWnd&, lpStartPath\$, lpFileMask\$)

lngResult& = cComboSearchFile(hWnd&, lpStartPath\$, lpFileMask\$)

lngResult& = cComboSearchFileAttrib(hWnd&, lpStartPath\$, lpFileMask\$, iSearchAttribute%)

lngResult& = cListSearchDir(hWnd&, lpStartPath\$, lpFileMask\$)

lngResult& = cListSearchFile(hWnd&, lpStartPath\$, lpFileMask\$)

lngResult& = cListSearchFileAttrib(hWnd&, lpStartPath\$, lpFileMask\$, iSearchAttribute%)

## **Where :**

hWnd& is the .hWnd property of a standard list or combo box.

lpStartPath\$ is the starting path to begin the search.

lpFileMask\$ is the file mask to match.

iSearchAttribute% is the file attribute mask (see [attributes](#))

## **Comments :**

For the pattern match, see [PatternMatch](#).

The search for directory/file is case-insensitive.

## **Examples :**

```
debug.print cComboSearchDir(Combo1.hWnd, "c:\", "t2win-32")  
debug.print cComboSearchFile(Combo1.hWnd, "c:\", "t2win-32.dll")
```

```
debug.print cComboSearchFileAttrib(Combo1.hWnd, "c:\", "t2win-32.dll", A_ARCHIVE Or A_HIDDEN)
debug.print cListSearchDir(List1.hWnd, "c:\", "t2win-32")
debug.print cListSearchFile(List1.hWnd, "c:\", "t2win-32.dll")
debug.print cListSearchFileAttrib(Combo1.hWnd, "c:\", "t2win-32.dll", A_ARCHIVE Or A_HIDDEN)
```

**See also :** [List box - combo box](#)

# ComboFiles, ListFiles

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[ComboFiles](#) fill a Combo Box with files with the specified attribute and mask.

[ListFiles](#) fill a List Box with files with the specified attribute and mask.

## **Declare Syntax :**

Declare Function cComboFiles Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal Attributes As Long, ByVal FilePathMask As String) As Integer

Declare Function cListFiles Lib "[time2win.dll](#)" (ByVal hWnd As Long, ByVal Attributes As Long, ByVal FilePathMask As String) As Integer

## **Call Syntax :**

## **Where :**

## **Comments :**

## **Examples :**

**See also :** [List box - combo box](#)



# ListSetTabs

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

**Purpose :**

[ListSetTabs](#) set tabulation in a List Box.

**Declare Syntax :**

Declare Function cListSetTabs Lib "[time2win.dll](#)" (ByVal hWnd As Long, TabArray() As Long) As Integer

**Call Syntax :**

**Where :**

**Comments :**

**Examples :**

**See also :** [List box - combo box](#)

## Task - File version : Overview

<a href="#">ChangeTaskName</a>	change the name of the task.
<a href="#">GetChangeTaskName</a>	get and change the name of the task.
<a href="#">GetFileVersion</a>	return a partial information over a specified file.
<a href="#">GetFileVersionInfo</a>	return a full information over a specified file in one call.
<a href="#">GetTaskName</a>	read the name of the task.
<a href="#">ModuleFind</a>	retrieves some parameters for a specified loaded module.
<a href="#">Module</a>	retrieves each loaded module one by one.
<a href="#">Process</a>	retrieves each process one by one.
<a href="#">Thread</a>	retrieves each thread one by one.

# GetFileVersion

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetFileVersion](#) return a partial information over a specified file.

## Declare Syntax :

Declare Function cGetFileVersion Lib "[time2win.dll](#)" (ByVal filename As String, ByVal nFonction As Integer) As String

## Call Syntax :

```
test$ = cGetFileVersion(filename, nFonction)
```

## Where :

filename            is the file to proceed  
nFonction            is the partial information to retrieve.  
test\$                is the returned information

## Comments :

The returned information can be an EMPTY string if the partial informations don't exists.

## Examples :

```
Dim i                As Integer  
Dim Tmp             As String
```

```
For i = VER_VERSION_PRODUCT To VER_PRODUCT_VERSION  
  Tmp = Tmp & i & " = " & cGetFileVersion("k:\windows\progman.exe", i) & Chr$(13)  
Next i
```

```
MsgBox Tmp
```

' On my system :

```
' -1 = 3.10.0.103  
' 0 = 3.10.0.103  
' 1 = Microsoft Corporation  
' 2 = Windows Program Manager application file  
' 3 = 3.10  
' 4 = PROGMAN  
' 5 = Copyright © Microsoft Corp. 1991-1992  
' 6 =  
' 7 =  
' 8 = Microsoft® Windows(TM) Operating System
```

**See also :** [Task - File version](#)

# GetFileVersionInfo

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[GetFileVersionInfo](#) return a full information over a specified file in one call.

## Declare Syntax :

Declare Function cGetFileVersionInfo Lib "[time2win.dll](#)" (ByVal filename As String, FILEVERSIONINFO As Any) As Integer

## Call Syntax :

test% = cGetFileVersion(filename, FILEVERSIONINFO)

## Where :

filename is the file to proceed  
FILEVERSIONINFO is a typed variable '[tagFILEVERSIONINFO](#)' which receives the full information  
test% TRUE if all is Ok  
FALSE if an error has occurred

## Comments :

## Examples :

```
Dim status As Integer  
Dim FILEVERSIONINFO As tagFILEVERSIONINFO
```

```
status = cGetFileVersionInfo("k:\windows\system\knl386.exe", FILEVERSIONINFO)
```

```
Debug.Print "FILEVERSIONINFO.VersionProduct = " & FILEVERSIONINFO.VersionProduct  
Debug.Print "FILEVERSIONINFO.FileDescription = " & FILEVERSIONINFO.FileDescription  
Debug.Print "FILEVERSIONINFO.FileVersion = " & FILEVERSIONINFO.FileVersion  
Debug.Print "FILEVERSIONINFO.InternalName = " & FILEVERSIONINFO.InternalName  
Debug.Print "FILEVERSIONINFO.LegalCopyright = " & FILEVERSIONINFO.LegalCopyright  
Debug.Print "FILEVERSIONINFO.LegalTrademarks = " & FILEVERSIONINFO.LegalTrademarks  
Debug.Print "FILEVERSIONINFO.Comments = " & FILEVERSIONINFO.Comments  
Debug.Print "FILEVERSIONINFO.ProductName = " & FILEVERSIONINFO.ProductName  
Debug.Print "FILEVERSIONINFO.ProductVersion = " & FILEVERSIONINFO.ProductVersion
```

' On my system :

```
' FILEVERSIONINFO.VersionProduct = 3.11.0.300  
' FILEVERSIONINFO.FileDescription = Windows Kernel  
' FILEVERSIONINFO.FileVersion = 3.11  
' FILEVERSIONINFO.InternalName = KRNL386  
' FILEVERSIONINFO.LegalCopyright = Copyright © Microsoft Corp. 1991-1993  
' FILEVERSIONINFO.LegalTrademarks =  
' FILEVERSIONINFO.Comments =  
' FILEVERSIONINFO.ProductName = Microsoft® Windows(TM) Operating System  
' FILEVERSIONINFO.ProductVersion = 3.11
```

**See also :** [Task - File version](#)

' definition for file version information

Public Const VER\_VERSION\_PRODUCT = -1

Public Const VER\_VERSION\_FILE = 0

Public Const VER\_COMPANY\_NAME = 1

Public Const VER\_FILE\_DESCRIPTION = 2

Public Const VER\_FILE\_VERSION = 3

Public Const VER\_INTERNAL\_NAME = 4

Public Const VER\_LEGAL\_COPYRIGHT = 5

Public Const VER\_LEGAL\_TRADEMARKS = 6

Public Const VER\_PRODUCT\_NAME = 7

Public Const VER\_PRODUCT\_VERSION = 8

' structure for file version information

```
Type tagFILEVERSIONINFO
  VersionProduct      As String
  VersionFile         As String
  CompanyName         As String
  FileDescription     As String
  FileVersion         As String
  InternalName        As String
  LegalCopyright      As String
  LegalTrademarks     As String
  Comments            As String
  ProductName         As String
  ProductVersion      As String
End Type
```

# WalkThruWindow

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[WalkThruWindow](#) walk in the window's list of all windows at a given moment.

## Declare Syntax :

Declare Function cWalkThruWindow Lib "time2win.dll" (Class As String, Caption As String, OwnerHwnd As Integer, OwnerClass As String, OwnerCaption As String, ByVal FirstNext As Integer) As Integer

## Call Syntax :

hWnd% = cWalkThruWindow(Class\$, Caption\$, OwnerHwnd%, OwnerClass\$, OwnerCaption\$, FirstNext%)

## Where :

Class\$	is the returned Name of the Window's Class for the hWnd founded.
Caption\$	is the returned Caption of the Window for the hWnd founded.
OwnerHwnd%	is the returned hWnd of the Owner for the hWnd founded
OwnerClass\$	is the returned Name of the Window's Class for the Owner for the hWnd founded.
OwnerCaption\$	is the returned Caption of the Window for the Owner for the hWnd founded.
FirstNext%	TRUE to begin the search, FALSE to continue the search.
hWnd%	is the returned hWnd founded.

## Comments :

## Examples :

```
Dim nClass           As String
Dim nCaption         As String
Dim nOwnerClass     As String
Dim nOwnerCaption   As String
Dim nOwnerHwnd      As Integer
```

```
Dim nhWnd           As Integer
```

```
nhWnd = cWalkThruWindow(nClass, nCaption, nOwnerHwnd, nOwnerClass, nOwnerCaption, True)
```

```
Do While (nhWnd <> 0)
    Debug.Print "Owner   = "; Hex$(nOwnerHwnd) & Chr$(9) & nOwnerCaption & " (" & nOwnerClass & ")"
    Debug.Print "Window = "; Hex$(nhWnd) & Chr$(9) & nCaption & " (" & nClass & ")"
    nhWnd = cWalkThruWindow(nClass, nCaption, nOwnerHwnd, nOwnerClass, nOwnerCaption, False)
Loop
```

' Part of the output on my system :

```
' Owner       = 42A4      Microsoft Visual Basic (ThunderMain)
' Window = 41BC      Time To WIN (Demo) (ThunderForm)
' Owner       = 42A4      Microsoft Visual Basic (ThunderMain)
' Window = 5878      (ToolsPalette)
' Owner       = 42A4      Microsoft Visual Basic (ThunderMain)
' Window = 56D4      TIME2WIN.MAK (PROJECT)
' Owner       = 42A4      Microsoft Visual Basic (ThunderMain)
' Window = 5B20      Debug Window [TIME2WIN.FRM] (OFEDT)
' Owner       = 42A4      Microsoft Visual Basic (ThunderMain)
```

' Window = 48AC	Microsoft Visual Basic [run] (wndclass_desked_gsk)
' Owner = 4A68	Properties (wndclass_pbrs)
' Window = 59A8	(CBar)
' Owner = 42A4	Microsoft Visual Basic (ThunderMain)
' Window = 4A68	Properties (wndclass_pbrs)
' Owner = 42A4	Microsoft Visual Basic (ThunderMain)
' Window = 5928	(CPal)
' Owner = 0	()
' Window = 42A4	Microsoft Visual Basic (ThunderMain)

**See also :** [Windows](#)



# ModuleFind

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ModuleFind](#) retrieve some parameters for a specified loaded module.

## Declare Syntax :

Declare Function cModuleFind Lib "[time2win.dll](#)" (MODULEENTRY As Any, ByVal ModuleName As String) As Integer

## Call Syntax :

test% = cModuleFind(MODULEENTRY, ModuleName)

## Where :

ModuleName	is the module to proceed
MODULEENTRY	is the type'd variable ' <a href="#">tagMODULEENTRY</a> ' which receives the parameters.
test%	TRUE if all is Ok FALSE if an error has occurred

## Comments :

dwSize	Specifies the size of the MODULEENTRY structure, in bytes.
szModule	Specifies the null-terminated string that contains the module name.
hModule	Identifies the module handle.
wcUsage	Specifies the reference count of the module. This is the same number returned by the GetModuleUsage function.
szExePath	Specifies the null-terminated string that contains the fully-qualified executable path for the module.
wNext	Specifies the next module in the module list. This member is reserved for internal use by Windows.

## Examples :

```
Dim status As Integer
Dim MODULEENTRY As tagMODULEENTRY
```

```
status = cModuleFind(MODULEENTRY, "KERNEL")
```

```
Debug.Print "MODULEENTRY.dwSize = " & MODULEENTRY.dwSize
Debug.Print "MODULEENTRY.szModule = " & MODULEENTRY.szModule
Debug.Print "MODULEENTRY.hModule = " & MODULEENTRY.hModule
Debug.Print "MODULEENTRY.wcUsage = " & MODULEENTRY.wcUsage
Debug.Print "MODULEENTRY.szExePath = " & MODULEENTRY.szExePath
Debug.Print "MODULEENTRY.wNext = " & MODULEENTRY.wNext
```

' On my system :

```
' MODULEENTRY.dwSize = 276
' MODULEENTRY.szModule = KERNEL
' MODULEENTRY.hModule = 295
' MODULEENTRY.wcUsage = 44
' MODULEENTRY.szExePath = K:\WIN95\SYSTEM\KRNL386.EXE
' MODULEENTRY.wNext = 279
```

**See also :** [Task - File version](#)

# Module, Process, Thread

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**Module** enumerates the modules residing in the system address space.

**Process** enumerates the processes residing in the system address space.

**Thread** enumerates the threads residing in the system address space.

## Declare Syntax :

Declare Function cModule Lib "t2w32mpt.dll" (MODULEENTRY As tagMODULEENTRY, ByVal FirstNext As Integer) As Integer

Declare Function cProcess Lib "t2w32mpt.dll" (PROCESSENTRY As tagPROCESSENTRY, ByVal FirstNext As Integer) As Integer

Declare Function cThread Lib "t2w32mpt.dll" (THREADENTRY As tagTHREADENTRY, ByVal FirstNext As Integer) As Integer

## Call Syntax :

intResult% = cModule(MODULEENTRY, FirstNext%)

intResult% = cProcess(PROCESSENTRY, FirstNext%)

intResult% = cThread(THREADENTRY, FirstNext%)

## Where :

MODULEENTRY	is the type'd variable 'tagMODULEENTRY' which receives the parameters.
FirstNext%	TRUE for the first module/process/thread FALSE for each next module/process/thread
intResult%	TRUE if all is Ok FALSE if an error has occured or if no more modules.

## Comments :

### For MODULEENTRY :

dwSize	Specifies the size of the MODULEENTRY structure, in bytes.
szModule	Specifies the null-terminated string that contains the module name.
hModule	Identifies the module handle.
wcUsage	Specifies the reference count of the module. This is the same number returned by the GetModuleUsage function.
szExePath	Specifies the null-terminated string that contains the fully-qualified executable path for the module.
wNext	Specifies the next module in the module list. This member is reserved for internal use by Windows.

### For PROCESSENTRY :

dwSize	Specifies the size of the PROCESSENTRY structure, in bytes.
cntUsage	Number of references to the process. A process exists as long as its usage count is nonzero. As soon as its usage count becomes zero, a process terminates.
th32ProcessID	Identifier of the process. The contents of this member can be used by Win32 API elements.
th32DefaultHeapID	Identifier of the default heap for the process. The contents of this member has meaning only to the tool help functions. It is not a handle, nor is it usable by Win32 API elements.
th32ModuleID	Module identifier of the process. The contents of this member has meaning only to the tool help functions. It is not a handle, nor is it usable by Win32 API elements.
cntThreads	Number of execution threads started by the process.
th32ParentProcessID	Identifier of the process that created the process being examined. The contents of this member can be used by Win32 API elements.
pcPriClassBase	Base priority of any threads created by this process.
dwFlags	Reserved; do not use.
szExeFile	Path and filename of the executable file for the process.

## For THREADENTRY :

dwSize	Specifies the size of the THREADENTRY structure, in bytes.
cntUsage	Number of references to the thread. A thread exists as long as its usage count is nonzero. As soon as its usage count becomes zero, a thread terminates.
th32ThreadID	Identifier of the thread. This identifier is compatible with the thread identifier returned by the CreateProcess function.
th32OwnerProcessID	Identifier of the process that created the thread. The contents of this member can be used by Win32 API elements.
tpBasePri	Initial priority level assigned to a thread. These values are defined: THREAD_PRIORITY_IDLE Indicates a base priority level of 1 for IDLE_PRIORITY_CLASS, NORMAL_PRIORITY_CLASS, or HIGH_PRIORITY_CLASS processes, and a base priority level of 16 for REALTIME_PRIORITY_CLASS processes. THREAD_PRIORITY_LOWEST Indicates 2 points below normal priority for the priority class. THREAD_PRIORITY_BELOW_NORMAL Indicates 1 point below normal priority for the priority class. THREAD_PRIORITY_NORMAL Indicates normal priority for the priority class. THREAD_PRIORITY_ABOVE_NORMAL Indicates 1 point above normal priority for the priority class. THREAD_PRIORITY_HIGHEST Indicates 2 points above normal priority for the priority class. THREAD_PRIORITY_TIME_CRITICAL Indicates a base priority level of 15 for IDLE_PRIORITY_CLASS, NORMAL_PRIORITY_CLASS, or HIGH_PRIORITY_CLASS processes, and a base priority level of 31 for REALTIME_PRIORITY_CLASS processes.
tpDeltaPri	Change in the priority level of a thread. This value is a signed delta from the base priority level assigned to the thread.
dwFlags	Reserved; do not use.

## **Examples :**

see samples in the demo.

**See also :** [Task - File version](#)

# Tasks

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

Tasks retrieves all tasks currently in memory.

## Declare Syntax :

Declare Function cTasks Lib "time2win.dll" (TASKENTRY As Any, ByVal firstnext As Integer) As Integer

## Call Syntax :

test% = cTasks(TASKENTRY, firstnext)

## Where :

TASKENTRY	is the typed variable which receives the parameters 'tagTASKENTRY'
firstnext	TRUE for the first module FALSE for each next module
test%	TRUE if all is Ok FALSE if an error has occurred or if no more tasks

## Comments :

The hTask parameter is the task number founded by the cModuleFind or cModules functions.

dwSize	Specifies the size of the TASKENTRY structure, in bytes.
hTask	Identifies the task handle for the stack.
hTaskParent	Identifies the parent of the task.
hInst	Identifies the instance handle of the task. This value is equivalent to the task's DGROUP segment selector.
hModule	Identifies the module that contains the currently executing function.
wSS	Contains the value in the SS register.
wSP	Contains the value in the SP register.
wStackTop	Specifies the offset to the top of the stack (lowest address on the stack).
wStackMinimum	Specifies the lowest segment number of the stack during execution of the task.
wStackBottom	Specifies the offset to the bottom of the stack (highest address on the stack).
wcEvents	Specifies the number of pending events.
hQueue	Identifies the task queue.
szModule	Specifies the name of the module that contains the currently executing function.
wPSPOffset	Specifies the offset from the program segment prefix (PSP) to the beginning of the executable code segment.
hNext	Identifies the next entry in the task list. This member is reserved for internal use by Windows.

## Examples :

```
Dim status As Integer
Dim TASKENTRY As tagTASKENTRY
```

```
Close #1
Open "c:\tmp.tmp" For Output Shared As #1
```

```
Print #1, "dwSize"; Chr$(9);
Print #1, "hTask"; Chr$(9);
Print #1, "hTaskParent"; Chr$(9);
Print #1, "hInst"; Chr$(9);
Print #1, "hModule"; Chr$(9);
Print #1, "wSS"; Chr$(9);
Print #1, "wSP"; Chr$(9);
```

```

Print #1, "wStackTop"; Chr$(9);
Print #1, "wStackMinimum"; Chr$(9);
Print #1, "wStackBottom"; Chr$(9);
Print #1, "wcEvents"; Chr$(9);
Print #1, "hQueue"; Chr$(9);
Print #1, "szModule"; Chr$(9);
Print #1, "wPSPOffset"; Chr$(9);
Print #1, "hNext"; Chr$(13)

```

```

status = cTasks(TASKENTRY, True)
Do While (status = True)

```

```

    Print #1, TASKENTRY.dwSize; Chr$(9);
    Print #1, TASKENTRY.hTask; Chr$(9);
    Print #1, TASKENTRY.hTaskParent; Chr$(9);
    Print #1, TASKENTRY.hInst; Chr$(9);
    Print #1, TASKENTRY.hModule; Chr$(9);
    Print #1, TASKENTRY.wSS; Chr$(9);
    Print #1, TASKENTRY.wSP; Chr$(9);
    Print #1, TASKENTRY.wStackTop; Chr$(9);
    Print #1, TASKENTRY.wStackMinimum; Chr$(9);
    Print #1, TASKENTRY.wStackBottom; Chr$(9);
    Print #1, TASKENTRY.wcEvents; Chr$(9);
    Print #1, TASKENTRY.hQueue; Chr$(9);
    Print #1, TASKENTRY.szModule; Chr$(9);
    Print #1, TASKENTRY.wPSPOffset; Chr$(9);
    Print #1, TASKENTRY.hNext

```

```

    status = cTasks(TASKENTRY, False)

```

```

Loop

```

```

Close #1

```

On my system :

dwSize	hTask	hTaskParent	hInst	hModule	wSS	wSP	wStackTop	wStackMinimum	wStackBottom	wcEvents	hQueue	szModule	wPSPOffset	hNext
40	4231	1783	8246	4367		8247	-27238	30418						-28190
	27076			8263	ICONBAR		8279		4439	0				
40	4439	1783	4398	4463		4399	5850	1022						5992
	5992			4471	WINEXIT		4447		16279	0				
40	16279	4231	15878	16295		15879	-4188	-23384						10032
	-4054			16255	MSVC		16271		2087	0				
40	2087	1783	8030	2095		8031	29198	9004						29334
	29334			8047	FASTLOAD		8063		1783	0				
40	1783	335	5846	1799		5847	8202	2358						5950
	8304			2079	PROGMAN		791		7007	0				
40	7007	4231	9926	6767		9927	-23760	13124						23498
	-23562			6879	FOREHELP		6903		4431	1				
40	4431	1783	4278	4455		4279	7654	2844						6998
	7814			4359	FREEMEM		4375		12127	1				
40	12127	1783	9022	12143		9023	-29164	16534						-31948
	28672			9039	VB		9231		0	0				

See also : [Task - File version](#)

# TaskFind

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

`TaskFind` retrieve some parameters for a specified loaded task.

## Declare Syntax :

Declare Function `cTaskFind` Lib "time2win.dll" (TASKENTRY As Any, ByVal hTask As Integer) As Integer

## Call Syntax :

`test%` = `cTaskFind`(TASKENTRY, hTask)

## Where :

<code>hTask</code>	is the task number
TASKENTRY	is the typed variable which receives the parameters 'tagTASKENTRY'
<code>test%</code>	TRUE if all is Ok FALSE if an error has occurred

## Comments :

The `hTask` parameter is the task number founded by the `cModuleFind` or `cModules` functions.

<code>dwSize</code>	Specifies the size of the TASKENTRY structure, in bytes.
<code>hTask</code>	Identifies the task handle for the stack.
<code>hTaskParent</code>	Identifies the parent of the task.
<code>hInst</code>	Identifies the instance handle of the task. This value is equivalent to the task's DGROUP segment selector.
<code>hModule</code>	Identifies the module that contains the currently executing function.
<code>wSS</code>	Contains the value in the SS register.
<code>wSP</code>	Contains the value in the SP register.
<code>wStackTop</code>	Specifies the offset to the top of the stack (lowest address on the stack).
<code>wStackMinimum</code>	Specifies the lowest segment number of the stack during execution of the task.
<code>wStackBottom</code>	Specifies the offset to the bottom of the stack (highest address on the stack).
<code>wcEvents</code>	Specifies the number of pending events.
<code>hQueue</code>	Identifies the task queue.
<code>szModule</code>	Specifies the name of the module that contains the currently executing function.
<code>wPSPOffset</code>	Specifies the offset from the program segment prefix (PSP) to the beginning of the executable code segment.
<code>hNext</code>	Identifies the next entry in the task list. This member is reserved for internal use by Windows.

## Examples :

```
Dim status As Integer
Dim MODULEENTRY As tagMODULEENTRY
```

```
status = cModuleFind(MODULEENTRY, "KERNEL")
```

```
Debug.Print "MODULEENTRY.dwSize = " & MODULEENTRY.dwSize
Debug.Print "MODULEENTRY.szModule = " & MODULEENTRY.szModule
Debug.Print "MODULEENTRY.hModule = " & MODULEENTRY.hModule
Debug.Print "MODULEENTRY.wcUsage = " & MODULEENTRY.wcUsage
Debug.Print "MODULEENTRY.szExePath = " & MODULEENTRY.szExePath
Debug.Print "MODULEENTRY.wNext = " & MODULEENTRY.wNext
```

' On my system :

```
' MODULEENTRY.dwSize = 276
' MODULEENTRY.szModule = KERNEL
' MODULEENTRY.hModule = 295
' MODULEENTRY.wcUsage = 44
' MODULEENTRY.szExePath = K:\WIN95\SYSTEM\KRNL386.EXE
' MODULEENTRY.wNext = 279
```

**See also :** [Task - File version](#)





' structure for modules

Type tagMODULEENTRY

dwSize	As Long
th32ModuleID	As Long
th32ProcessID	As Long
GblcntUsage	As Long
ProccntUsage	As Long
modBaseAddr	As Byte
modBaseSize	As Long
hModule	As Long
szModule	As String * 256
szExePath	As String * 260

End Type

' constants for Mime Headers in cMNEncode

Public Const UUVBE\_NONE = 0

Public Const UUVBE\_SIMPLE = 1

Public Const UUVBE\_MAIL = 2

Public Const UUVBE\_NEWS = 3

' No MIME headers (use standard headers)

' Simple MIME (no addressing)

' MIME E-mail format

' MIME News format

# FilesInfoInDir

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FilesInfoInDir](#) retrieve each file in the specified directory and returns name, size, Int date, Int time, attribute.

## Declare Syntax :

Declare Function cFilesInfoInDir Lib "[time2win.dll](#)" (ByVal nDir As String, FILEINFO As tagFILEINFO, ByVal FirstNext As Integer) As String

## Call Syntax :

test\$ = cFilesInfoInDir(nDir, FI, firstnext )

## Where :

nDir	the directory to proceed with the file mask (*. * for all)
FI	the type'd variable ' <a href="#">tagFILEINFO</a> '.
firstnext	TRUE for the first file FALSE for each next file
test\$	the returned file

## Comments :

If the nDir is invalid or if an error occurs when accessing a file, the returned filename is an empty string and all sub-variables in the type'd variable are -1.

## Examples :

```
Dim i           As Integer
Dim Tmp        As String
Dim FI         As tagFILEINFO
```

```
i = 0
Tmp = cFilesInfoInDir("c:\*.*", FI, True)
```

```
Debug.Print "The first 7 files in C:\ are : "
```

```
Do While (Len(Tmp) > 0)
  Debug.Print Tmp & ", " & FI.fSize & ", " & FI.fDate & ", " & FI.fTime & ", " & FI.fAttribute
  Tmp = cFilesInfoInDir("c:\*.*", FI, False)
  i = i + 1
  If (i >= 7) Then Exit Do
Loop
```

```
' On my system:
```

```
' The first 7 files in C:\ are :
```

```
' SUHDLOG.DAT, 5166, 728480, 76033, 3
' BOOTLOG.TXT, 22886, 728480, 78500, 2
' MSDOS.---, 22, 728480, 75079, 2
' DBLSPACE.001, 79036439, 728519, 48662, 7
' SYSTEM.1ST, 230144, 728480, 76027, 7
' WINA20.386, 9349, 727632, 21600, 0
' AUTOEXEC.BAK, 968, 728456, 78015, 0
```

**See also :** [File](#)



' structure for File Information

Type tagFILEINFO

fSize	As Long	'size of the file
fDate	As Long	'date of the file (Int date)
fTime	As Long	'time of the file (Int time)
fAttribute	As Integer	'attribute of the file

End Type

# CenterWindow

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[CenterWindow](#) center a window in the screen.

## **Declare Syntax :**

Declare Sub cCenterWindow Lib "[time2win.dll](#)" (ByVal hWnd As Long)

## **Call Syntax :**

Call cCenterWindow(hWnd%)

## **Where :**

hWnd% is the handle of a form.

## **Comments :**

## **Examples :**

Call cCenterWindow(Form1.hWnd)

**See also :** [Windows](#)

# ArrangeDesktopIcons

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[ArrangeDesktopIcons](#) arrange all desktop icons.

## **Declare Syntax :**

Declare Sub cArrangeDesktopIcons Lib "[time2win.dll](#)" ()

## **Call Syntax :**

Call cArrangeDesktopIcons()

## **Where :**

## **Comments :**

## **Examples :**

**See also** : [Windows](#)

# GetBitX

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetBitB](#) return if a given bit in a given BYTE is Set or Reset.

[GetBitD](#) return if a given bit in a given DOUBLE is Set or Reset.

[GetBitI](#) return if a given bit in a given INTEGER is Set or Reset.

[GetBitL](#) return if a given bit in a given LONG is Set or Reset.

[GetBitS](#) return if a given bit in a given SINGLE is Set or Reset.

## Declare Syntax :

Declare Function cGetBitB Lib "[time2win.dll](#)" (ByVal Value As Byte, ByVal Position As Integer) As Integer

Declare Function cGetBitD Lib "[time2win.dll](#)" (ByVal Value As Double, ByVal Position As Integer) As Integer

Declare Function cGetBitI Lib "[time2win.dll](#)" (ByVal Value As Integer, ByVal Position As Integer) As Integer

Declare Function cGetBitL Lib "[time2win.dll](#)" (ByVal Value As Long, ByVal Position As Integer) As Integer

Declare Function cGetBitS Lib "[time2win.dll](#)" (ByVal Value As Single, ByVal Position As Integer) As Integer

## Call Syntax :

test% = cGetBitB(Value, Position)

test% = cGetBitD(Value, Position)

test% = cGetBitI(Value, Position)

test% = cGetBitL(Value, Position)

test% = cGetBitS(Value, Position)

## Where :

Value	the value to proceed
Position	the bit position
test	TRUE if the bit is Set FALSE if the bit is Reset

## Comments :

The first bit in the value is the bit 0.

**See also :** [Binary](#)



# GetCurrentDrive

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[GetCurrentDrive](#) return the current default drive.

## **Declare Syntax :**

Declare Function cGetCurrentDrive Lib "[time2win.dll](#)" () As String

## **Call Syntax :**

```
test$ = cGetCurrentDrive()
```

## **Where :**

test\$            the drive in a letter

## **Comments :**

## **Examples :**

**See also** : [Windows](#)

# EnumOpenFiles

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[EnumOpenFiles](#) enumerate all open files and/or all unmovable open files.

## Declare Syntax :

Declare Function cEnumOpenFiles Lib "time2win.dll" (ByVal nDrive As String, ByVal EnumType As Integer, ByVal FirstNext As Integer, OpenFileName As String, OpenFileMode As Long, OpenFileType As Long) As Integer

## Call Syntax :

intResult% = cEnumOpenFiles(nDrive\$, EnumType%, FirstNext%, OpenFileName\$, OpenFileMode%, OpenFileType%)

## Where :

nDrive\$	is the drive letter on which you want to search open files. (use "" for the current drive).
EnumType%	ENUMERATE_ALL_OPEN_FILES ENUMERATE_ONLY_OPEN_UNMOVABLE_FILES
FirstNext%	True : to search the first open file False : to search the next open file
OpenFileName\$	is the returned open file name.
OpenFileMode&	is the returned open file mode.
OpenFileType&	is the returned open file type.
intResult%	NO_ERROR_OPEN_FILES NO_MORE_OPEN_FILES ERROR_LOCK_LOGICAL_VOLUME ERROR_ENUMERATE_OPEN_FILES ERROR_UNLOCK_LOGICAL_VOLUME

Kind of file to enumerate. This parameter can be ENUMERATE\_ALL\_OPEN\_FILES to enumerate all open files or ENUMERATE\_ONLY\_OPEN\_UNMOVABLE\_FILES to enumerate only open unmovable files, including open memory-mapped files and other open unmovable files (32-bit Windows-based DLLs and executables). : to enumerate all open files

## Comments :

### About Returned Value :

NO\_ERROR\_OPEN\_FILES : no error, you can continue the enumeration of open files.  
NO\_MORE\_OPEN\_FILES : no more open files, the enumeration is finish.  
ERROR\_LOCK\_LOGICAL\_VOLUME : can't lock the logical volume.  
ERROR\_ENUMERATE\_OPEN\_FILES : error when enumerating open files.  
ERROR\_UNLOCK\_LOGICAL\_VOLUME : can't unlock the logical volume.

### About EnumType :

Kind of file to enumerate. This parameter can be ENUMERATE\_ALL\_OPEN\_FILES to enumerate all open files or ENUMERATE\_ONLY\_OPEN\_UNMOVABLE\_FILES to enumerate only open unmovable files, including open memory-mapped files and other open unmovable files (32-bit Windows-based DLLs and executables).

### About OpenFileMode :

Mode that the file was opened in, which is a combination of access mode, sharing mode, and open flags. It can be one value each from the access and sharing modes and any combination of open flags.

#### Access modes

```
OPEN_ACCESS_READONLY (0000h)
OPEN_ACCESS_WRITEONLY (0001h)
OPEN_ACCESS_READWRITE (0002h)
OPEN_ACCESS_RO_NOMODLASTACCESS (0004h)
```

#### Share modes

```
OPEN_SHARE_COMPATIBLE (0000h)
OPEN_SHARE_DENYREADWRITE (0010h)
OPEN_SHARE_DENYWRITE (0020h)
OPEN_SHARE_DENYREAD (0030h)
OPEN_SHARE_DENYNONE (0040h)
```

#### Open flags

```
OPEN_FLAGS_NOINHERIT (0080h)
OPEN_FLAGS_NO_BUFFERING (0100h)
OPEN_FLAGS_NO_COMPRESS (0200h)
OPEN_FLAGS_ALIAS_HINT (0400h)
OPEN_FLAGS_NOCRITERR (2000h)
OPEN_FLAGS_COMMIT (4000h)
```

#### About OpenFileType :

- 0 For normal files
- 1 For a memory-mapped files (memory-mapped files are unmovable)
- 2 For any other unmovable files (32-bit Windows-based DLLs and executables)
- 4 For the swap file

Note that if a memory-mapped file is returned (OpenFileType = 1), the value returned in OpenFileMode is limited to the following values:

```
OPEN_ACCESS_READONLY (0000h)
OPEN_ACCESS_READWRITE (0002h)
```

#### Examples :

```
Dim intResult As Integer
Dim OpenFileName As String
Dim OpenFileMode As Long
Dim OpenFileType As Long
```

```
intResult = cEnumOpenFiles("C", ENUMERATE_ALL_OPEN_FILES, True, OpenFileName, OpenFileMode,
OpenFileType)
```

```
While (intResult = NO_ERROR_OPEN_FILES)
    Debug.Print intResult, OpenFileName, OpenFileMode, OpenFileType
    intResult = cEnumOpenFiles("C", ENUMERATE_ALL_OPEN_FILES, False, OpenFileName, OpenFileMode,
OpenFileType)
Wend
```

```
intResult = cEnumOpenFiles("D", ENUMERATE_ALL_OPEN_FILES, True, OpenFileName, OpenFileMode,
OpenFileType)
```

```
While (intResult = NO_ERROR_OPEN_FILES)
```

```
    Debug.Print intResult, OpenFileName, OpenFileMode, OpenFileType
    intResult = cEnumOpenFiles("D", ENUMERATE_ALL_OPEN_FILES, False, OpenFileName, OpenFileMode,
OpenFileType)
Wend
```

```
intResult = cEnumOpenFiles("E", ENUMERATE_ALL_OPEN_FILES, True, OpenFileName, OpenFileMode,
OpenFileType)
```

```
While (intResult = NO_ERROR_OPEN_FILES)
    Debug.Print intResult, OpenFileName, OpenFileMode, OpenFileType
    intResult = cEnumOpenFiles("E", ENUMERATE_ALL_OPEN_FILES, False, OpenFileName, OpenFileMode,
OpenFileType)
Wend
```

**See also :** [File](#)

# DESEncrypt, DESDecrypt

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DESEncrypt](#) encode a string with a password using the U.S. Data Encryption Standard cipher.

[DESDecrypt](#) decode a string with a password using the U.S. Data Encryption Standard cipher.

## Declare Syntax :

Declare Function cDESEncrypt Lib "time2win.dll" (Text As String, Key As String) As String

Declare Function cDESDecrypt Lib "time2win.dll" (Text As String, Key As String) As String

## Call Syntax :

testE = cDESEncrypt(Text, Key)

testD = cDESDecrypt(Text, Key)

## Where :

Text	is the string to encrypt/decrypt
Key	is the key to use for encryption/decryption
test	is the string encrypted/decrypted

## Comments :

The Key is case sensitive.

The length of Text can be any size.

The length of Key must be greater or equal to 8 characters.

The encrypted string is always a multiple of 8 characters + 1 character.

## Examples :

```
Dim TextAs String
Dim Key          As String
Dim Enc As String
Dim Dec As String
```

```
Text = "Under the blue sky, the sun is yellow"
```

```
Key = "a new encryption"
```

```
Enc = cDESEncrypt(Text, Key)
```

```
Dec = cDESDecrypt(Enc, Key)
```

**See also :** [Encryption](#)

# IDEAencrypt, IDEAdecrypt

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[IDEAencrypt](#) encode a string with a password using the International Data Encryption Algorithm cipher.

[IDEAdecrypt](#) decode a string with a password using the International Data Encryption Algorithm cipher.

## Declare Syntax :

```
Declare Function cIDEAencrypt Lib "time2win.dll" (Text As String, Key As String) As String
```

```
Declare Function cIDEAdecrypt Lib "time2win.dll" (Text As String, Key As String) As String
```

## Call Syntax :

```
testE = cIDEAencrypt(Text, Key)
```

```
testD = cIDEAdecrypt(Text, Key)
```

## Where :

Text	is the string to encrypt/decrypt
Key	is the key to use for encryption/decryption
test	is the string encrypted/decrypted

## Comments :

The Key is case sensitive.

The length of Text can be any size.

The length of Key must be greater or equal to 16 characters.

The encrypted string is always a multiple of 8 characters + 1 character.

## Examples :

```
Dim TextAs String
Dim Key          As String
Dim Enc As String
Dim Dec As String
```

```
Text = "Under the blue sky, the sun is yellow"
```

```
Key = "a new encryption"
```

```
Enc = cIDEAencrypt(Text, Key)
```

```
Dec = cIDEAdecrypt(Enc, Key)
```

**See also :** [Encryption](#)

# LZARlcompress, LZARlexpand

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[LZARlcompress](#) compress a file into a compressed format using arithmetic compression.

[LZARlexpand](#) expand a compressed file into a normal format using arithmetic compression.

## Declare Syntax :

Declare Function cLZARlcompress Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String) As Long  
Declare Function cLZARlexpand Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String) As Long

## Call Syntax :

Test& = cLZARlcompress(FileIn\$, FileOut\$)

Test& = cLZARlexpand(FileIn\$, FileOut\$)

## Where :

FileIn\$	is the original/compressed file.
FileOut\$	is the compressed/original file.
Test&	<0, an error has occurred. >=0, the length of the created file.

## Comments :

The following constants are used to explain the error code :

```
Public Const CMPEXP_FILEIN_CANT_BE_NULL = -1
' occurs when the FileIn is an empty string
Public Const CMPEXP_FILEOUT_CANT_BE_NULL = -2
' occurs when the FileOut is an empty string
Public Const CMPEXP_FILEIN_AND_FILEOUT_CANT_BE_THE_SAME = -3
' occurs when the FileIn and FileOut are the same
Public Const CMPEXP_FILEIN_CANT_BE_OPENED = -4
' occurs when the FileIn can't be opened (not valid, not exist or disk error)
Public Const CMPEXP_FILEOUT_CANT_BE_CREATED = -5
' occurs when the FileOut can't be created (not valid or disk error)
Public Const CMPEXP_COMPRESS_OR_EXPAND_ERROR = -6
' occurs when compression or expansion can't be performed (disk error)
Public Const CMPEXP_CANT_GET_FILEOUT_SIZE = -7
' occurs when the length of FileOut can be read (disk error)
```

## Examples :

```
Dim FileIn      As String
Dim FileOut     As String
Dim FileOut2    As String
Dim LengthIn   As Long
Dim LengthOut  As Long
```

```
FileIn = "c:\win95\system\msjt3032.dll"
FileOut = "c:\tmp\test.ari"
FileOut = "c:\tmp\test.unari"
```

```
LengthOut = cLZARlcompress(FileIn, FileOut)
LengthIn = cLZARlexpand(FileOut, FileOut2)
```

**See also :** [Compression](#)



# DIAMONDencrypt, DIAMONDdecrypt

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[DIAMONDencrypt](#) encode a string with a password using the Diamond Encryption Algorithm (4 modes).

[DIAMONDdecrypt](#) decode a string with a password using the Diamond Encryption Algorithm (4 modes).

## Declare Syntax :

Declare Function cDIAMONDencrypt Lib "time2win.dll" (Text As String, Key As String, ByVal Mode As Integer) As String

Declare Function cDIAMONDdecrypt Lib "time2win.dll" (Text As String, Key As String, ByVal Mode As Integer) As String

## Call Syntax :

testE = cDIAMONDencrypt(Text, Key)

testD = cDIAMONDdecrypt(Text, Key)

## Where :

Text	is the string to encrypt/decrypt
Key	is the key to use for encryption/decryption
Mode	Public Const DIAMOND_FULL_MODE1 = 1 ' more strongest (slowest) Public Const DIAMOND_FULL_MODE2 = 3 ' more strong (slow) Public Const DIAMOND_LITE_MODE1 = 2 ' strongest (fast) Public Const DIAMOND_LITE_MODE2 = 4 ' strong (fastest)
test	is the string encrypted/decrypted

## Comments :

The Key is case sensitive.

The length of Text can be any size.

The length of Key must be greater or equal to 5 characters (best result with a key of 16 characters or more).

In FULL MODE, the length of the encrypted string is always a multiple of 16 characters + 1 character.

In LITE MODE, the length of the encrypted string is always a multiple of 8 characters + 1 character.

## Examples :

```
Dim TextAs String
Dim Key          As String
Dim Enc As String
Dim Dec As String
```

```
Text = "Under the blue sky, the sun is yellow"
```

```
Key = "a new encryption"
```

```
Enc = cDIAMONDencrypt(Text, Key, DIAMOND_FULL_MODE1)
```

```
Dec = cDIAMONDdecrypt(Enc, Key, DIAMOND_LITE_MODE1)
```

**See also :** [Encryption](#)

# GZIPFileCompress, GZIPFileExpand

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[GZIPFileCompress](#) compress a file into a compressed format using GZIP compression method.

[GZIPFileExpand](#) expand a compressed file into a normal format using GZIP compression method.

## Declare Syntax :

```
Declare Function cGZIPFileCompress Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String) As Long
Declare Function cGZIPFileExpand Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String) As Long
```

## Call Syntax :

```
Test& = cGZIPFileCompress(FileIn$, FileOut$)
```

```
Test& = cGZIPFileExpand(FileIn$, FileOut$)
```

## Where :

FileIn\$	is the original/compressed file.
FileOut\$	is the compressed/original file.
Test&	<0, an error has occurred. >=0, the length of the created file.

## Comments :

The following constants are used to explain the error code :

```
Public Const CMPEXP_FILEIN_CANT_BE_NULL = -1
' occurs when the FileIn is an empty string
Public Const CMPEXP_FILEOUT_CANT_BE_NULL = -2
' occurs when the FileOut is an empty string
Public Const CMPEXP_FILEIN_AND_FILEOUT_CANT_BE_THE_SAME = -3
' occurs when the FileIn and FileOut are the same
Public Const CMPEXP_FILEIN_CANT_BE_OPENED = -4
' occurs when the FileIn can't be opened (not valid, not exist or disk error)
Public Const CMPEXP_FILEOUT_CANT_BE_CREATED = -5
' occurs when the FileOut can't be created (not valid or disk error)
Public Const CMPEXP_COMPRESS_OR_EXPAND_ERROR = -6
' occurs when compression or expansion can't be performed (disk error)
Public Const CMPEXP_CANT_GET_FILEOUT_SIZE = -7
' occurs when the length of FileOut can be read (disk error)
```

## Examples :

```
Dim FileIn      As String
Dim FileOut     As String
Dim FileOut2    As String
Dim LengthIn   As Long
Dim LengthOut   As Long
```

```
FileIn = "c:\win95\system\msjt3032.dll"
FileOut = "c:\tmp\test.gzi"
FileOut = "c:\tmp\test.ugz"
```

```
LengthOut = cGZIPFileCompress(FileIn, FileOut)
LengthIn = cGZIPFileExpand(FileOut, FileOut2)
```

**See also :** [Compression](#)

# DisplaySplash, DestroySplash

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[DisplaySplash](#) display a bitmap as splash screen.

[DestroySplash](#) destroy a bitmap displayed by DisplaySplash.

## Declare Syntax :

```
Declare Function cDisplaySplash Lib "time2win.dll" (ByVal lpFilename As String, ByVal MinimumTimeToStay As Integer, ByVal AutoDestruction As Integer) As Long
```

```
Declare Function cDestroySplash Lib "time2win.dll" (ByVal hWnd As Long) As Integer
```

## Call Syntax :

```
hWnd& = cDisplaySplash(lpFileName$, MinimumTimeToStay%)  
intResult% = cDestroySplash(hWnd&)
```

## Where :

lpFileName\$	is the name of the file to read the DDB (Device-Dependent Bitmap) or DIB (Device-Independent Bitmap).
MinimumTimeToStay%	is the minimum time (in milliseconds) that the bitmap is displayed on the screen.
AutoDestruction%	TRUE : the splash bitmap is automatically destroyed after that the MinimumTimeToStay is reached. FALSE : the splash bitmap is NOT automatically destroyed after that the MinimumTimeToStay is reached.
hWnd&	is the return hWnd value of the windows containing the bitmap (used by DestroySplash).
intResult%	TRUE : all is OK (destroy is successful). FALSE : an error has occurred when destroying the windows containing the bitmap.

## Comments :

If MinimumTimeToStay is set to 0, the splash screen is always displayed until you call cDestroySplash function.  
If AutoDestruction is set to FALSE, you MUST CALL the cDestroySplash function to destroy the splash screen.

## Examples :

With AutoDestruction set to FALSE :

```
Dim hWnd As Long  
Dim intResult As Integer  
  
hWnd = cDisplaySplash(App.Path + "\time2win.dib", 7000, False)  
  
' some codes  
' some initializations  
  
intResult = cDestroySplash(hWnd)
```

With AutoDestruction set to TRUE :

```
Dim hWnd As Long  
  
hWnd = cDisplaySplash(App.Path + "\time2win.dib", 7000, True)  
  
' some codes
```

' some initializations

**See also :** [Bitmap](#)

# GetAllSettings

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetAllSettings](#) returns a list of key settings and their respective values from an application's entry in the registry.

## Declare Syntax :

```
Declare Function cGetAllSettings Lib "time2win.dll" (Strarray() As String, ByVal IpSection As String) As Integer
```

## Call Syntax :

```
test% = cGetAllSettings(Strarray(), IpSection$)
```

## Where :

Strarray()	is the array to receive the list (must be a two-dimensionnal array in the form Strarray(1 to X, 1 to 2)).
IpSection\$	string expression containing the name of the section where the key setting is being saved.
test%	>0 : the number of elements <0 : see <a href="#">error code</a>

## Comments :

If the array is too bigger, the non-necessary element are set to empty.

## Examples :

```
Dim Strarray(1 To 7, 1 To 2) As String
```

```
Debug.Print cGetAllSettings(Strarray(), "Software\VB and VBA Program Settings\MyApp\Startup")  
Debug.Print Strarray(1, 1), Strarray(1, 2)  
Debug.Print Strarray(2, 1), Strarray(2, 2)  
Debug.Print Strarray(3, 1), Strarray(3, 2)  
Debug.Print Strarray(4, 1), Strarray(4, 2)  
Debug.Print Strarray(5, 1), Strarray(5, 2)
```

**See also :** [Registry key](#)

# ASHFileCompress, ASHFileExpand

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ASHFileCompress](#) compress a file into a compressed format using ASH arithmetic compression.

[ASHFileExpand](#) expand a compressed file into a normal format using ASH arithmetic compression.

## Declare Syntax :

```
Declare Function cASHFileCompress Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, ByVal Order As Integer) As Long
```

```
Declare Function cASHFileExpand Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, ByVal Order As Integer) As Long
```

## Call Syntax :

```
Test& = cASHFileCompress(FileIn$, FileOut$)
```

```
Test& = cASHFileExpand(FileIn$, FileOut$)
```

## Where :

FileIn\$	is the original/compressed file.
FileOut\$	is the compressed/original file.
Order%	is the compression order form 0 to 9.
Test&	<0, an error has occurred. >=0, the length of the created file.

## Comments :

The ASH compression method is very performant but is very slow. Use with care on large files.

The following constants are used to explain the error code :

```
Public Const CMPEXP_FILEIN_CANT_BE_NULL = -1
' occurs when the FileIn is an empty string
Public Const CMPEXP_FILEOUT_CANT_BE_NULL = -2
' occurs when the FileOut is an empty string
Public Const CMPEXP_FILEIN_AND_FILEOUT_CANT_BE_THE_SAME = -3
' occurs when the FileIn and FileOut are the same
Public Const CMPEXP_FILEIN_CANT_BE_OPENED = -4
' occurs when the FileIn can't be opened (not valid, not exist or disk error)
Public Const CMPEXP_FILEOUT_CANT_BE_CREATED = -5
' occurs when the FileOut can't be created (not valid or disk error)
Public Const CMPEXP_COMPRESS_OR_EXPAND_ERROR = -6
' occurs when compression or expansion can't be performed (disk error)
Public Const CMPEXP_CANT_GET_FILEOUT_SIZE = -7
' occurs when the length of FileOut can be read (disk error)
```

## Examples :

```
Dim FileIn As String
Dim FileOut As String
Dim FileOut2 As String
Dim LengthIn As Long
Dim LengthOut As Long
```

```
FileIn = "c:\win95\system\msjt3032.dll"
```

```
FileOut = "c:\tmp\test.ash"
```

FileOut = "c:\tmp\test.uah"

LengthOut = cASHFileCompress(FileIn, FileOut)  
LengthIn = cASHFileExpand(FileOut, FileOut2)

**See also :** [Compression](#)



# IDEAencryptFile, IDEAdecryptFile

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[IDEAencryptFile](#) copy one file to an another file but with IDEA encryption.

[IDEAdecryptFile](#) copy one file to an another file but with IDEA decryption.

## Declare Syntax :

```
Declare Function cIDEAencryptFile Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, Key As String) As Long
```

```
Declare Function cIDEAdecryptFile Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, Key As String) As Long
```

## Call Syntax :

```
test& = cIDEAencryptFile(FileIn, FileOut, Key)
```

```
test& = cIDEAdecryptFile(FileIn, FileOut, Key)
```

## Where :

FileIn\$	is the source file.
FileOut\$	is the destination file.
Key	is the key to use for encryption/decryption.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

The Key is case sensitive.

The length of Text can be any size.

The length of Key must be greater or equal to 16 characters.

The encrypted file is always a multiple of 8 characters + 1 character.

If the returned code is a negative value, it take the following value :

```
Public Const CRYPTO_KEY_TOO_SMALL = -1
Public Const CRYPTO_CANT_INIT_KEY = -2
Public Const CRYPTO_CANT_INIT_BUFFER = -11
Public Const CRYPTO_CANT_OPEN_FILEIN = -21
Public Const CRYPTO_CANT_CREATE_FILEOUT = -22
Public Const CRYPTO_ERROR_READING_FILEIN = -31
Public Const CRYPTO_ERROR1_WRITING_FILEOUT = -41
Public Const CRYPTO_ERROR2_WRITING_FILEOUT = -42
Public Const CRYPTO_ERROR1_WRITING_LASTBYTE = -51
Public Const CRYPTO_ERROR2_WRITING_LASTBYTE = -52
Public Const CRYPTO_BAD_LASTBYTE = -61
```

## Examples :

```
Dim TestAs Long
```

```
Test = cIDEAencryptFile("c:\autoexec.bat", "c:\autoexec.tb1", "Time To Win")
```

```
Test = cIDEAdecryptFile("c:\autoexec.tb1", "c:\autoexec.tb2", "Time To Win")
```

**See also :** [Encryption](#)

# DIAMONDencryptFile, DIAMONDdecryptFile

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**DIAMONDencryptFile** copy one file to an another file but with Diamond Encryption Algorithm (4 modes).

**DIAMONDdecryptFile** copy one file to an another file but with Diamond Encryption Algorithm (4 modes).

## Description :

Diamond is a cipher designed to exceed DES in strength. Diamond uses a variable length key of at least 40 bits. The use of at least a 16 bytes key is recommended for long term protection of very sensitive data, as a hedge against the possibility of computing power increasing by several orders of magnitudes in the coming years.

It is conjectured that Diamond Lite (Mode 2) and a key length of 16 bytes is at least equivalent in security to the IDEA cipher, and more secure than the ageing DES algorithm.

## Declare Syntax :

Declare Function cDIAMONDencryptFile Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, Key As String, ByVal Mode As Integer) As Long

Declare Function cDIAMONDdecryptFile Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, Key As String, ByVal Mode As Integer) As Long

## Call Syntax :

test& = cDIAMONDencryptFile(FileIn, FileOut, Key, Mode)

test& = cDIAMONDdecryptFile(FileIn, FileOut, Key, Mode)

## Where :

FileIn\$ is the source file.

FileOut\$ is the destination file.

Key is the key to use for encryption/decryption.

Mode Public Const DIAMOND\_FULL\_MODE1 = 1 ' more strongest (slowest)

Public Const DIAMOND\_FULL\_MODE2 = 3 ' more strong (slow)

Public Const DIAMOND\_LITE\_MODE1 = 2 ' strongest (fast)

Public Const DIAMOND\_LITE\_MODE2 = 4 ' strong (fastest)

test& > 0 if all is OK (the returned value is the total bytes copied),  
< 0 if an error has occurred.

## Comments :

The Key is case sensitive.

The length of Text can be any size.

The length of Key must be greater or equal to 5 characters (best result with a key of 16 characters or more).

In FULL MODE, the length of the encrypted file is always a multiple of 16 characters + 1 character.

In LITE MODE, the length of the encrypted file is always a multiple of 8 characters + 1 character.

If the returned code is a negative value, it take the following value :

Public Const CRYPTO\_KEY\_TOO\_SMALL = -1

Public Const CRYPTO\_CANT\_INIT\_KEY = -2

Public Const CRYPTO\_CANT\_INIT\_BUFFER = -11

Public Const CRYPTO\_CANT\_OPEN\_FILEIN = -21

Public Const CRYPTO\_CANT\_CREATE\_FILEOUT = -22

Public Const CRYPTO\_ERROR\_READING\_FILEIN = -31

Public Const CRYPTO\_ERROR1\_WRITING\_FILEOUT = -41

Public Const CRYPTO\_ERROR2\_WRITING\_FILEOUT = -42

```
Public Const CRYPTO_ERROR1_WRITING_LASTBYTE = -51
Public Const CRYPTO_ERROR2_WRITING_LASTBYTE = -52
Public Const CRYPTO_BAD_LASTBYTE = -61
```

**Examples :**

```
Dim TestAs Long
```

```
Test = cDIAMONDencryptFile("c:\autoexec.bat", "c:\autoexec.tb1", "Time To Win", DIAMOND_FULL_MODE1)
Test = cDIAMONDdecryptFile("c:\autoexec.tb1", "c:\autoexec.tb2", "Time To Win", DIAMOND_LITE_MODE1)
```

**See also :** [Encryption](#)

# RUBYencryptFile, RUBYdecryptFile

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[RUBYencryptFile](#) copy one file to an another file but with RUBY algorithm (7 modes).

[RUBYdecryptFile](#) copy one file to an another file but with RUBY algorithm (7 modes).

## Declare Syntax :

Declare Function cRUBYencryptFile Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, Key As String, ByVal Mode As Integer) As Long

Declare Function cRUBYdecryptFile Lib "time2win.dll" (ByVal FileIn As String, ByVal FileOut As String, Key As String, ByVal Mode As Integer) As Long

## Call Syntax :

test& = cRUBYencryptFile(FileIn, FileOut, Key, Mode)

test& = cRUBYdecryptFile(FileIn, FileOut, Key, Mode)

## Where :

FileIn\$ is the source file.

FileOut\$ is the destination file.

Key is the key to use for encryption/decryption.

Mode Public Const RUBY\_MODE\_MINIMUM = 1 ' speed is of the essence, security

secondary. Public Const RUBY\_MODE\_DESK\_LOCK = 2 ' reasonable compromise of speed vs security.

things. Public Const RUBY\_MODE\_DEAD\_BOLT = 4 ' default = probably good enough for most things.

Public Const RUBY\_MODE\_PORTABLE\_SAFE = 5 ' security is more important than speed.

Public Const RUBY\_MODE\_ANCHORED\_SAFE = 8 ' speed isn't much of a concern.

anyway. Public Const RUBY\_MODE\_BANK\_VAULT = 10 ' your pentium has nothing better to do,

Public Const RUBY\_MODE\_FORT\_KNOX = 16 ' be cool.

test& > 0 if all is OK (the returned value is the total bytes copied),  
< 0 if an error has occurred.

## Comments :

The Key is case sensitive.

The length of Text can be any size.

The length of Key must be greater or equal to 6 characters.

If the returned code is a negative value, it take the following value :

```
Public Const CRYPTO_KEY_TOO_SMALL = -1
Public Const CRYPTO_CANT_INIT_KEY = -2
Public Const CRYPTO_CANT_INIT_BUFFER = -11
Public Const CRYPTO_CANT_OPEN_FILEIN = -21
Public Const CRYPTO_CANT_CREATE_FILEOUT = -22
Public Const CRYPTO_ERROR_READING_FILEIN = -31
Public Const CRYPTO_ERROR1_WRITING_FILEOUT = -41
Public Const CRYPTO_ERROR2_WRITING_FILEOUT = -42
Public Const CRYPTO_ERROR1_WRITING_LASTBYTE = -51
Public Const CRYPTO_ERROR2_WRITING_LASTBYTE = -52
Public Const CRYPTO_BAD_LASTBYTE = -61
```

## Examples :

Dim TestAs Long

```
Test = cRUBYencryptFile("c:\autoexec.bat", "c:\autoexec.tb1", "Time To Win", RUBY_MODE_DESK_LOCK)  
Test = cRUBYdecryptFile("c:\autoexec.tb1", "c:\autoexec.tb2", "Time To Win", RUBY_MODE_DESK_LOCK)
```

**See also :** [Encryption](#)

# DeleteSubString

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DeleteSubString](#) deletes all occurrences of one string, in another string.

## Declare Syntax :

Declare Function cDeleteSubString Lib "[time2win.dll](#)" (ByVal Txt As String, ByVal SubString As String, ByVal Sensitivity As Integer) As String

## Call Syntax :

```
test$ = cDeleteSubString(Txt$, SubString$, Sensitivity%)
```

## Where :

Txt\$	the string to proceed.
SubString\$	the sub-string to be deleted
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
test\$	the result.

## Comments :

## Examples :

test\$ = cDeleteSubString("this is a test", "is", True)	-> "th a test"
test\$ = cDeleteSubString("this is a test", "is", False)	-> "th a test"
test\$ = cDeleteSubString("this is a test", "IS", True)	-> "this is a test"
test\$ = cDeleteSubString("this is a test", "IS", False)	-> "th a test"
test\$ = cDeleteSubString("this is a test", "T", True)	-> "this is a test"
test\$ = cDeleteSubString("this is a test", "T", False)	-> "his is a es"

**See also :** [String](#)

# SetBitX

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SetBitB](#) set a given bit in a given BYTE to Set state or Reset state.  
[SetBitD](#) set a given bit in a given DOUBLE to Set state or Reset state.  
[SetBitI](#) set a given bit in a given INTEGER to Set state or Reset state.  
[SetBitL](#) set a given bit in a given LONG to Set state or Reset state.  
[SetBitS](#) set a given bit in a given SINGLE to Set state or Reset state.

## Declare Syntax :

```
Declare Sub cSetBitB Lib "time2win.dll" (Value As Byte, ByVal Position As Integer, ByVal BitValue As Integer)
Declare Sub cSetBitD Lib "time2win.dll" (Value As Double, ByVal Position As Integer, ByVal BitValue As Integer)
Declare Sub cSetBitI Lib "time2win.dll" (Value As Integer, ByVal Position As Integer, ByVal BitValue As Integer)
Declare Sub cSetBitL Lib "time2win.dll" (Value As Long, ByVal Position As Integer, ByVal BitValue As Integer)
Declare Sub cSetBitS Lib "time2win.dll" (Value As Single, ByVal Position As Integer, ByVal BitValue As Integer)
```

## Call Syntax :

```
Call cSetBitB(Value, Position, BitValue)
Call cSetBitD(Value, Position, BitValue)
Call cSetBitI(Value, Position, BitValue)
Call cSetBitL(Value, Position, BitValue)
Call cSetBitS(Value, Position, BitValue)
```

## Where :

Value	the value to proceed
Position	the bit position
BitValue	TRUE to Set the bit FALSE to Reset the bit

## Comments :

The first bit in the string is the bit 0.

**See also :** [Binary](#)

# FromZ9, ToZ9

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FromZ9](#) convert a Z9 string to an ascii string.

[ToZ9](#) convert a ascii string to Z9 string.

## Declare Syntax :

```
Declare Function cFromZ9 Lib "time2win.dll" (Text As String) As String
```

```
Declare Function cToZ9 Lib "time2win.dll" (Text As String) As String
```

## Call Syntax :

```
test$ = cFromZ9(Text)
```

```
test$ = cToZ9(Text)
```

## Where :

Text                    the string to proceed

test\$                   the result

## Comments :

The returned string from ToZ9 is always a multiple of 2.

If the size of the string passed to FromZ9 is not a multiple of 2, only n-1 chars are used.

## Examples :

```
test$ = cToZ9("ABCDEFGH")
```

```
' "1T1U1V1W1X1Y1Z"
```

```
test$ = cFromZ9("1T1U1V1W1X1Y1Z")
```

```
' "ABCDEFGH"
```

**See also** : [Binary](#)



# ModX

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Mod10](#) calculate the mod-10 of the given string.  
[Mod10R](#) calculate the reverse mod-10 of the given string.  
[Mod11](#) calculate the mod-11 of the given string.  
[Mod11R](#) calculate the reverse mod-11 of the given string.  
[Mod137](#) calculate the mod-1.3.7 of the given string.  
[Mod137R](#) calculate the reverse mod-1.3.7 of the given string.

## Declare Syntax :

```
Declare Function cMod10 Lib "time2win.dll" (ByVal Text As String) As Integer
Declare Function cMod10R Lib "time2win.dll" (ByVal Text As String) As Integer
Declare Function cMod11 Lib "time2win.dll" (ByVal Text As String) As Integer
Declare Function cMod11R Lib "time2win.dll" (ByVal Text As String) As Integer
Declare Function cMod137 Lib "time2win.dll" (ByVal Text As String) As Integer
Declare Function cMod137R Lib "time2win.dll" (ByVal Text As String) As Integer
```

## Call Syntax :

```
Modulo = cMod10(Text)
Modulo = cMod10R(Text)
Modulo = cMod11(Text)
Modulo = cMod11R(Text)
Modulo = cMod137(Text)
Modulo = cMod137R(Text)
```

## Where :

Text	the string to proceed
Modulo	the result

## Comments :

For Mod10, Mod10R, Mod11 and Mod11R :

The string to proceed must be an hexa string (A-Z, 0-9)

For Mod137 and Mod137R :

The string to proceed must be an ascii string (chr\$(0) - chr\$(255))

## Examples :

```
Modulo = cMod10("19960825")           ' -> 8
Modulo = cMod10R("19960825")         ' -> 2
Modulo = cMod11("19960825")          ' -> 3
Modulo = cMod11R("19960825")         ' -> 9
Modulo = cMod137("19960825")         ' -> 2
Modulo = cMod137R("19960825")        ' -> 2
Modulo = cMod137("TIME TO WIN")      ' -> 4
Modulo = cMod137R("TIME TO WIN")     ' -> 4
```

**See also :** [Protection](#)

# MinNotXD, MinNotXI, MinNotXL, MinNotXS

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**MinNotXD** will return the smallest value in a **Double** array, not equal to a specified value.

**MinNotXI** will return the smallest value in an **Integer** array, not equal to a specified value.

**MinNotXL** will return the smallest value in a **Long** array, not equal to a specified value.

**MinNotXS** will return the smallest value in a **Single** array, not equal to a specified value.

## Declare Syntax :

Declare Function cMinNotXD Lib "[time2win.dll](#)" (array() As Double, ByVal ValueNotToReturn As Double) As Double

Declare Function cMinNotXI Lib "[time2win.dll](#)" (array() As Integer, ByVal ValueNotToReturn As Double) As Integer

Declare Function cMinNotXL Lib "[time2win.dll](#)" (array() As Long, ByVal ValueNotToReturn As Double) As Long

Declare Function cMinNotXS Lib "[time2win.dll](#)" (array() As Single, ByVal ValueNotToReturn As Double) As Single

## Call Syntax :

smallest# = cMinNotXD(array(), ValueNotToReturn#)

smallest% = cMinNotXI(array(), ValueNotToReturn%)

smallest& = cMinNotXL(array(), ValueNotToReturn&)

smallest! = cMinNotXS(array(), ValueNotToReturn!)

## Where :

array() is the array (Double, Integer, Long, Single).

smallest is the smallest value (Double, Integer, Long, Single) from all of the elements of the array (Double, Integer, Long, Single).

ValueNotToReturn is the value which can not be returned.

## Comments :

**See Also :** [Array](#)

# SplitFile

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SplitFile](#) split a file into target files based on a part size.

## Declare Syntax :

Declare Function cSplitFile Lib "[time2win.dll](#)" (ByVal SourceFile As String, ByVal TargetFiles As String, ByVal PartSize As Long) As Long

## Call Syntax :

test& = cSplitFile(SourceFile\$, TargetFiles\$, PartSize&)

## Where :

SourceFiles\$	is the source file.
TargetFiles\$	is the target filename without extension.
PartSize&	is the size of each target files.
test&	> 0 if all is OK (the returned value is the number of target files created), < 0 if an error has occurred.

## Comments :

The extension of each target file created start at 000 and is incremented by 1.

The returned value can be negative and have the following value :

```
Public Const SPLIT_BAD_PARTSIZE = -1
Public Const SPLIT_BAD_SOURCE_FILENAME = -2
Public Const SPLIT_BAD_TARGET_FILENAME = -3
Public Const SPLIT_CANT_OPEN_SOURCE = -4
Public Const SPLIT_CANT_CREATE_TARGET = -5

-32720 the number of chars in a block for writing differs from the number of chars for reading.
-32730 reading error for SourceFile.
-32740 writing error for TargetFiles.
-32760 allocation error for memory buffer.
```

## Examples :

' Size of "c:\tmp\vb.hlp" is 3826138 bytes

```
Debug.Print cSplitFile("c:\tmp\vb.hlp", "d:\temp\vbttst1", 1457664) ' creation of 3 files
' file 1 is d:\temp\vbttst1.000 size is 1457664
' file 2 is d:\temp\vbttst1.001 size is 1457664
' file 3 is d:\temp\vbttst1.002 size is 910810

Debug.Print cSplitFile("c:\tmp\vb.hlp", "d:\temp\vbttst2", 2823457) ' creation of 2 files
' file 1 is d:\temp\vbttst2.000 size is 2823457
' file 2 is d:\temp\vbttst2.001 size is 1002681
```

**See also :** [File](#)

# MaxNotXD, MaxNotXI, MaxNotXL, MaxNotXS

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**MaxNotXD** will return the largest value in a **Double** array, not equal to a specified value.

**MaxNotXI** will return the largest value in an **Integer** array, not equal to a specified value.

**MaxNotXL** will return the largest value in a **Long** array, not equal to a specified value.

**MaxNotXS** will return the largest value in a **Single** array, not equal to a specified value.

## Declare Syntax :

Declare Function cMaxNotXD Lib "[time2win.dll](#)" (array() As Double, ByVal ValueNotToReturn As Double) As Double

Declare Function cMaxNotXI Lib "[time2win.dll](#)" (array() As Integer, ByVal ValueNotToReturn As Double) As Integer

Declare Function cMaxNotXL Lib "[time2win.dll](#)" (array() As Long, ByVal ValueNotToReturn As Double) As Long

Declare Function cMaxNotXS Lib "[time2win.dll](#)" (array() As Single, ByVal ValueNotToReturn As Double) As Single

## Call Syntax :

largest# = cMaxNotXD(array(), ValueNotToReturn#)

largest% = cMaxNotXI(array(), ValueNotToReturn%)

largest& = cMaxNotXL(array(), ValueNotToReturn&)

largest! = cMaxNotXS(array(), ValueNotToReturn!)

## Where :

array() is the array (Double, Integer, Long, Single).

largest (Double, Integer, Long, Single) is the largest value (Double, Integer, Long, Single) from all of the elements of the array.

ValueNotToReturn is the value which can not be returned.

## Comments :

**See Also :** [Array](#)

' definition for FilesCopy

```
Public Const FC_OVERWRITE_EXISTING = 1  
Public Const FC_INCLUDE_SUB_DIRECTORY = 2  
Public Const FC_DONT_COPY_EMPTY_DIRECTORY = 4  
Public Const FC_DISPLAY_2_LINES = 8
```

' definition for FilesMove

```
Public Const FM_REPLACE_EXISTING = 1  
Public Const FM_INCLUDE_SUB_DIRECTORY = 2  
Public Const FM_DONT_COPY_EMPTY_DIRECTORY = 4  
Public Const FM_DISPLAY_2_LINES = 8
```

# CutFile

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

CutFile cut a file in two parts (creation of two files).

## Declare Syntax :

Declare Function cCutFile Lib "time2win.dll" (ByVal SourceFile As String, ByVal TargetFile1 As String, ByVal TargetFile2 As String, ByVal CutPosition As Long) As Long

## Call Syntax :

test& = cCutFile(SourceFile\$, TargetFile1\$, TargetFile2\$, CutPosition&)

## Where :

SourceFiles\$	is the source file.
TargetFile1\$	is the target filename for the first part.
TargetFile2\$	is the target filename for the second part.
CutPosition&	is the position in the source file to cut this source file in two.
test&	> 0 if all is OK (the returned value is the sum of the two target files createn), < 0 if an error has occurred.

## Comments :

The returned value can be negative and have the following value :

```
Public Const CUT_BAD_POSITION = -1
Public Const CUT_BAD_SOURCE_FILENAME = -2
Public Const CUT_BAD_TARGET1_FILENAME = -3
Public Const CUT_BAD_TARGET2_FILENAME = -4
Public Const CUT_POSITION_TOO_BIG = -5
Public Const CUT_CANT_OPEN_SOURCE = -6
Public Const CUT_CANT_CREATE_TARGET1 = -7
Public Const CUT_CANT_CREATE_TARGET2 = -8
```

```
-32720 the number of chars in a block for writing differs from the number of chars for reading.
-32730 reading error for SourceFile.
-32740 writing error for TargetFiles.
-32760 allocation error for memory buffer.
```

## Examples :

```
Debug.Print cFileSize("e:\vb4\vb.hlp")
Debug.Print cCutFile("e:\vb4\vb.hlp", "d:\temp\vb10.hlp", "d:\temp\vb20.hlp", 1234567)
Debug.Print cCutFile("e:\vb4\vb.hlp", "d:\temp\vb11.hlp", "d:\temp\vb21.hlp", 0)
Debug.Print cCutFile("e:\vb4\vb.hlp", "d:\temp\vb12.hlp", "d:\temp\vb22.hlp", 1)
Debug.Print cCutFile("e:\vb4\vb.hlp", "d:\temp\vb13.hlp", "d:\temp\vb23.hlp", cFileSize("e:\vb4\vb.hlp"))
Debug.Print cFileSize("d:\temp\vb10.hlp"), cFileSize("d:\temp\vb20.hlp")
Debug.Print cFileSize("d:\temp\vb11.hlp"), cFileSize("d:\temp\vb21.hlp")
Debug.Print cFileSize("d:\temp\vb12.hlp"), cFileSize("d:\temp\vb22.hlp")
Debug.Print cFileSize("d:\temp\vb13.hlp"), cFileSize("d:\temp\vb23.hlp")
```

## Results :

```
3826138
3826138 => (no error : same size)
```

3826138  
3826138  
3826138  
1234567  
0  
1  
3826138

2591571  
3826138  
3826137  
0

=> (no error : same size)  
=> (no error : same size)  
=> (no error : same size)

**See also :** [File](#)

```
' definition for error type for SplitFile
Public Const SPLIT_BAD_PARTSIZE = -1
Public Const SPLIT_BAD_SOURCE_FILENAME = -2
Public Const SPLIT_BAD_TARGET_FILENAME = -3
Public Const SPLIT_CANT_OPEN_SOURCE = -4
Public Const SPLIT_CANT_CREATE_TARGET = -5
```



' definition for error type for CutFile

Public Const CUT\_BAD\_POSITION = -1

Public Const CUT\_BAD\_SOURCE\_FILENAME = -2

Public Const CUT\_BAD\_TARGET1\_FILENAME = -3

Public Const CUT\_BAD\_TARGET2\_FILENAME = -4

Public Const CUT\_POSITION\_TOO\_BIG = -5

Public Const CUT\_CANT\_OPEN\_SOURCE = -6

Public Const CUT\_CANT\_CREATE\_TARGET1 = -7

Public Const CUT\_CANT\_CREATE\_TARGET2 = -8

# References

Below, there is an alphabetical list of all companies which have registered one or more versions of TIME TO WIN.

## Thanks to all

10 licenses :

Mobil Corporation

2 licenses :

BusinessOnline GmbH

Impac Technologies

1 license :

3nP Systems

A.D.Min

A2i

AIRBUS INDUSTRIE (TRAINING CENTER)

Alcatel

Alpha Info Lab.

American Inventory Res.

ANALOG INFORMATICA s.r.l.

ArtCom GmbH

ArtNet GmbH

Ascom AUSTRIA G.m.b.H.

Atril Software

AT&T

ATTA NORM sa

Avantext, Inc

Beck Systemtechnik

BEKKER CONSULTING

BFI-Fochler GmbH

BodySculpture

Boeing

Butler Manufacturing Co.

C.H.U. de NICE

C/O Mlle Vaillant

CDLP REG.

CheqTech Systems Corp

Chevron Oil

Choice Business Systems

Cmc Systems

Coformex,s.a. de c.v.

COMMO-GODS Software

Compuware Corporation

Concepts & Solutions

Cowie Interleasing

CROWE'S COMPUTING SERVICE

CSF AG

CS Software GmbH

CT Innovations

CyberMatrix.

D & L Computing

Data Baur & Co.

Data General

DataService Steinforth

DDSystems

DemosDatamat

DevoSoft  
DMF Software  
DST Belvedere  
Duracell, Inc.  
E2 Ltd  
ECS  
EDV-BERATUNG BECKER  
EXACT  
FACILITEURS MICRO  
Farmacia-Laboratory Cohen  
Fenestra  
Gallup  
GamBit Software  
GEPA GmbH  
Gestion Sand-Ho Inc.  
HAGEE KONZEPT GmbH  
HCC  
hsd ComputerService GmbH  
Hydrogeological  
Hydrogeological Cons. Ltd  
ICAL  
ICI C&P  
Infosofia snc  
Ing. Büro MAP  
Int'l Consultants  
Integra G.m.b.H.  
Interactive  
IPS Ltd  
JAGENBERG Papiertechnik  
Johnson Controls, Inc  
Kodiak Computer, Inc.  
Leica bv.  
Leuchter Informatik AG  
M.T.M. Software  
Maestro Software  
Makoski und Preuss GbR  
MARC WINTER COMPUTER  
MEMO-DATA GmbH  
MEMSYS Inc.  
MEVATEC Corporation  
MICAD Systems  
MICRODA  
Middle Atlantic Prod.  
MOC Asset Management  
N&MO  
Naegele Computer  
Nett & So GmbH  
New Motion  
NOR Data i Bergslagen  
NOVELL  
Novo Nordisk IT  
Odense Skolefoto A/S  
Pausch Messtechnik GmbH  
PINCUS  
Pink Flamingo Software  
PlanB Software Concepts  
Precision Systems  
Procornea ned bv  
Professional Mgmt Systems  
ProGResS, Inc.  
Qualitech GmbH  
R C Associates, Inc.

Rainy River Forest Prdt  
RH Consulting  
Rolitron  
Sahara Soft  
Salsa Soft  
Salt and Light Software  
Santec Ltd  
Seadrift Coke, L.P.  
Sedasis Ingenierie  
SIGOLD  
Skymark packaging int Ltd  
Sofamor Danek  
Software 4 You  
Spartan Stores, INc.  
Star-Kist Foods  
Stone-Consolidated  
Sunlight Software  
Tantra publishing  
TecnoLab s.r.l.  
TMI Publishing A/S  
TRI-CEPTS  
UB Networks  
University of Queensland  
U.S. Dept. of Energy  
USG Corporation  
UVA Medical Center  
VIEW SL  
VLSI Technology, Inc.  
Wells Fargo  
WRF Software Inc

Thanks to all

# GetRegistryExt, KillRegistryExt, PutRegistryExt

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetRegistryExt](#) return a key setting value from an application's Windows registry entry for any key handle.

[KillRegistryExt](#) delete a section or key setting from the Windows registry entry for any key handle.

[PutRegistryExt](#) save or create an application entry in the Windows registry entry for any key handle.

## Declare Syntax :

```
Declare Function cGetRegistryExt Lib "time2win.dll" (ByVal KeyType As Integer, ByVal IpSection As String, ByVal IpKey As String, ByVal IpDefault As String) As String
```

```
Declare Function cPutRegistryExt Lib "time2win.dll" (ByVal KeyType As Integer, ByVal IpSection As String, ByVal IpKey As String, ByVal IpValue As String) As Integer
```

```
Declare Function cKillRegistryExt Lib "time2win.dll" (ByVal KeyType As Integer, ByVal IpSection As String, ByVal IpKey As String) As Integer
```

## Call Syntax :

```
retCode% = cPutRegistryExt(KeyType%, IpSection$, IpKey$, IpValue$)
```

```
retData$ = cGetRegistryExt(KeyType%, IpSection$, IpKey$, IpDefault$)
```

```
retCode% = cKillRegistryExt(KeyType%, IpSection$, IpKey$)
```

## Where :

KeyType%            see [configuration code](#)

IpSection\$          string expression containing the name of the section where the key setting is being saved.

IpKey\$              string expression containing the name of the key setting being saved.

IpValue\$ string expression containing the value that key is being set to.

IpDefault\$          a string that specifies the default value for the given entry if the entry cannot be found in the specified section.

retCode%            [error/success code](#).

## Comments :

## Examples :

```
Debug.Print cGetRegistryExt(RK_HKEY_LOCAL_MACHINE, "HARDWARE\DESCRIPTION\System\CentralProcessor0", "Identifier", "?")
```

```
Debug.Print cGetRegistryExt(RK_HKEY_LOCAL_MACHINE, "HARDWARE\DESCRIPTION\System\CentralProcessor0", "VendorIdentifier", "?")
```

**See also :** [Registry key](#)

# GetFileDateTime, SetFileDateTime

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[GetFileDateTime](#) retrieve in one routine all date & time informations (creation, last access, last write) for a file.  
[SetFileDateTime](#) set in one routine all date & time informations (creation, last access, last write) for a file.

## Declare Syntax :

```
Declare Function cGetFileDateTime Lib "time2win.dll" (ByVal nFilename As String, SysCreation As Any, SysLastAccess As Any, SysLastWrite As Any) As Integer
Declare Function cSetFileDateTime Lib "time2win.dll" (ByVal nFilename As String, SysCreation As Any, SysLastAccess As Any, SysLastWrite As Any) As Integer
```

## Call Syntax :

```
test = cGetFileDateTime(nFilename, SysCreation, SysLastAccess, SysLastWrite)
test = cSetFileDateTime(nFilename, SysCreation, SysLastAccess, SysLastWrite)
```

## Where :

nFilename\$	the file to retrieve/set the date & time informations.
SysCreation	the type'd variable <a href="#">tagSYSTEMTIME</a> for the creation of the file.
SysLastAccess	the type'd variable <a href="#">tagSYSTEMTIME</a> for the last access of the file.
SysLastWrite	the type'd variable <a href="#">tagSYSTEMTIME</a> for the last write of the file.
test%	is the <a href="#">returned code</a> .

## Comments :

For TIME2WIN, T2WIN-16 :

The created, access, modified time/date are the same because Win 3.xx don't handle the different date/time information.

## Examples :

**See also :** [File](#)

' structure for GetFileDateTime & SetFileDateTime

Type tagSYSTEMTIME

wYear	As Integer
wMonth	As Integer
wDayOfWeek	As Integer
wDay	As Integer
wHour	As Integer
wMinute	As Integer
wSecond	As Integer
wMilliseconds	As Integer

End Type

' definition for error type for GetFileDateTime

```
Public Const GET_FILE_DT_NO_ERROR = -1
Public Const GET_FILE_DT_CANT_OPEN_FILE = 8
Public Const GET_FILE_DT_CANT_GET_TIME = 9
Public Const GET_FILE_DT_CANT_CNV_CREATION = 1
Public Const GET_FILE_DT_CANT_CNV_LAST_ACCESS = 2
Public Const GET_FILE_DT_CANT_CNV_LAST_WRITE = 4
```

' definition for error type for SetFileDateTime

```
Public Const SET_FILE_DT_NO_ERROR = -1
Public Const SET_FILE_DT_CANT_OPEN_FILE = 8
Public Const SET_FILE_DT_CANT_SET_TIME = 9
Public Const SET_FILE_DT_CANT_CNV_CREATION = 1
Public Const SET_FILE_DT_CANT_CNV_LAST_ACCESS = 2
Public Const SET_FILE_DT_CANT_CNV_LAST_WRITE = 4
```



# GetFileDisplayName, GetFileTypeName

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT4.0x}, MSOffice 95

## Purpose :

[GetFileDisplayName](#) retrieves the EXPLORER (95/NT4.0x) display name of a file.

[GetFileTypeName](#) retrieves the EXPLORER (95/NT4.0x) type name of a file.

## Declare Syntax :

```
Declare Function cGetFileDisplayName Lib "time2win.dll" (ByVal nFileName As String) As String
Declare Function cGetFileTypeName Lib "time2win.dll" (ByVal nFileName As String) As String
```

## Call Syntax :

```
test$ = cGetFileDisplayName(nFileName$)
test$ = cGetFileTypeName(nFilename$)
```

## Where :

nFileName\$	the name of the file that you want retrieve the information.
test\$	the result.

## Comments :

## Examples :

```
Debug.Print cGetFileDisplayName("C:\WIN95\VBA.INI")           ' Vba.ini
Debug.Print cGetFileDisplayName("C:\WIN95\SYSTEM.INI")       ' System.ini
Debug.Print cGetFileDisplayName("C:\WIN95\NOCLOSE.PIF")' Noclose

Debug.Print cGetFileTypeName("C:\WIN95\VBA.INI")           ' Configuration Settings
Debug.Print cGetFileTypeName("C:\WIN95\SYSTEM.INI")       ' Configuration Settings
Debug.Print cGetFileTypeName("C:\WIN95\NOCLOSE.PIF")       ' Shortcut to MS-DOS Program
```

**See also** : [Windows 95](#)

' definition for error type for FileMergeExt

Public Const FILE\_MERGE\_NO\_ERROR = -1

Public Const FILE\_MERGE\_EMPTY\_TARGET\_FILENAME = -2

Public Const FILE\_MERGE\_EMPTY\_BAD\_TARGET\_FILENAME = -3

Public Const FILE\_MERGE\_CANT\_SET\_FILE\_BUFFER = -4

Public Const FILE\_MERGE\_CANT\_CREATE\_TARGET\_FILE = -5

Public Const FILE\_MERGE\_ERROR\_READING\_FILE = -6

Public Const FILE\_MERGE\_ERROR\_WRITING\_FILE = -7

Public Const FILE\_MERGE\_ERROR\_BAD\_WRITE = -8

Public Const FILE\_MERGE\_BAD\_FILE\_AT\_POSITION = -10

# ExplorerAddToRecentDocs, ExplorerClearAllRecentDocs

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT4.0x}](#), [MSOffice 95](#)

## Purpose :

[ExplorerAddToRecentDocs](#) adds a document to the shell's list of recently used documents.

[ExplorerClearAllRecentDocs](#) clears all documents from the shell's list of recently used documents.

## Declare Syntax :

```
Declare Sub cExplorerAddToRecentDocs Lib "time2win.dll" (ByVal nFileName As String)
Declare Sub cExplorerClearAllRecentDocs Lib "time2win.dll" ()
```

## Call Syntax :

```
Call cExplorerAddToRecentDocs(nFileName$)
Call cExplorerClearAllRecentDocs
```

## Where :

nFileName\$                    the name of the file (document) that you want add to the shell's list..

## Comments :

No errors are returned by these functions.

## Examples :

```
Call cExplorerClearAllRecentDocs
```

```
Call cExplorerAddToRecentDocs("f:\my documents\account.xls")
```

**See also** : [Windows 95](#)

```
Public Enum mcGradientGranularityEnum
    mcDefault = 0
    mcHigh = 1
    mcMedium = 2
    mcLow = 3
    mcVeryLow = 4
End Enum
```

# FilesCopy, FilesMove

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FilesCopy](#) copy files from one directory to an another directory/disk.

[FilesMove](#) move files from one directory to an another directory/disk.

## Declare Syntax :

Declare Function cFilesCopy Lib "[time2win.dll](#)" (ByVal SourcePath As String, ByVal DestinationPath As String, ByVal CopyPattern As String, ByVal Flags As Integer) As Long

Declare Function cFilesMove Lib "[time2win.dll](#)" (ByVal SourcePath As String, ByVal DestinationPath As String, ByVal MovePattern As String, ByVal Flags As Integer) As Long

## Call Syntax :

test& = cFilesCopy(SourcePath\$, DestinationPath\$, CopyPattern\$, Flags%)

test& = cFilesMove(SourcePath\$, DestinationPath\$, MovePattern\$, Flags%)

## Where :

SourcePath\$ is the source path;

DestinationPath\$ is the destination path;

CopyPattern\$/MovePattern\$ is the pattern of files to copy/move (\*.\* or \*.dat or \*.s?s)

Flags% is the combination of the following [value](#).

test& >= 0 if all is OK (the returned value is the number of files copied),

test& < 0 if an error has occurred (the returned value is the number of files that haven't been copied).

## Comments :

## Examples :

```
Debug.Print cFilesCopy("C:\WIN95", "C:\TMP", "*.LIC", FC_OVERWRITE_EXISTING Or  
FC_INCLUDE_SUB_DIRECTORY)
```

```
Debug.Print cFilesMove("C:\TMP", "C:\TEMP", "*.LIC", FM_REPLACE_EXISTING Or  
FC_INCLUDE_SUB_DIRECTORY)
```

**See also :** [File](#)

# Math : Overview

## **COMPLEX**

[CpxAdd](#)

adds two complex.

[CpxSub](#)

subtracts two complex.

[CpxMul](#)

multiplies two complex.

[CpxDiv](#)

divides two complex.

[CpxConjugue](#)

computes the conjugue of a complex.

[CpxModulus](#)

computes the modulus of a complex.

[CpxArgument](#)

computes the argument of a complex.

[CpxPowerN](#)

computes the power N of a complex

[CpxSqrt](#)

computes the square root of a complex.

## **ROOT**

[RootN](#)

computes the root N of a double number.

## **ALGOMATH**

[GreatCommonDivisor](#)

returns the Greatest Common Divisor of two numbers.

[Rotate](#)

rotates the digits of a number, x times, in base 10.

[SortDigits](#)

sorts the digits of a number.

[SumDigits](#)

returns the sum of the digits of a number.

[SumDigitsAlt](#)

returns the alternating sum of the digits of a number.

[SumDivisors](#)

returns the sum of all possible divisors of a number, number not included.

## **ROUND**

[Round](#)

rounds a number with a precision.

[RoundNearest](#)

rounds a number to the nearest value.

' success/error code

Public Const ERR\_STA\_NO\_ERROR = -1

Public Const ERR\_STA\_NO\_ITEMS\_TO\_SORT = 0

# SortTypedArray

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[SortTypedArray](#) sort a Type'd array which can contain one or more elements of any data type, array, or a previously defined user-defined type.

## Declare Syntax :

```
Declare Function cSortTypedArray Lib "time2win.dll" (TypedArray() As Any, ConfigArray As tagCONFIGARRAY) As Integer
```

## Call Syntax :

```
status% = cSortTypedArray(TArray(), CArray())
```

## Where :

TArray() is a type'd array which can contain one or more elements of any data type, array, or a previously defined user-defined type.

CArray() is the array for the configuration of the keys to be used to sort the array.

status% is the success/error code.

## Comments :

1) This function can sort any Type'd Array. However the sort can be only make with the following data type :

Boolean, Byte, Integer, Long, Single, Double, Currency, Date, Fixed/Variable String, Fixed/Variable String Number, Variant which encapsulate all the data type enumerated.

What is the Fixed/Variable String Number ?

Fixed/Variable String Number is a Fixed/Variable String which contains only numerical value. With this data type, the Fixed/Variable String Number is not sorted as a String but as a Number.

2) The Type'd array can be sorted with maximum **10** keys. Each key can be sorted by ASCending or DESCending order.

3) The 'KeyType' in the CArray() must be one of the following definition.

4) IMPORTANT NOTICE about the calculation of the 'KEYOFFSET' parameter in the 'CONFIGARRAY' for VB 4.0 (32-Bit) :

Don't forget that the 32-Bit version of VB (4.0 or Higher) packs the members of a user-defined type on a dword (4-byte) alignment boundary.

So, if you've the following user-defined type :

## Private Type UDT

intgr	As Integer
lng	As Long
sng	As Single
dbl	As Double
cur	As Currency
dtm	As Date
bln	As Boolean
byt	As Byte
vnt	As Variant



```

vstrg      As String
strg       As String * 11
End Type

```

The first member is an integer which is 2 bytes in size. Hence there is a padding of 2 bytes following it, as the next member is a long which is 4 bytes and cannot fit in the same "4-byte alignment boundary". The "sng" is 4 bytes (so, no padding). The "dbl" is 8 bytes (so, no padding). The "cur" is 8 bytes (so, no padding). The "dtm" is 8 bytes (so, no padding). The "bln" and "byt" member are packed together and there is a 1 byte padding after it. The "vnt" is 16 bytes (so, no padding). The "vstrg" is 4 bytes (so, no padding). Finally, the "strg" member in the VB UDT is a fixed length string of size 22 bytes as stored in VB and 11 bytes when passed to the DLL (so you need, to take care about this and calculate the length of a fixed string with the definition number of chars).

Each member in our user-defined type has the following offset :

```

intgr      ' offset 0      (internal size is 2)
                                     | -> padding of 2 bytes   | 4-byte alignment
boundary
lng        ' offset 4      (internal size is 4)
                                     | 4-byte alignment boundary
sng        ' offset 8      (internal size is 4)
                                     | 4-byte alignment boundary

dbl        ' offset 12     (internal size is 8)
                                     | 4-byte alignment boundary
cur        ' offset 20     (internal size is 8)
                                     | 4-byte alignment boundary
dtm        ' offset 28     (internal size is 8)
                                     | 4-byte alignment boundary
bln        ' offset 36     (internal size is 2)
byt        ' offset 38     (internal size is 1)
                                     | -> padding of 1 byte   | 4-byte alignment
boundary
vnt        ' offset 40     (internal size is 16)
                                     | 4-byte alignment
boundary
vstrg     ' offset 56     (internal size is 4)
                                     | 4-byte alignment boundary
strg      ' offset 60     (internal size is 11)
                                     | -> padding of 3 byte   | 4-byte alignment
boundary

```

So in our example, if I need to sort the type'd array on the following 7 keys : lng, dbl, byt, strg, vstrg, dtm, vnt. The ConfigArray is configured with the following parameters :

```

lng      KeyOffset(1) = 4   KeyLength(1) = -1 (auto)   KeyType(1) = STA_LONG
         KeyOrder(1) = 1 (asc)
dbl      KeyOffset(2) = 12  KeyLength(2) = -1 (auto)   KeyType(2) = STA_DOUBLE
         KeyOrder(2) = 1 (asc)
byt      KeyOffset(3) = 38  KeyLength(3) = -1 (auto)   KeyType(3) = STA_BYTE
         KeyOrder(3) = 1 (asc)
strg     KeyOffset(4) = 60  KeyLength(4) = 11          KeyType(4) = STA_FIXSTRING_CS
         KeyOrder(4) = -1 (desc)
vstrg   KeyOffset(5) = 56  KeyLength(5) = -1 (var)   KeyType(5) = STA_VARSTRING_CS
         KeyOrder(5) = 1 (asc)
dtm     KeyOffset(6) = 28  KeyLength(6) = -1 (auto)   KeyType(6) = STA_DATE
         KeyOrder(6) = -1 (desc)
vnt     KeyOffset(7) = 40  KeyLength(7) = -1 (auto/var) KeyType(7) = STA_VT_VARSTRING_CI
         KeyOrder(7) = 1 (asc)

```

5) IMPORTANT NOTICE about the calculation of the 'KEYOFFSET' parameter in the 'CONFIGARRAY' for VB 4.0 (16-Bit) :

Don't forget that the 16-Bit version of VB 4.0 DON'T pack the members of a user-defined type on a dword (4-byte) alignment boundary but on byte alignment boundary.

So, if you've the following user-defined type :

```

Private Type UDT
intgr      As Integer

```

```

lng      As Long
sng      As Single
dbl      As Double
cur      As Currency
dtm      As Date
bln      As Boolean
byt      As Byte
vnt      As Variant
vstrg    As String
strg     As String * 11
End Type

```

Each member in our user-defined type has the following offset :

```

intgr    ' offset 0      (internal size is 2)
lng      ' offset 2      (internal size is 4)
sng      ' offset 6      (internal size is 4)
dbl      ' offset 10     (internal size is 8)
cur      ' offset 18     (internal size is 8)
dtm      ' offset 26     (internal size is 8)
bln      ' offset 34     (internal size is 2)
byt      ' offset 36     (internal size is 1)
vnt      ' offset 37     (internal size is 16)
vstrg    ' offset 53     (internal size is 4)
strg     ' offset 57     (internal size is 11)

```

So in our example, if I need to sort the type'd array on the following 7 keys : lng, dbl, byt, strg, vstrg, dtm, vnt. The ConfigArray is configured with the following parameters :

```

lng      KeyOffset(1) = 2   KeyLength(1) = -1 (auto)   KeyType(1) = STA_LONG
         KeyOrder(1) = 1 (asc)
dbl      KeyOffset(2) = 10  KeyLength(2) = -1 (auto)   KeyType(2) = STA_DOUBLE
         KeyOrder(2) = 1 (asc)
byt      KeyOffset(3) = 36  KeyLength(3) = -1 (auto)   KeyType(3) = STA_BYTE
         KeyOrder(3) = 1 (asc)
strg     KeyOffset(4) = 57  KeyLength(4) = 11         KeyType(4) = STA_FIXSTRING_CS
         KeyOrder(4) = -1 (desc)
vstrg    KeyOffset(5) = 53  KeyLength(5) = -1 (var)   KeyType(5) = STA_VARSTRING_CS
         KeyOrder(5) = 1 (asc)
dtm      KeyOffset(6) = 26  KeyLength(6) = -1 (auto)   KeyType(6) = STA_DATE
         KeyOrder(6) = -1 (desc)
vnt      KeyOffset(7) = 40  KeyLength(7) = -1 (auto/var) KeyType(7) = STA_VT_VARSTRING_CI
         KeyOrder(7) = 1 (asc)

```

### Examples :

```

Private Type tagTESTARRAY
    i1      As Long
    i2      As Integer
    i3      As String
End Type

```

```
Dim intResult      As Integer
```

```
Dim i              As Integer
```

```
Dim TestArray(1 To 10) As tagTESTARRAY
Dim CA              As tagCONFIGARRAY
```

```
CA.KeyOffset(1) = 0 ' offset 0 (internal size is 4)
```

```
' for VB 4.0 (16-Bit) : offset 0 (internal size is
```

```
4)
CA.KeyOffset(2) = 4      ' offset 4 (internal size is 2) (padding of 2 bytes)      ' for VB 4.0 (16-Bit) : offset 4
                          (internal size is 2)
CA.KeyOffset(3) = 8      ' offset 8 (internal size is 4)                          ' for VB 4.0 (16-Bit) : offset 6 (internal size is
4)
```

```
CA.KeyLength(1) = -1    ' automatic internal size (4)
CA.KeyLength(2) = -1    ' automatic internal size (2)
CA.KeyLength(3) = -1    ' automatic internal size (variable)
```

```
CA.KeyType(1) = STA_LONG           ' Long
CA.KeyType(2) = STA_INTEGER        ' Integer
CA.KeyType(3) = STA_VARSTRING_CS   ' Variable length string Case-Sensitive
```

```
CA.KeyOrder(1) = -1            ' descending
CA.KeyOrder(2) = 1             ' ascending
CA.KeyOrder(3) = -1            ' descending
```

```
TestArray(1).i1 = 3: TestArray(1).i2 = 1: TestArray(1).i3 = "BBB"
TestArray(2).i1 = 9: TestArray(2).i2 = 7: TestArray(2).i3 = "ZZZ"
TestArray(3).i1 = 1: TestArray(3).i2 = 99: TestArray(3).i3 = "ZZZ"
TestArray(4).i1 = 1: TestArray(4).i2 = -1: TestArray(4).i3 = "AAA"
TestArray(5).i1 = 1: TestArray(5).i2 = -2: TestArray(5).i3 = "BBB"
TestArray(6).i1 = -1: TestArray(6).i2 = 102: TestArray(6).i3 = "aaa"
TestArray(7).i1 = 3: TestArray(7).i2 = 1: TestArray(7).i3 = "AAA"
TestArray(8).i1 = -1: TestArray(8).i2 = 102: TestArray(8).i3 = "bbb"
TestArray(9).i1 = -1: TestArray(9).i2 = 102: TestArray(9).i3 = "BBB"
TestArray(10).i1 = -1: TestArray(10).i2 = 102: TestArray(10).i3 = "AAA"
```

```
For i = 1 To 10
    List1.AddItem TestArray(i).i1 & vbTab & TestArray(i).i2 & vbTab & TestArray(i).i3
Next i
```

```
intResult = cSortTypedArray(TestArray(), CA)
```

```
For i = 1 To 10
    List2.AddItem TestArray(i).i1 & vbTab & TestArray(i).i2 & vbTab & TestArray(i).i3
Next i
```

**See Also :** [Array](#)

' structure for SortTypedArray

Type tagCONFIGARRAY

RecordLength	As Long	' internal use
ActualKey	As Integer	' internal use
PreviousKey	As Integer	' internal use
KeyOffset(1 To 7)	As Integer	' 0 is the higher key, 1 is the lower key
KeyLength(1 To 7)	As Integer	' length of the key (only for fixed string)
KeyType(1 To 7)	As Integer	' type of the key (see standard/variant data type)
KeyOrder(1 To 7)	As Integer	' -1 is reverse order, 0 is not used, 1 is ascending order

End Type

' standard data type

Public Const STA_BOOLEAN = 1	' internal size = 2
Public Const STA_BYTE = 2	' internal size = 1
Public Const STA_INTEGER = 3	' internal size = 2
Public Const STA_LONG = 4	' internal size = 4
Public Const STA_SINGLE = 5	' internal size = 4
Public Const STA_DOUBLE = 6	' internal size = 8
Public Const STA_CURRENCY = 7	' internal size = 8
Public Const STA_DATE = 8	' internal size = 8
Public Const STA_FIXSTRING_CI = 9	' internal size = * : fixed string length (case insensitive)
Public Const STA_FIXSTRING_CS = 10	' internal size = * : fixed string length (case sensitive)
Public Const STA_VARSTRING_CI = 11	' internal size = 4 : variable string length (case insensitive)
Public Const STA_VARSTRING_CS = 12	' internal size = 4 : variable string length (case sensitive)
Public Const STA_FIXSTRING_NUMBER = 13	' internal size = * : fixed string length (only number)
Public Const STA_VARSTRING_NUMBER = 14	' internal size = 4 : variable string length (only number)

' variant data type

Public Const STA_VT_BOOLEAN = -STA_BOOLEAN	' internal size = 16
Public Const STA_VT_BYTE = -STA_BYTE	' internal size = 16
Public Const STA_VT_INTEGER = -STA_INTEGER	' internal size = 16
Public Const STA_VT_LONG = -STA_LONG	' internal size = 16
Public Const STA_VT_SINGLE = -STA_SINGLE	' internal size = 16
Public Const STA_VT_DOUBLE = -STA_DOUBLE	' internal size = 16
Public Const STA_VT_CURRENCY = -STA_CURRENCY	' internal size = 16
Public Const STA_VT_DATE = -STA_DATE	' internal size = 16
Public Const STA_VT_FIXSTRING_CI = -STA_VARSTRING_CI	' internal size = 16
Public Const STA_VT_FIXSTRING_CS = -STA_VARSTRING_CS	' internal size = 16
Public Const STA_VT_VARSTRING_CI = -STA_VARSTRING_CI	' internal size = 16
Public Const STA_VT_VARSTRING_CS = -STA_VARSTRING_CS	' internal size = 16
Public Const STA_VT_FIXSTRING_NUMBER = -STA_VARSTRING_NUMBER	' internal size = 16
Public Const STA_VT_VARSTRING_NUMBER = -STA_VARSTRING_NUMBER	' internal size = 16

# DateToInt, IntToDate

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[DateToInt](#) compute a Int from all date parts.

[IntToDate](#) decompose a Int date into these components.

## Declare Syntax :

Declare Function cDateToInt Lib "[time2win.dll](#)" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Integer

Declare Sub cIntToDate Lib "[time2win.dll](#)" (ByVal Scalar As Integer, nYear As Integer, nMonth As Integer, nDay As Integer)

## Call Syntax :

Int% = cDateToInt(nYear%, nMonth%, nDay%)

Call cIntToDate(Scalar%, nYear%, nMonth%, nDay%)

## Where :

nYear%	is the year.
nMonth%	is the month.
nDay%	is the day.
Scalar%	is the returned computed Int.

## Comments :

The date is computed from 1997.01.01, so -32768 is 1997.01.01

For DateToInt :

If the parameters are not correct, the returned value is -1.

## Examples :

```
Dim Scalar As Integer
Dim nYear As Integer
Dim nMonth As Integer
Dim nDay As Integer
```

```
Scalar% = cDateToInt(1997, 1, 1) '-> -32768
```

```
Call cIntToDate(Scalar%, nYear%, nMonth%, nDay%)
```

```
' nYear% ' 1997
' nMonth% ' 1
' nDay% ' 1
```

**See also** : [Date and time](#)

# ShiftLeftD, ShiftLeftI, ShiftLeftL, ShiftLeftS ShiftRightD, ShiftRightI, ShiftRightL, ShiftRightS

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**ShiftLeftB** shift a **Byte** array to Left and set the last element to a value.  
**ShiftLeftD** shift a **Double** array to Left and set the last element to a value.  
**ShiftLeftI** shift an **Integer** array to Left and set the last element to a value.  
**ShiftLeftL** shift a **Long** array to Left and set the last element to a value.  
**ShiftLeftS** shift a **Single** array to Left and set the last element to a value.

**ShiftRightB** shift a **Byte** array to Right and set the first element to a value.  
**ShiftRightD** shift a **Double** array to Right and set the first element to a value.  
**ShiftRightI** shift an **Integer** array to Right and set the first element to a value.  
**ShiftRightL** shift a **Long** array to Right and set the first element to a value.  
**ShiftRightS** shift a **Single** array to Right and set the first element to a value.

## Declare Syntax :

```
Declare Function cShiftLeftB Lib "time2win.dll" (bArray() As Byte, ByVal bValue As Byte) As Integer
Declare Function cShiftLeftD Lib "time2win.dll" (dArray() As Double, ByVal dValue As Double) As Integer
Declare Function cShiftLeftI Lib "time2win.dll" (iArray() As Integer, ByVal iValue As Integer) As Integer
Declare Function cShiftLeftL Lib "time2win.dll" (lArray() As Long, ByVal lValue As Long) As Integer
Declare Function cShiftLeftS Lib "time2win.dll" (sArray() As Single, ByVal sValue As Single) As Integer
```

```
Declare Function cShiftRightB Lib "time2win.dll" (bArray() As Byte, ByVal bValue As Byte) As Integer
Declare Function cShiftRightD Lib "time2win.dll" (dArray() As Double, ByVal dValue As Double) As Integer
Declare Function cShiftRightI Lib "time2win.dll" (iArray() As Integer, ByVal iValue As Integer) As Integer
Declare Function cShiftRightL Lib "time2win.dll" (lArray() As Long, ByVal lValue As Long) As Integer
Declare Function cShiftRightS Lib "time2win.dll" (sArray() As Single, ByVal sValue As Single) As Integer
```

## Call Syntax :

```
status% = cShiftLeftB(bArray(), bValue)
status% = cShiftLeftD(dArray(), dValue)
status% = cShiftLeftI(iArray(), iValue)
status% = cShiftLeftL(lArray(), lValue)
status% = cShiftLeftS(sArray(), sValue)
```

```
status% = cShiftRightB(bArray(), bValue)
status% = cShiftRightD(dArray(), dValue)
status% = cShiftRightI(iArray(), iValue)
status% = cShiftRightL(lArray(), lValue)
status% = cShiftRightS(sArray(), sValue)
```

## Where :

xArray()	is the array (Byte, Double, Integer, Long, Single).
xValue element.	is the value (Byte, Double, Integer, Long, Single) to set at the last(ShiftLeft)/first(ShiftRight)
status%	always TRUE

## Comments :

ShiftLeftB and ShiftRightB are not supported for VB 3.0 and VB 4.0 (16-Bit).

**See Also** : [Array](#)

# Convert

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Convert](#) perform conversion between date, time, hour, minute, hundred.

## Declare Syntax :

Declare Function cConvert Lib "[time2win.dll](#)" (ByVal WhichConversion As Integer, WhichValue As Variant, ConvertError As Integer) As String

## Call Syntax :

```
test$ = cConvert(WhichConversion%, WhichValue, ConvertError%)
```

## Where :

WhichConversion%	is the conversion to perform (see <a href="#">constant</a> )
WhichValue	is the variant value to convert
ConvertError%	is the return <a href="#">success/error code</a>
test\$	is the result inf a form of a string

## Comments :

## Examples :

```
Debug.Print cConvert(CNV_BALANCE_FILL_ZERO_AND_WITH_ALWAYS_SIGN, "1172", ConvertError)
'-> +00019:32
```

```
Debug.Print cConvert(CNV_HUNDRED_BALANCE_FILL_ZERO_AND_WITH_ALWAYS_SIGN, "1172",
ConvertError)
'-> +00019.53
```

```
Debug.Print cConvert(CNV_DATE_LONG, "729057", ConvertError)
'-> 02.02.1997
```

```
Debug.Print cConvert(CNV_DATE_SHORT, "-32736", ConvertError)
'-> 02.02.1997
```

**See also** : [Miscellaneous](#)



' definition for shortcut file

```
Public Const SHORTCUTFILE_GETPATH = 1  
Public Const SHORTCUTFILE_GETPATH83 = 2  
Public Const SHORTCUTFILE_GETWORKDIR = 3  
Public Const SHORTCUTFILE_GETDESCRIPTION = 4  
Public Const SHORTCUTFILE_GETARGUMENTS = 5
```

```
Public Const SHORTCUTFILE_SETPATH = -1  
Public Const SHORTCUTFILE_SETWORKDIR = -2  
Public Const SHORTCUTFILE_SETDESCRIPTION = -3  
Public Const SHORTCUTFILE_SETARGUMENTS = -4
```

' definition for error code for CONVERT

Public Const CNV\_NO\_ERROR = -1  
Public Const CNV\_ERROR\_NO\_CONVERSION\_AVAILABLE = 0  
Public Const CNV\_ERROR\_CONVERSION\_TO\_STRING = 1  
Public Const CNV\_ERROR\_TIME\_OUTSIDE\_LIMIT = 2  
Public Const CNV\_ERROR\_MISSING\_TIME\_SEPARATOR = 3  
Public Const CNV\_ERROR\_BAD\_HOUR = 4  
Public Const CNV\_ERROR\_BAD\_MINUTE = 5  
Public Const CNV\_ERROR\_MISSING\_DATE\_SEPARATOR = 6  
Public Const CNV\_ERROR\_BAD\_DATE = 7

' definition for conversion parameter for CONVERT

Public Const CNV\_TIME\_FILL\_ZERO = 100  
Public Const CNV\_TIME = 101

Public Const CNV\_HUNDRED\_TIME\_FILL\_ZERO = 105  
Public Const CNV\_HUNDRED\_TIME = 106

Public Const CNV\_HOUR\_FILL\_ZERO\_AND\_WITH\_ALWAYS\_SIGN = 110  
Public Const CNV\_HOUR\_FILL\_ZERO\_AND\_WITH\_SIGN = 111  
Public Const CNV\_HOUR\_WITH\_ALWAYS\_SIGN = 112  
Public Const CNV\_HOUR\_AND\_WITH\_SIGN = 113

Public Const CNV\_HUNDRED\_HOUR\_FILL\_ZERO\_AND\_WITH\_ALWAYS\_SIGN = 115  
Public Const CNV\_HUNDRED\_HOUR\_FILL\_ZERO\_AND\_WITH\_SIGN = 116  
Public Const CNV\_HUNDRED\_HOUR\_WITH\_ALWAYS\_SIGN = 117  
Public Const CNV\_HUNDRED\_HOUR\_AND\_WITH\_SIGN = 118

Public Const CNV\_BALANCE\_FILL\_ZERO\_AND\_WITH\_ALWAYS\_SIGN = 120  
Public Const CNV\_BALANCE\_FILL\_ZERO\_AND\_WITH\_SIGN = 121  
Public Const CNV\_BALANCE\_WITH\_ALWAYS\_SIGN = 122  
Public Const CNV\_BALANCE\_AND\_WITH\_SIGN = 123

Public Const CNV\_HUNDRED\_BALANCE\_FILL\_ZERO\_AND\_WITH\_ALWAYS\_SIGN = 125  
Public Const CNV\_HUNDRED\_BALANCE\_FILL\_ZERO\_AND\_WITH\_SIGN = 126  
Public Const CNV\_HUNDRED\_BALANCE\_WITH\_ALWAYS\_SIGN = 127  
Public Const CNV\_HUNDRED\_BALANCE\_AND\_WITH\_SIGN = 128

Public Const CNV\_DATE\_SHORT = 200  
Public Const CNV\_DATE\_LONG = 201

# DateHourToLong, LongToDateHour

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[DateHourToLong](#) compute a Long from all date-hour parts.

[LongToDateHour](#) decompose a Long date-hour into these components.

## Declare Syntax :

```
Declare Function cDateHourToLong Lib "time2win.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer, ByVal nHour As Integer, ByVal nMinute As Integer, ByVal nExtra As Integer) As Long
```

```
Declare Sub cLongToDateHour Lib "time2win.dll" (ByVal Scalar As Long, nYear As Integer, nMonth As Integer, nDay As Integer, nHour As Integer, nMinute As Integer, nExtra As Integer)
```

## Call Syntax :

```
DateHour& = cDateHourToLong(nYear%, nMonth%, nDay%, nHour%, nMinute%, nExtra%)
```

```
Call cLongToDateHour(DateHour&, nYear%, nMonth%, nDay%, nHour%, nMinute%, nExtra%)
```

## Where :

nYear%	is the year (starting from 1900);
nMonth%	is the month;
nDay%	is the day;
nHour%	is the hour;
nMinute%	is the minute;
nExtra%	is the extra part of a DateHour;
DateHour&	is the returned computed Long.

## Comments :

Maximum value for each item in DateHour :

**Year** must be between **1900** and **2079**.

**Month** must be between **1** and **12**.

**Day** must be between **1** and **31**.

**Hour** must be between **0** and **23**.

**Minute** must be between **0** and **59**.

**Extra** must be between **0** and **31**.

The date is computed from **1900.01.01 00:00** to **2079.06.06 23:59** but no verifications are performed on input.

You can use this date-hour data type to perform sorting operation on date because the internal members of this data type are :

Date, Hour, Extra in this order.

## Examples :

```
Dim DateHour As Long
Dim nYear As Integer
Dim nMonth As Integer
Dim nDay As Integer
Dim nHour As Integer
Dim nMinute As Integer
Dim nExtra As Integer
```

```
DateHour& = cDateHourToLong(1997, 1, 1, 23, 59, 31) '-> -1973046273
```

Call `cLongToDateHour(DateHour&, nYear%, nMonth%, nDay%, nHour%, nMinute%, nExtra%)`

' nYear%        is 1997  
' nMonth%       is 1  
' nDay%         is 1  
' nHour%        is 23  
' nMinute%      is 59  
' nExtra%       is 7

**See also :** [Date and time](#)

# MultitasksKeys

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT4.0x}](#), [MSOffice 95](#)

## Purpose :

[MultitasksKeys](#) disables/enables CTRL+ALT+DEL, ALT+TAB and CTRL+ESC.

## Declare Syntax :

For VB 32-Bit :

```
Declare Sub cMultitasksKeys Lib "t2win-32.dll" (ByVal Status As Integer)
```

For VB 16-Bit :

```
Declare Sub cMultitasksKeys95 Lib "time2win.dll" (ByVal Status As Integer)
```

## Call Syntax :

```
Result% = cMultitasksKeys(Status%)
```

## Where :

Status%	TRUE to disable the MultiTasks Keys; FALSE to enable the MultiTasks Keys.
---------	------------------------------------------------------------------------------

## Comments :

**Don't forget to Re-Enable the MultiTasks Keys when your program terminates.**

## Examples :

Call cMultitasksKeys(True)'-> so, CTRL+ALT+DEL, ALT+TAB and CTRL+ESC are disabled

Call cMultitasksKeys(False)       '-> so, CTRL+ALT+DEL, ALT+TAB and CTRL+ESC are enabled

**See also :** [Windows 95](#)

' constants for TypeOfEncoding in cMNEncode and cMNWalkInList

Public Const MN_UU_ENCODED = 1	'UUencoded data
Public Const MN_B64_ENCODED = 2	'Mime-Base64 data
Public Const MN_XX_ENCODED = 3	'XXencoded data
Public Const MN_BH_ENCODED = 4	'Binhex encoded
Public Const MN_PT_ENCODED = 5	'Plain-Text encoded (MIME)
Public Const MN_QP_ENCODED = 6	'Quoted-Printable (MIME)

This function works with the following software :

VB 4.0 (32-Bit)	under Win95/WinNT
VB 5.0	under Win95/WinNT
VBA 5.0	under Win95/WinNT
Msoffice97	under Win95/WinNT

and

All softwares supporting VBA 5.0

# FileForceCopy

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileForceCopy](#) copy one file to an another file with the same file attribute.

## Declare Syntax :

Declare Function cFileForceCopy Lib "[time2win.dll](#)" (ByVal File1 As String, ByVal File2 As String, ByVal ForceOrNot As Integer) As Long

## Call Syntax :

```
test& = cFileCopy(file1, file2, ForceOrNot%)
```

## Where :

file1\$	is the source file.
file2\$	is the destination file.
ForceOrNot%	TRUE to force to copy with the same file attribute FALSE : same as <a href="#">cFileCopy</a>
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occurred.

## Comments :

The returned value can be negative and have the following value :

-1	can't set file attribute
-32720	the number of chars in a block for writing differs from the number of chars for reading.
-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.
-32760	allocation error for memory buffer.

## Examples :

```
test& = cFileForceCopy("c:\autoexec.bat", "c:\autoexec.tab", True)
```

**See also :** [File](#)



# ShortcutFileGetX, ShortcutFileSetX

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT4.0x}, MSOffice 95

## Purpose :

[ShortcutCreate](#) creates a shortcut from a specified file.

[ShortcutFileGetArguments](#) retrieves the argument list from a shortcut file (\*.lnk).

[ShortcutFileGetDescription](#) retrieves the description from a shortcut file (\*.lnk).

[ShortcutFileGetIconLocation](#) retrieves the icon location from a shortcut file (\*.lnk).

[ShortcutFileGetPath](#) retrieves the path from a shortcut file (\*.lnk).

[ShortcutFileGetPath83](#) retrieves the path (in format 8.3) from a shortcut file (\*.lnk).

[ShortcutFileGetWorkingDir](#) retrieves the working directory from a shortcut file (\*.lnk).

[ShortcutFileGetInfo](#) is a generic function to retrieve the above parameter from a shortcut file (\*.lnk).

[ShortcutFileSetArguments](#) modifies the argument list in a shortcut file (\*.lnk).

[ShortcutFileSetDescription](#) modifies the description in a shortcut file (\*.lnk).

[ShortcutFileSetIconLocation](#) modifies the icon location in a shortcut file (\*.lnk).

[ShortcutFileSetPath](#) modifies the path in a shortcut file (\*.lnk).

[ShortcutFileSetWorkingDir](#) modifies the working directory in a shortcut file (\*.lnk).

[ShortcutFileSetInfo](#) is a generic function to modify the above parameter in a shortcut file (\*.lnk).

## Declare Syntax :

```
Declare Function cShortcutFileGetInfo Lib "time2win.dll" (ByVal pszShortcutFile As String, ByVal nFunction As Integer) As String
```

```
Declare Function cShortcutFileGetArguments Lib "time2win.dll" (ByVal pszShortcutFile As String) As String
```

```
Declare Function cShortcutFileGetDescription Lib "time2win.dll" (ByVal pszShortcutFile As String) As String
```

```
Declare Function cShortcutFileGetIconLocation Lib "time2win.dll" (ByVal pszShortcutFile As String, lIconOffset As Long) As String
```

```
Declare Function cShortcutFileGetPath Lib "time2win.dll" (ByVal pszShortcutFile As String) As String
```

```
Declare Function cShortcutFileGetPath83 Lib "time2win.dll" (ByVal pszShortcutFile As String) As String
```

```
Declare Function cShortcutFileGetWorkingDir Lib "time2win.dll" (ByVal pszShortcutFile As String) As String
```

```
Declare Function cShortcutFileSetInfo Lib "time2win.dll" (ByVal pszShortcutFile As String, ByVal pszValue As String, ByVal nFunction As Integer) As Integer
```

```
Declare Function cShortcutFileSetArguments Lib "time2win.dll" (ByVal pszShortcutFile As String, ByVal pszValue As String) As Integer
```

```
Declare Function cShortcutFileSetDescription Lib "time2win.dll" (ByVal pszShortcutFile As String, ByVal pszValue As String) As Integer
```

```
Declare Function cShortcutFileSetIconLocation Lib "time2win.dll" (ByVal pszShortcutFile As String, ByVal pszValue As String, ByVal lIconOffset As Long) As Integer
```

```
Declare Function cShortcutFileSetPath Lib "time2win.dll" (ByVal pszShortcutFile As String, ByVal pszValue As String) As Integer
```

```
Declare Function cShortcutFileSetWorkingDir Lib "time2win.dll" (ByVal pszShortcutFile As String, ByVal pszValue As String) As Integer
```

## Call Syntax :

```
test$ = cShortcutFileGetInfo(pszShortcutFile$, nFunction%)
```

```
test$ = cShortcutFileGetArguments(pszShortcutFile$)
```

```
test$ = cShortcutFileGetDescription(pszShortcutFile$)
```

```
test$ = cShortcutFileGetIconLocation(pszShortcutFile$, lIconOffset&)
```

```
test$ = cShortcutFileGetPath(pszShortcutFile$)
```

```
test$ = cShortcutFileGetPath83(pszShortcutFile$)
```

```
test$ = cShortcutFileGetWorkingDir(pszShortcutFile$)
```

```
test% = cShortcutFileSetInfo(pszShortcutFile$, pszValue$, nFunction%)
```

```
test% = cShortcutFileSetArguments(pszShortcutFile$, pszValue$)
```

```
test% = cShortcutFileSetDescription(pszShortcutFile$, pszValue$)
```

```
test% = cShortcutFileSetIconLocation(pszShortcutFile$, pszValue$, lIconOffset&)
test% = cShortcutFileSetPath(pszShortcutFile$, pszValue$)
test% = cShortcutFileSetWorkingDir(pszShortcutFile$, pszValue$)
```

**Where :**

```
pszShortcutFile$    the shell link's file (*.lnk).
nFunction%         the generic parameter (see parameter).
pszValue$          the new value
lIconOffset&       is the icon offset in the file which contains the icon location
test$              the result for ShortcutFileGetX.
test%              the result for ShortcutFileSetX :
                   -1 : no error
                   0 : an error has occurred
```

**Comments :**

**Examples :**

```
    Debug.Print cShortcutFileGetPath("D:\WIN95\Startup\WinZip.lnk")           ' D:\Program Files\WinZip\
WINZIP32.EXE
    Debug.Print cShortcutFileGetPath83("D:\WIN95\Startup\WinZip.lnk")       ' D:\PROGRA~1\WINZIP\
WINZIP32.EXE
    Debug.Print cShortcutFileGetDescription("D:\WIN95\Startup\WinZip.lnk")   ' WinZip

Dim lIconOffset    As Long

' create a new shortcut in the directory \WIN95\SENDTO
Debug.Print cShortcutCreate("c:\temp\shortcut.test", "t2wlink.lnk", SC_SENDTO)      '-1 = ok

' create a new shortcut on the DESKTOP (in fact \WIN95\DESKTOP)
Debug.Print cShortcutCreate("c:\temp\shortcut.test", "t2wlink.lnk", SC_DESKTOP)    '-1 = ok

Debug.Print cShortcutFileGetIconLocation("d:\win95\sendto\t2wlink.lnk", lIconOffset), lIconOffset ' "", 0
(because no associated icon)
Debug.Print cShortcutFileSetIconLocation("d:\win95\sendto\t2wlink.lnk", "shell32.dll", 1)      '-1 = ok
Debug.Print cShortcutFileGetIconLocation("d:\win95\sendto\t2wlink.lnk", lIconOffset), lIconOffset ' "shell32.dll" 1
(icon offset = 1)

Debug.Print cShortcutFileGetIconLocation("d:\win95\desktop\t2wlink.lnk", lIconOffset), lIconOffset ' "", 0
(because no associated icon)
Debug.Print cShortcutFileSetIconLocation("d:\win95\desktop\t2wlink.lnk", "shell32.dll", 3)      '-1 = ok
Debug.Print cShortcutFileGetIconLocation("d:\win95\desktop\t2wlink.lnk", lIconOffset), lIconOffset '
"shell32.dll" 1 (icon offset = 1)
```

**See also :** [Windows 95](#)

# LockKeyboard, LockMouse

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT4.0x}](#), [MSOffice 95](#)

## Purpose :

[LockKeyboard](#) locks/unlocks the keyboard for any applications.

[LockMouse](#) locks/unlocks the mouse (any events : left and/or middle and/or right click) for any applications.

## Declare Syntax :

```
Declare Function cLockKeyboard Lib "t2win-32.dll" (ByVal LockUnlock As Integer, ByVal SpecialUnlockKey As Integer) As Integer
```

```
Declare Function cLockMouse Lib "t2win-32.dll" (ByVal LockUnlock As Integer, ByVal ClickToBeHooked As Integer) As Integer
```

## Call Syntax :

```
intResult% = cLockKeyboard(LockUnlock%, SpecialUnlockKey%)
```

```
intResult% = cLockMouse(LockUnlock%, ClickToBeHooked%)
```

## Where :

LockUnlock%	TRUE to disable the keyboard/mouse; FALSE to enable the keyboard/mouse.
SpecialUnlockKey%	code of the key to disable the locking of the keyboard (in case of ...).
ClickToBeHooked%	type of <a href="#">click</a> to be hooked.

## Comments :

1) There is no need to call this function with the FALSE parameter when you stop your program. The lock of the keyboard/mouse will be automatically removed when TIME TO WIN 32-Bit will be removed from the memory.

2) When you set a SpecialUnlockKey, you must press ALT+CTRL+SHIFT+[SpecialUnlockKey] to unlock the keyboard.

3) Don't forget that the multi task keys (ALT+CTRL+DEL, ALT+TAB, CTRL+ESC) are not handled by this routine, so use the [cMultitaskKeys](#) function to disable it.

## Examples :

```
Debug.Print cLockKeyboard(True, &H70) ' lock the keyboard and set the
special unlock key to F1 (use ALT+CTRL+SHIFT+F1).
Debug.Print cSleep(7000) ' sleep the current task for 7
seconds.
Debug.Print cLockKeyboard(False, 0) ' unlock the keyboard.

Debug.Print cLockMouse(True, LOCK_MOUSE_ALL_CLICK) ' lock the mouse.
Debug.Print cSleep(7000) ' sleep the current task for 7
seconds.
Debug.Print cLockMouse(False, 0) ' unlock the mouse.
```

**See also :** [Windows 95](#)

' constants for LockMouse

Public Const LOCK\_MOUSE\_LCLICK = 1

Public Const LOCK\_MOUSE\_MCLICK = 2

Public Const LOCK\_MOUSE\_RCLICK = 4

Public Const LOCK\_MOUSE\_ALL\_CLICK = LOCK\_MOUSE\_LCLICK Or LOCK\_MOUSE\_MCLICK Or  
LOCK\_MOUSE\_RCLICK

# LFill, RFill

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

**LFill** pads a string to the left, to a know length.

**RFill** pads a string to the left, to a know length.

## Declare Syntax :

Declare Function cLFill Lib "[time2win.dll](#)" (ByVal Txt As String, ByVal CharToFill As String, ByVal Length As Long) As String

Declare Function cRFill Lib "[time2win.dll](#)" (ByVal Txt As String, ByVal CharToFill As String, ByVal Length As Long) As String

## Call Syntax :

```
test$ = cLFill(Txt$, CharToFill$, Length&)
```

```
test$ = cRFill(Txt$, CharToFill$, Length&)
```

## Where :

Txt\$	the string to proceed.
CharToFill\$	the pad char
Length	the new length of the string.
test\$	the result.

## Comments :

## Examples :

```
test$ = cLFill("1234", "0", 10)      -> "0000001234"  
test$ = cRFill("1234", "0", 10)     -> "1234000000"
```

```
test$ = cLFill("TEST", " ", 10)     -> "    TEST"  
test$ = cRFill("TEST", " ", 10)     -> "TEST   "
```

```
test$ = cLFill("TEST", "_", 12)     -> "_____1234"  
test$ = cRFill("TEST", "_", 12)     -> "1234_____"
```

**See also :** [String](#)

# MatchTable

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[MatchTable](#) compares a string with a set of strings delimited by a separator.

## Declare Syntax :

Declare Function cMatchTable Lib "time2win.dll" (ByVal Txt As String, ByVal Match As String, ByVal Separator As String, ByVal Sensitivity As Integer) As Integer

## Call Syntax :

test% = cMatchTable(Txt\$, Match\$, Separator\$, Sensitivity%)

## Where :

Txt\$	the string to compare.
Match\$	the set of strings used to compare.
Separator\$	the separator in the set of strings.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
test%	the position of the string.

## Comments :

If the string can't be found in the set of strings, the returned value is -1.

## Examples :

```
test$ = cMatchTable("A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "bc", "/", True)    -> 7
test$ = cMatchTable("A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "bc", "/", False) -> 2

test$ = cMatchTable("A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "BC", "/", True)  -> 2
test$ = cMatchTable("A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "BC", "/", False) -> 2

test$ = cMatchTable("A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "abc", "/", True)  -> -1
test$ = cMatchTable("A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "abc", "/", False) -> -1
```

**See also :** [String](#)

# LSetIn, RSetIn

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**LSetIn** inserts a string (to the left) into a sub-string delimited by a separator in a given string.

**RSetIn** inserts a string (to the right) into a sub-string delimited by a separator in a given string.

## Declare Syntax :

Declare Function cLSetIn Lib "time2win.dll" (ByVal Txt As String, ByVal Separator As String, ByVal Position As Long, ByVal StringToInsert As String) As String

Declare Function cRSetIn Lib "time2win.dll" (ByVal Txt As String, ByVal Separator As String, ByVal Position As Long, ByVal StringToInsert As String) As String

## Call Syntax :

```
test$ = cLSetIn(Txt$, Separator$, Position&, StringToInsert$)
```

```
test$ = cRSetIn(Txt$, Separator$, Position&, StringToInsert$)
```

## Where :

Txt\$	the string to proceed.
Separator\$	the delimiter for each sub-string.
Position&	the position of the sub-string.
StringToInsert\$	the string to be inserted.
test\$	the result.

## Comments :

\* If the size of the string is 0 or if the position is < 1 or greater than the maximum block is the string or if the length is 0. The returned string is an empty string.

## Examples :

```
test$ = cLSetIn("A/BC/DEF/G", "/", 4, "***")      -> "A/BC/DEF/***G"  
test$ = cRSetIn("A/BC/DEF/G", "/", 4, "***")     -> "A/BC/DEF/G***"
```

```
test$ = cLSetIn("A/BC/DEF/G", "/", 1, "**")       -> "**A/BC/DEF/G"  
test$ = cRSetIn("A/BC/DEF/G", "/", 1, "**")       -> "A*/BC/DEF/G"
```

```
test$ = cLSetIn("A/BC/DEF/G", "/", 0, "**")       -> "A/BC/DEF/G"  
test$ = cRSetIn("A/BC/DEF/G", "/", 0, "**")       -> "A/BC/DEF/G"
```

```
test$ = cLSetIn("A/BC/DEF/G", "/", 2, "")        -> "A/BC/DEF/G"  
test$ = cRSetIn("A/BC/DEF/G", "/", 2, "")        -> "A/BC/DEF/G"
```

**See also** : [String](#)

# StringReplace

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[StringReplace](#) searches for known strings, and replaces them with another string.

## **Declare Syntax :**

Declare Function cStringReplace Lib "[time2win.dll](#)" (ByVal Text As String, ByVal Pattern As String, ByVal Sensitivity As Integer) As String

## **Call Syntax :**

```
test$ = cStringReplace(Txt$, Pattern$)
```

## **Where :**

Txt\$	the string to proceed.
Pattern\$	the pattern string under the form "xxx yyy;aaa bbb";... ..."
test\$	the result.

## **Comments :**

The length of the returned string can't be greater than 25 times the length of the Txt string.

## **Examples :**

```
test$ = cStringReplace("thsi si a test and this is na naother test", "si|is;na|an")  
-> "this is a test and this is an another test"
```

```
test$ = cStringReplace("this si a test and this is na another test", "this si a test|this is an another test;this is na  
another test|this is really an another test")  
-> "this is an another test and this is really an another  
test"
```

**See also :** [String](#)



# InStr

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[InStr](#) Finds the position of the first occurrence of one string within another (like VB Instr function).

## Declare Syntax :

Declare Function cInStr Lib "[time2win.dll](#)" (ByVal StartPosition As Long, ByVal Txt As String, ByVal Search As String, ByVal Sensitivity As Integer) As Long

## Call Syntax :

test& = cInStr(StartPosition&, Txt\$, Search\$, Sensitivity%)

## Where :

StartPosition&	numeric expression that sets the starting position for each search.
Txt\$	string expression being searched.
Search\$	string expression sought.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
test&	the position of the string.

## Comments :

If Txt\$ is zero-length, the returned value is 0  
If Search\$ is zero-length, the returned value is StartPosition  
If Search\$ is not found, the returned value is 0  
If Search\$ is found within Txt\$, the returned value is the position at which match is found  
If StartPosition > Len(Txt\$), the returned value is 0  
If StartPosition < 1, the returned value is -1

## Examples :

```
test$ = cInStr(1, "A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "bc", True)           '-> 24
test$ = cInStr(1, "A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "bc", False)        '-> 3

test$ = cInStr(7, "A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "bc", True)         '-> 24
test$ = cInStr(7, "A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "bc", False)       '-> 24

test$ = cInStr(0, "A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "bc", True)        '-> -1

test$ = cInStr(128, "A/BC/DEF/GHIJ/KLMNOP/a/bc/def/ghij/klmnop", "bc", True)      '-> 0
```

**See also :** [String](#)

' constants for error processing

Public Const MNRET\_OK = 0  
Public Const MNRET\_IOERR = 1  
Public Const MNRET\_NOMEM = 2  
Public Const MNRET\_ILLVAL = 3  
Public Const MNRET\_NODATA = 4  
Public Const MNRET\_NOEND = 5  
Public Const MNRET\_UNSUP = 6  
Public Const MNRET\_EXISTS = 7  
Public Const MNRET\_CONT = 8  
Public Const MNRET\_CANCEL = 9

'everything went fine  
'I/O Error - examine errno  
'not enough memory  
'illegal value for operation  
'decoder didn't find any data  
'encoded data wasn't ended properly  
'unsupported function (encoding)  
'file exists (decoding)  
'continue -- special from ScanPart  
'operation canceled

# MNInitialize, MNShutdown

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[MNInitialize](#) initializes the Mail & News memory to perform encoding/decoding.  
[MNShutdown](#) shutdowns the Mail & News memory (cleanup).

## **Declare Syntax :**

```
Declare Function cMNInitialize Lib "time2win.dll" () As Long  
Declare Function cMNShutdown Lib "time2win.dll" () As Long
```

## **Call Syntax :**

```
IReturn& = cMNInitialize()  
IReturn& = cMNShutdown()
```

## **Where :**

IReturn& is the MNRET\_ code

## **Comments :**

## **Examples :**

**See also** : [Mail & News](#)

# MNEncode

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[MNEncode](#) encodes a file in single-part or multi-part using "UUE", "Mime-Base64", "XXE"..

## Declare Syntax :

Declare Function cMNEncode Lib "[time2win.dll](#)" (ByVal FileIn As String, ByVal FileOut As String, ByVal NameInFile As String, ByVal TypeOfEncoding As Long, ByVal LinesPerFile As Long, ByVal MimeHeaders As Integer, ByVal Sender As String, ByVal Receiver As String, ByVal Subject As String) As Long

## Call Syntax :

IReturn& = cMNEncode(FileIn\$, FileOut\$, NameInFile\$, TypeOfEncoding&, LinesPerFiles&, MimeHeaders%, Sender\$, Receiver\$, Subject\$)

## Where :

FileIn\$	is the file to encode
FileOut\$	is the result file
NameInFile\$	is the name of the file to be placed in the result file (can be usefull for the decoding)
TypeOfEncoding&	is the encoding method (see <a href="#">MN_???_ENCODED</a> code) (at this time, only UUE, XXE, Mime-Base64 are supported)
LinesPerFiles&	is the number of lines per file (0 = single-part, >0 = multi-part)
MimeHeaders%	is the header to be placed (see <a href="#">MNHDR_</a> code)
Sender\$	is the sender of the message
Receiver\$	is the receiver of the message
Subject\$	is the subject of the message
IReturn&	< 0 : negative values of the <a href="#">MNRET_</a> code
	>=0 : size of the result file

## Comments :

## Examples :

**See also** : [Mail & News](#)

# MNGetOption, MNSetOption

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[MNGetOption](#) retrieves options.

[MNSetOption](#) sets options.

## Declare Syntax :

Declare Function cMNGetOption Lib "[time2win.dll](#)" (ByVal OptionNo As Long, LongVal As Long, StringVal As String) As Integer

Declare Function cMNSetOption Lib "[time2win.dll](#)" (ByVal OptionNo As Long, ByVal LongVal As Long, ByVal StringVal As String) As Integer

## Call Syntax :

iReturn% = cMNGetOption(OptionNo&, LongVal&, StringVal\$)

## Where :

OptionNo&	is the option to retrieve or set (see <a href="#">MNOPT_</a> code)
LongVal&	is the value of a long option to retrieve or set
StringVal\$	is the value of a string option to retrieve or set
iReturn%	True if all is OK False if an error has occurred

## Comments :

## Examples :

**See also :** [Mail & News](#)

# MNDecodeFromList

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[MNDecodeFromList](#) decodes a file from the list of files using "UUE", "Mime-Base64/PT/QP", "XXE", "BinHex".

## Declare Syntax :

Declare Function cMNDecodeFromList Lib "[time2win.dll](#)" (ByVal Ptr As Long, FileState As Long, ErrorCodeForDecodeOperation As Long, ErrorCodeForMoveOperation As Long) As Long

## Call Syntax :

IResult& = cMNDecodeFromList(Ptr&, FileState&, ErrorCodeForDecodeOperation&, ErrorCodeForMoveOperation&)

## Where :

Ptr&		is the file pointer from <a href="#">cWalkInList</a>
FileState&		is an OR'ed combination of <a href="#">MNSTATE_</a> code
ErrorCodeForDecodeOperation&		(not yet implemented)
ErrorCodeForMoveOperation&		(not yet implemented)
IReturn&	< 0	: negative values of the <a href="#">MNRET_</a> code
	>=0	: size of the result file

## Comments :

## Examples :

see Mail & News in the demo program

**See also** : [Mail & News](#)

# Math : Complex

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[CpxAdd](#) adds two complex.  
[CpxSub](#) subtracts two complex.  
[CpxMul](#) multiplies two complex.  
[CpxDiv](#) divides two complex.  
[CpxConjugue](#) computes the conjugue of a complex.  
[CpxModulus](#) computes the modulus of a complex.  
[CpxArgument](#) computes the argument of a complex.  
[CpxPowerN](#) computes the power N of a complex.  
[CpxSqrt](#) computes the square root of a complex.

## Declare Syntax :

```
Type tagCOMPLEX
    a           As Double
    b           As Double
End Type
```

```
Declare Function cCpxAdd Lib "time2win.dll" (z1 As tagCOMPLEX, z2 As tagCOMPLEX) As tagCOMPLEX
Declare Function cCpxSub Lib "time2win.dll" (z1 As tagCOMPLEX, z2 As tagCOMPLEX) As tagCOMPLEX
Declare Function cCpxMul Lib "time2win.dll" (z1 As tagCOMPLEX, z2 As tagCOMPLEX) As tagCOMPLEX
Declare Function cCpxDiv Lib "time2win.dll" (z1 As tagCOMPLEX, z2 As tagCOMPLEX) As tagCOMPLEX
Declare Function cCpxConjugue Lib "time2win.dll" (z1 As tagCOMPLEX) As tagCOMPLEX
Declare Function cCpxModulus Lib "time2win.dll" (z1 As tagCOMPLEX) As Double
Declare Function cCpxArgument Lib "time2win.dll" (z1 As tagCOMPLEX) As Double
Declare Function cCpxPowerN Lib "time2win.dll" (z1 As tagCOMPLEX, ByVal n As Integer) As tagCOMPLEX
Declare Function cCpxSqrt Lib "time2win.dll" (z1 As tagCOMPLEX) As tagCOMPLEX
```

## Call Syntax :

```
z = CpxAdd(z1, z2)
z = CpxSub(z1, z2)
z = CpxMul(z1, z2)
z = CpxDiv(z1, z2)
z = CpxConjugue(z1)
m# = CpxModulus(z1)
a# = CpxArgument(z1)
z = CpxPowerN(z1, n%)
z = CpxSqrt(z1)
```

## Where :

z1	is a complex number (see tagCOMPLEX definition)
z2	is a complex number (see tagCOMPLEX definition)
z	is the result complex number (see tagCOMPLEX definition)
n%	is the power to be applied to a complex number
m#	is the modulus of a complex number
a#	is the argument of a complex number

## Comments :

## Examples :

```

Dim z          As tagCOMPLEX
Dim z1         As tagCOMPLEX
Dim z2         As tagCOMPLEX
Dim r          As Double

z1.a = 3: z1.b = 4
z2.a = 6: z2.b = 8

z = cCpxAdd(z1, z2)
Debug.Print "add : a="; z.a; " b="; z.b           ' add : a= 9  b= 12

z = cCpxSub(z1, z2)
Debug.Print "sub : a="; z.a; " b="; z.b           ' sub : a=-3  b=-4

z = cCpxMul(z1, z2)
Debug.Print "mul : a="; z.a; " b="; z.b           ' mul : a=-14  b= 48

z = cCpxDiv(z1, z2)
Debug.Print "div : a="; z.a; " b="; z.b           ' div : a= 0,5  b= 0

z = cCpxConjugue(z1)
Debug.Print "conj z1 : a="; z.a; " b="; z.b       ' conj z1 : a= 3  b=-4

z = cCpxConjugue(z2)
Debug.Print "conj z2 : a="; z.a; " b="; z.b       ' conj z2 : a= 6  b=-8

r = cCpxModulus(z1)
Debug.Print "modul z1 : r="; r                    ' modul z1 : r= 5

r = cCpxModulus(z2)
Debug.Print "modul z2 : r="; r                    ' modul z2 : r= 10

r = cCpxArgument(z1)
Debug.Print "arg z1 : phi="; r                    ' arg z1 : phi= 0,927295218001612

r = cCpxArgument(z2)
Debug.Print "arg z2 : phi="; r                    ' arg z2 : phi= 0,927295218001612

z = cCpxPowerN(z1, 3)
Debug.Print "Power3 z1 : a="; z.a; " b="; z.b     ' Power3 z1 : a=-117  b= 44

z = cCpxPowerN(z2, 3)
Debug.Print "Power3 z2 : a="; z.a; " b="; z.b     ' Power3 z2 : a=-936  b= 352

z = cCpxSqrt(z1)
Debug.Print "sqrt z1 : a="; z.a; " b="; z.b       ' sqrt z1 : a= 2  b= 1

z = cCpxSqrt(z2)
Debug.Print "sqrt z2 : a="; z.a; " b="; z.b       ' sqrt z2 : a= 2,82842712474619  b= 1,4142135623731

```

**See also :** [Math](#)



' constants for FileState (may be OR'ed) in cMNWalkInList

Public Const MNSTATE_READ = 0	'Read in, but not further processed
Public Const MNSTATE_MISPART = 1	'Missing Part(s) detected
Public Const MNSTATE_NOBEGIN = 2	'No 'begin' found
Public Const MNSTATE_NOEND = 4	'No 'end' found
Public Const MNSTATE_NODATA = 8	'File does not contain valid udata
Public Const MNSTATE_OK = 16	'All Parts found, ready to decode
Public Const MNSTATE_ERROR = 32	'Error while decoding
Public Const MNSTATE_DECODED = 64	'Successfully decoded
Public Const MNSTATE_TMPFILE = 128	'Temporary decoded file exists

' constants for OptionNO in cMNGetOption and cMNSetOption

Public Const MNOPT_VERSION = 0	'version number MAJOR.MINOR.PATCH (ro)
Public Const MNOPT_FAST = 1	'assumes only one part per file
Public Const MNOPT_DUMBNESS = 2	'switch off the program's intelligence
Public Const MNOPT_BRACKPOL = 3	'give numbers in [] higher precedence
Public Const MNOPT_VERBOSE = 4	'generate informative messages
Public Const MNOPT_DESPERATE = 5	'try to decode incomplete files
Public Const MNOPT_IGNREPLY = 6	'ignore RE:plies (off by default)
Public Const MNOPT_OVERWRITE = 7	'whether it's OK to overwrite ex. files
Public Const MNOPT_SAVEPATH = 8	'prefix to save-files on disk
Public Const MNOPT_IGNMODE = 9	'ignore the original file mode
Public Const MNOPT_DEBUG = 10	'print messages with FILE/LINE info
Public Const MNOPT_ERRNO = 14	'get last error code for MNRET_IOERR (ro)
Public Const MNOPT_PROGRESS = 15	'retrieve progress information
Public Const MNOPT_USETEXT = 16	'handle text messages
Public Const MNOPT_PREAMB = 17	'handle Mime preambles/epilogues
Public Const MNOPT_TINYB64 = 18	'detect short B64 outside of Mime
Public Const MNOPT_ENCEXT = 19	'extension for single-part encoded files

# Math : RootN

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

RootN computes the root N of a double number.

## Declare Syntax :

Declare Function cRootN Lib "time2win.dll" (ByVal x As Double, ByVal n As Integer, ByVal Precision As Integer, Iteration As Long) As Double

## Call Syntax :

v# = cRootN(x#, n%, Precision%, Iteration&)

## Where :

x#	is the double number to be proceeded
n%	is the root N to applied to the double number
Precision%	is the precision to be applied to find the root N of the double number (precision is computed by $10^{(-n)}$ )
Iteration&	is the returned number of iteration to satisfy the precision.
v#	is the returned root N of the double number.

## Comments :

If ( x# = 0.0 ) then the returned value is -1.  
If ( n% < 2 ) then the returned value is -2.  
if (Precision% < 3) then the returned value is -3.

## Examples :

```
Dim itr      As Long
Dim v        As Double
```

```
' computes the root 7 of the number with a precision of 10^(-10)
v = cRootN(1234567.98765432, 7, 10, itr)
Debug.Print x & " - " & itr & " - " & x ^ 7           ' 7,4167976458079 - 37 - 1234567,98765432
              (37 iterations)
```

```
' computes the root 27 of the number with a precision of 10^(-10)
v = cRootN(1234567.98765432, 27, 10, itr)
Debug.Print x & " - " & itr & " - " & x ^ 27         ' 1,68117013762404 - 176 - 1234567,98765432
              (176 iterations)
```

```
' computes the root 100 of the number with a precision of 10^(-20)
v = cRootN(1234567.98765432, 100, 20, itr)
Debug.Print x & " - " & itr & " - " & x ^ 100        ' 1,15057557433956 - 690 - 1234567,98765433
              (690 iterations)
```

See also : [Math](#)

# RmvDupD, RmvDupI, RmvDupL, RmvDupS

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**RmvDupD** will remove duplicate values in a **Double** array (one and two dimensions).

**RmvDupI** will remove duplicate values in an **Integer** array (one and two dimensions).

**RmvDupL** will remove duplicate values in a **Long** array (one and two dimensions).

**RmvDupS** will remove duplicate values in a **Single** array (one and two dimensions).

## Declare Syntax :

Declare Function cRmvDupD Lib "time2win.dll" (Darray() As Double, ByVal UseFileTemp As Integer, ByVal FileTemp As String, ByVal AutomaticResize As Integer) As Long

Declare Function cRmvDupI Lib "time2win.dll" (Iarray() As Integer, ByVal UseFileTemp As Integer, ByVal FileTemp As String, ByVal AutomaticResize As Integer) As Long

Declare Function cRmvDupL Lib "time2win.dll" (Larray() As Long, ByVal UseFileTemp As Integer, ByVal FileTemp As String, ByVal AutomaticResize As Integer) As Long

Declare Function cRmvDupS Lib "time2win.dll" (Sarray() As Single, ByVal UseFileTemp As Integer, ByVal FileTemp As String, ByVal AutomaticResize As Integer) As Long

## Call Syntax :

IngResult& = cRmvDupD(array(), UseFileTemp%, FileTemp\$, AutomaticResize%)

IngResult& = cRmvDupI(array(), UseFileTemp%, FileTemp\$, AutomaticResize%)

IngResult& = cRmvDupL(array(), UseFileTemp%, FileTemp\$, AutomaticResize%)

IngResult& = cRmvDupS(array(), UseFileTemp%, FileTemp\$, AutomaticResize%)

## Where :

array()	is the array (Double, Integer, Long, Single).
UseFileTemp%	True if you want to use a temporary file to perform the operation. False if you don't.
FileTemp\$	name of the temporary file (only if UseFileTemp% = True)
AutomaticResize%	True if you want to perform an automatic resize (same as Redim Preserve) False if you want to perform a "manual" resize (so you need to use Redim Preserve)
IngResult&	returnd the number of remaining element in the array for the last dimension.

## Comments :

WARNING, these routines need to sort the array before removing duplicate values.

If you want to remove duplicate values on a big array (more than 10.000 lines), you can use the temporary file. In this case, the usage of the temporary file is faster than the memory usage because, I don't need to move bigger portion of memory.

## Examples :

```
Dim IngResult As Long
ReDim TestArray(1 TO 5)As Integer
```

```
TestArray(1) = 1: TestArray(2) = 2: TestArray(3) = 3: TestArray(4) = 2: TestArray(5) = 1
```

```
IngResult = cRmvDupI(TestArray(), False, "", False) '-> value is 3
```

```
ReDim Preserve TestArray(1 To IngResult)
```

```
'----->
```

```
ReDim TestArray(1 To 3, 1 To 100000) As Integer
```

```
Dim i, j As Long  
Dim k As Integer
```

```
For j = 1 To 100000  
    k = (j Mod 1000)  
    TestArray(1, j) = k  
    TestArray(2, j) = k  
    TestArray(3, j) = k  
    If (j Mod 10000) = 0 Then Debug.Print i, j  
Next j
```

```
Call cStartBasisTimer
```

```
Debug.Print cRmvDupl(TestArray(), True, "c:\test\testarr.dat", False)
```

```
MsgBox "Elapsed time to remove :" & cReadBasisTimer()
```

**See Also :** [Array](#)

# TrayBar

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT4.0x}](#), [MSOffice 95](#)

## Purpose :

[TrayBar](#) hides/shows the tray bar.

## Declare Syntax :

```
Declare Sub cTrayBar Lib "time2win.dll" (ByVal HideOrShow As Integer)
```

## Call Syntax :

```
Call cTrayBar(HideOrShow%)
```

## Where :

HideOrShow%                    TRAYBAR\_HIDE to hide the tray bar;  
                                 TRAYBAR\_SHOW to show the tray bar.

## Comments :

## Examples :

```
Call cTrayBar(TRAYBAR_HIDE)    '-> so, tray bar is hidden
```

```
Call cTrayBar(TRAYBAR_SHOW)   '-> so, tray bar is showed
```

**See also :** [Windows 95](#)

# Extract

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**Extract** extracts a sub-string with a key in a string.

## Declare Syntax :

Declare Function cExtract Lib "time2win.dll" (ByVal Txt As String, ByVal Key As String, ByVal FieldSep As String, ByVal Sensitivity As Integer) As String

## Call Syntax :

sResult\$ = cExtract(Txt\$, Key\$, FieldSep\$, Sensitivity%)

## Where :

Txt\$	string expression being processed.
Key\$	string expression sought.
FieldSep\$	field separator (any number of chars).
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
sResult\$	the sub-string associated with the key.

## Comments :

If Txt\$ is zero-length, the returned string is empty.  
If Key\$ is zero-length, the returned string is empty.  
If FieldSep\$ is zero-length, the returned string is empty.

If FieldSep\$ can't be found in Txt\$, the returned string is the string from the key until the end of Txt\$.

The Key\$ is never returned in the returned string.  
The FieldSep\$ is never returned in the returned string.

## Examples :

Dim Str1 As String

```
Str1 = Str1 + "Path: ourworld.compuserve.com/homepages/alpouda" & vbCrLf
Str1 = Str1 + "From: time to win 32-bit" & vbCrLf
Str1 = Str1 + "Newsgroups: comp.lang.basic.visual.misc" & vbCrLf
Str1 = Str1 + "Subject: extract a sub-string with a key" & vbCrLf
Str1 = Str1 + "Date: 17 Aug 1997 16:25:43 GMT" & vbCrLf
Str1 = Str1 + "Organization: The M.C.R. Company" & vbCrLf
Str1 = Str1 + "Lines: 25" & vbCrLf
Str1 = Str1 + "Message-ID: <5t5u41$8ai$11@ourworld.compuserve.com>" & vbCrLf
Str1 = Str1 + "NNTP-Posting-Host: ppp054.103.ourworld.compuserve.com" & vbCrLf
Str1 = Str1 + "Post-Count: 000518" & vbCrLf
Str1 = Str1 + "Xref: time to win 32-bit (v" & cGetVersion() & ")" & vbCrLf
```

Debug.Print cExtract(Str1, "Path:", vbCrLf, True)	'-> ourworld.compuserve.com/homepages/alpouda
Debug.Print cExtract(Str1, "Subject:", vbCrLf, True)	'-> extract a sub-string with a key
Debug.Print cExtract(Str1, "Newsgroups:", vbCrLf, True)	'-> comp.lang.basic.visual.misc
Debug.Print cExtract(Str1, "from:", vbCrLf, True)	'-> ""
Debug.Print cExtract(Str1, "from:", vbCrLf, False)	'-> time to win 32-bit

```
Debug.Print cExtract(Str1, "from:", "", True)      '-> ""  
Debug.Print cExtract("", "from:", vbCrLf, True)  '-> ""  
Debug.Print cExtract(Str1, "", vbCrLf, True)     '-> ""
```

**See also :** [String](#)



# ExtractIsolate

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[ExtractIsolate](#) extracts a left/right part of a string from a key and a field separator.

## Declare Syntax :

Declare Function cExtractIsolate Lib "time2win.dll" (ByVal Txt As String, ByVal Key As String, ByVal FieldSep As String, ByVal LeftOrRight As Integer, ByVal Sensitivity As Integer) As String

## Call Syntax :

sResult\$ = cExtract(Txt\$, Key\$, FieldSep\$, LeftOrRight%, Sensitivity%)

## Where :

Txt\$	string expression being processed.
Key\$	string expression sought.
FieldSep\$	field separator (any number of chars).
LeftOrRight%	EXTRACT_ISOLATE_LEFT isolates the left part EXTRACT_ISOLATE_RIGHT isolates the right part
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
sResult\$	the string associated with the key and the field separator.

## Comments :

If Txt\$ is zero-length, the returned string is empty.  
If FieldSep\$ is zero-length, the returned string is empty.

The Key\$ is never returned in the returned string.  
The FieldSep\$ is never returned in the returned string.

This function can be usefull to separate the header and the body of an email or news message (set Key\$ = "").

## Examples :

```
Dim Str1 As String
```

```
Str1 = Str1 + "Path: ourworld.compuserve.com/homepages/alpouda" & vbCrLf  
Str1 = Str1 + "From: time to win 32-bit" & vbCrLf  
Str1 = Str1 + "Newsgroups: comp.lang.basic.visual.misc" & vbCrLf  
Str1 = Str1 + "" & vbCrLf  
Str1 = Str1 + "Body Part" & vbCrLf
```

```
Debug.Print cExtractIsolate(Str1, "Path:", vbCrLf, EXTRACT_ISOLATE_LEFT, True)
```

```
'->
```

```
Debug.Print cExtractIsolate(Str1, "Path:", vbCrLf, EXTRACT_ISOLATE_RIGHT, True)
```

```
'-> ourworld.compuserve.com/homepages/alpouda  
'-> From: time to win 32-bit  
'-> Newsgroups: comp.lang.basic.visual.misc  
'->  
'-> Body Part
```

```
Debug.Print cExtractIsolate(Str1, "Newsgroups:", vbCrLf, EXTRACT_ISOLATE_LEFT, True)
```

```
'-> Path: ourworld.compuserve.com/homepages/alpouda  
'-> From: time to win 32-bit
```

```
Debug.Print cExtractIsolate(Str1, "Newsgroups:", vbCrLf, EXTRACT_ISOLATE_RIGHT, True)
```

```
'-> comp.lang.basic.visual.misc  
'->  
'-> Body Part
```

```
Debug.Print cExtractIsolate(Str1, "", vbCrLf, EXTRACT_ISOLATE_LEFT, True)
```

```
'-> Path: ourworld.compuserve.com/homepages/alpouda  
'-> From: time to win 32-bit  
'-> Newsgroups: comp.lang.basic.visual.misc
```

```
Debug.Print cExtractIsolate(Str1, "", vbCrLf, EXTRACT_ISOLATE_RIGHT, True)
```

```
'-> Body Part
```

**See also :** [String](#)

# GetOSx

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetOSBuildNumber](#) identifies the build number of the operating system.

[GetOSCSVersion](#) provides arbitrary additional information about the operating system.

[GetOSMajorVersion](#) identifies the major version number of the operating system.

[GetOSMinorVersion](#) identifies the minor version number of the operating system.

[GetOSVersion](#) identifies the version number of the operating system (major.minor version).

[GetOSPlatformId](#) identifies the operating system platform in a numerical format.

[GetOSPlatformName](#) identifies the operating system platform in a readable format (string).

## Declare Syntax :

```
Declare Function cGetOSBuildNumber Lib "time2win.dll" () As Long
Declare Function cGetOSCSVersion Lib "time2win.dll" () As String
Declare Function cGetOSMajorVersion Lib "time2win.dll" () As Long
Declare Function cGetOSMinorVersion Lib "time2win.dll" () As Long
Declare Function cGetOSVersion Lib "time2win.dll" () As String
Declare Function cGetOSPlatformId Lib "time2win.dll" () As Long
Declare Function cGetOSPlatformName Lib "time2win.dll" () As String
```

## Call Syntax :

```
IResult& = cGetOSBuildNumber()
sResult$ = cGetOSCSVersion()
IResult& = cGetOSMajorVersion()
IResult& = cGetOSMinorVersion()
sResult& = cGetOSVersion()
IResult& = cGetOSPlatformId()
sResult$ = cGetOSPlatformName()
```

## Where :

## Comments :

## Examples :

```
Dim strDisplay As String
```

```
strDisplay = ""
```

```
strDisplay = strDisplay + "OSBuildNumber : " & cGetOSBuildNumber() & " (" & (cGetOSBuildNumber() And &HFFFF&) & "-" & ((cGetOSBuildNumber() And &HFFFF0000) / 65536) & ")" & vbCrLf & vbCrLf
strDisplay = strDisplay + "OSCSVersion : " & cGetOSCSVersion() & vbCrLf & vbCrLf
strDisplay = strDisplay + "OSMajorVersion : " & cGetOSMajorVersion() & vbCrLf & vbCrLf
strDisplay = strDisplay + "OSMinorVersion : " & cGetOSMinorVersion() & vbCrLf & vbCrLf
strDisplay = strDisplay + "OSVersion : " & cGetOSVersion() & vbCrLf & vbCrLf
strDisplay = strDisplay + "OSPlatformId : " & cGetOSPlatformId() & vbCrLf & vbCrLf
strDisplay = strDisplay + "OSPlatformName : " & cGetOSPlatformName() & vbCrLf & vbCrLf
```

```
Debug.Print strDisplay
```

' On my system :

' OSBuildNumber : 67109814 (950:1024)  
' OSCSDVersion :  
' OSMajorVersion : 4  
' OSMinorVersion : 0  
' OSVersion : 4.0  
' OSPlatformId : 1  
' OSPlatformName : Windows 95

**See also** : [Windows 95](#)

# GetBitX2

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetBitB2](#) return if a given bit (real bit position) in a given BYTE is Set or Reset.

[GetBitI2](#) return if a given bit (real bit position) in a given INTEGER is Set or Reset.

[GetBitL2](#) return if a given bit (real bit position) in a given LONG is Set or Reset.

## Declare Syntax :

Declare Function cGetBitB2 Lib "[time2win.dll](#)" (ByVal Value As Byte, ByVal Position As Integer) As Integer

Declare Function cGetBitI2 Lib "[time2win.dll](#)" (ByVal Value As Integer, ByVal Position As Integer) As Integer

Declare Function cGetBitL2 Lib "[time2win.dll](#)" (ByVal Value As Long, ByVal Position As Integer) As Integer

## Call Syntax :

test% = cGetBitB2(Value, Position)

test% = cGetBitI2(Value, Position)

test% = cGetBitL2(Value, Position)

## Where :

Value	the value to proceed
Position	the bit position
test	TRUE if the bit is Set FALSE if the bit is Reset

## Comments :

The first bit in the value is the bit 0.

**See also :** [Binary](#)

# SetBitX2

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[SetBitB2](#) set a given bit (real bit position) in a given BYTE to Set state or Reset state.

[SetBitI2](#) set a given bit (real bit position) in a given INTEGER to Set state or Reset state.

[SetBitL2](#) set a given bit (real bit position) in a given LONG to Set state or Reset state.

## Declare Syntax :

Declare Sub cSetBitB2 Lib "time2win.dll" (Value As Byte, ByVal Position As Integer, ByVal BitValue As Integer)

Declare Sub cSetBitI2 Lib "time2win.dll" (Value As Integer, ByVal Position As Integer, ByVal BitValue As Integer)

Declare Sub cSetBitL2 Lib "time2win.dll" (Value As Long, ByVal Position As Integer, ByVal BitValue As Integer)

## Call Syntax :

Call cSetBitB2(Value, Position, BitValue)

Call cSetBitI2(Value, Position, BitValue)

Call cSetBitL2(Value, Position, BitValue)

## Where :

Value	the value to proceed
Position	the bit position
BitValue	TRUE to Set the bit FALSE to Reset the bit

## Comments :

The first bit in the string is the bit 0.

**See also :** [Binary](#)

# GetNetUser

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[GetNetUser](#) retrieves the current default user name or the user name used to establish a network connection.

## Declare Syntax :

Declare Function cGetNetUser Lib "[time2win.dll](#)" (ErrCode As Integer) As String

## Call Syntax :

```
test$ = cGetNetConnection(ErrCode)
```

## Where :

ErrCode	TRUE is all is ok <> TRUE if an error has occurred
test\$	the returned name of the current default user name

## Comments :

## Examples :

```
Debug.Print cGetNetUser(ErrCode%) ' TIME2WIN
```

**See also :** [Network](#)

# Gradient

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[Gradient](#) can create a gradient color in 8 effects.

## Declare Syntax :

Declare Sub cGradient Lib "time2win.dll" (ByVal hDC As Long, ByVal iLeft As Integer, ByVal iTop As Integer, ByVal iWidth As Integer, ByVal iHeight As Integer, ByVal iGradientStyle As Integer, ByVal iGradientGranularity As Integer, ByVal IGradientColor As Long)

## Call Syntax :

Call cGradient(hDC&, iLeft%, iTop%, iWidth%, iHeight%, iGradientStyle%, iGradientGranularity%, IGradientColor&)

## Where :

hDC&	is the hDC handle of the object (normally Form and Picture)
iLeft%	is the left position to start the gradient (normally 0)
iTop%	is the top position to start the gradient (normally 0)
iWidth%	is the width size in PIXEL of the object (use .ScaleMode = 3)
iHeight%	is the height size in PIXEL of the object (use .ScaleMode = 3)
iGradientStyle%	is the gradient style (see constant <a href="#">GradientStyle</a> )
iGradientGranularity%	is the gradient granularity (see constant <a href="#">GradientGranularity</a> )
IGradientColor&	is the gradient color to start

## Comments :

## Examples :

Call cGradient(Me.hDC, 0, 0, Me.ScaleWidth, Me.ScaleHeight, mcGradientStyleEnum.mcOuterVGradient, mcGradientGranularity.mcHigh, vbWhite)

**See also :** [Windows](#)



# GetNetAdapterMacAddress

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetNetAdapterMacAddress](#) retrieves the encoded address of the network adapter (MAC address).

## Declare Syntax :

Declare Function cGetNetAdapterMacAddress Lib "t2win-32.dll" (ByVal iLanAdapter As Integer, ByVal iFormat As Integer, iErrorCode As Integer) As String

## Call Syntax :

test\$ = cGetNetAdapterMacAddress(iLanAdapter%, iFormat%, iErrorCode%)

## Where :

iLanAdapter%	is the number of the adapter (physical address) to use to find the MAC address
iFormat%	is the format to receive (1 = aabbccddeeff, 2 = aa-bb-cc-dd-ee-ff)
iErrorCode%	>0 if error (see <a href="#">Error Code for Adapter</a> ) -1 if no error
test\$	the encoded address of the network adapter.

## Comments :

## Examples :

```
Debug.Print cGetNetAdapterMacAddress(1, 1, iErrorCode%) ' 006097a8f11e
Debug.Print cGetNetAdapterMacAddress(1, 2, iErrorCode%) ' 00-60-97-a8-f1-1e

Debug.Print cGetNetAdapterMacAddress(0, 2, iErrorCode%) ' iErrorCode = 4 (ADAPTER_BAD_NUMBER)
```

**See also** : [Network](#)

# GetNetNumberOfAdapter

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetNetNumberOfAdapter](#) retrieves the number of LAN adapter (number of network card).

## Declare Syntax :

```
Declare Function cGetNetNumberOfAdapter Lib "t2win-32.dll" (iErrorCode As Integer) As Integer
```

## Call Syntax :

```
test% = cGetNetAdapterMacAddress(iErrorCode%)
```

## Where :

iErrorCode%	>0 if error (see <a href="#">Error Code for Adapter</a> ) -1 if no error
test%	the number of LAN adapter (number of network card).

## Comments :

## Examples :

```
Debug.Print cGetNetNumberOfAdapter(iErrorCode%) ' 1
```

**See also :** [Network](#)

' error code for adapter

```
Public Const ADAPTER_NO_ERROR = -1
Public Const ADAPTER_CANT_LOAD_APIDLL = 1
Public Const ADAPTER_CANT_FIND_ENTRYPOINT = 2
Public Const ADAPTER_CANT_BE_ENUMERATED = 3
Public Const ADAPTER_BAD_NUMBER = 4
Public Const ADAPTER_CANT_BE_RESETEED = 5
Public Const ADAPTER_CANT_BE_READ = 6
```

# GetNetAdapterNumber

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetNetAdapterNumber](#) retrieves the logical address of a physical LAN adapter.

## Declare Syntax :

Declare Function cGetNetAdapterNumber Lib "t2win-32.dll" (ByVal iLanAdapter As Integer, iErrorCode As Integer) As Integer

## Call Syntax :

test% = cGetNetAdapterNumber(iLanAdapter%, iErrorCode%)

## Where :

iLanAdapter%	is the number of the adapter to use to find the logical address
iErrorCode%	>0 if error (see <a href="#">Error Code for Adapter</a> )
	-1 if no error
test%	the logical address.

## Comments :

## Examples :

```
Debug.Print cGetNetAdapterNumber(1, iErrorCode%)' 7
```

**See also :** [Network](#)

# FileLineCount2

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileLineCount2](#) counts the total number of lines in an ASCII file.

## Declare Syntax :

Declare Function cFileLineCount2 Lib "[time2win.dll](#)" (ByVal lpFilename As String) As Long

## Call Syntax :

```
test& = cFileLineCount2(lpFilename$)
```

## Where :

lpFilename\$	is the name of the file.
test&	is the total number of lines.

## Comments :

Each line (*maximum 32768 chars*) is determined only if a CR is ending the line.

The returned value can be negative and have the following value :

- 1 error opening file (not exist, not a valid filename).
- 2 error reading file.
- 3 error when allocating memory buffer.

## Examples :

```
test& = cFileLineCount("c:\autoexec.bat")
```

On my system :

```
test& = 31
```

**See also :** [File](#), [FileLineCount](#)

# GetCVx

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GetCVB](#), [GetCVC](#), [GetCVD](#), [GetCVI](#), [GetCVL](#) and [GetCVS](#) returns number in a certain precision given a string containing the IEEE representation of the number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG and SINGLE. These functions is a fully replacement of the cCVx(MID\$(string, offset, length)).

## Declare Syntax :

```
Declare Function cGetCVB Lib "time2win.dll" (Value As String, ByVal iPosition As Integer) As Integer
Declare Function cGetCVC Lib "time2win.dll" (Value As String, ByVal iPosition As Integer) As Currency
Declare Function cGetCVD Lib "time2win.dll" (Value As String, ByVal iPosition As Integer) As Double
Declare Function cGetCVI Lib "time2win.dll" (Value As String, ByVal iPosition As Integer) As Integer
Declare Function cGetCVL Lib "time2win.dll" (Value As String, ByVal iPosition As Integer) As Long
Declare Function cGetCVS Lib "time2win.dll" (Value As String, ByVal iPosition As Integer) As Single
```

## Call Syntax :

test% = cGetCVB(Value\$, iPosition%)	'equivalent to cCVB(Value\$, 1 + ((iPosition% - 1) * 1), 1)
test@ = cGetCVC(Value\$, iPosition%)	'equivalent to cCVC(Value\$, 1 + ((iPosition% - 1) * 8), 8)
test# = cGetCVD(Value\$, iPosition%)	'equivalent to cCVD(Value\$, 1 + ((iPosition% - 1) * 8), 8)
test% = cGetCVI(Value\$, iPosition%)	'equivalent to cCVI(Value\$, 1 + ((iPosition% - 1) * 2), 2)
test& = cGetCVL(Value\$, iPosition%)	'equivalent to cCVL(Value\$, 1 + ((iPosition% - 1) * 4), 4)
test! = cGetCVS(Value\$, iPosition%)	'equivalent to cCVS(Value\$, 1 + ((iPosition% - 1) * 4), 4)

## Where :

Value\$	is the string which held IEEE string
iPosition%	is the position of the IEEE string in Value\$
test?	receives the value represented by the IEEE string held in Value\$

## Comments :

**See also :** [PutMKx](#)

# PutMKx

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[PutMKB](#), [PutMKC](#), [PutMKD](#), [PutMKI](#), [PutMKL](#), and [PutMKS](#) returns a string containing the IEEE representation of a number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG, SINGLE. These functions are a fully replacement of the `Mid$(string, offset, length) = Value`.

[PutMKBs](#), [PutMKCs](#), [PutMKDs](#), [PutMKIs](#), [PutMKLs](#), and [PutMKSS](#) returns a string containing the IEEE representation of a number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG, SINGLE. These functions are the same as above but must be used for fields in a recordset.

## Declare Syntax :

```
Declare Sub cPutMKB Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal bValue As Byte)
Declare Sub cPutMKC Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal cValue As Currency)
Declare Sub cPutMKD Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal dValue As Double)
Declare Sub cPutMKI Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal iValue As Integer)
Declare Sub cPutMKL Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal lValue As Long)
Declare Sub cPutMKS Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal sValue As Single)
```

```
Declare Function cPutMKBs Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal bValue As Byte) As String
Declare Function cPutMKCs Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal cValue As Currency) As String
Declare Function cPutMKDs Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal dValue As Double) As String
Declare Function cPutMKIs Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal iValue As Integer) As String
Declare Function cPutMKLs Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal lValue As Long) As String
Declare Function cPutMKSS Lib "time2win.dll" (Value As String, ByVal iPosition As Integer, ByVal sValue As Single) As String
```

## Call Syntax :

Call cPutMKB(Value\$, iPosition%, bValue)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 1), 1) = bValue
Call cPutMKC(Value\$, iPosition%, cValue@)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 8), 8) = cValue@
Call cPutMKD(Value\$, iPosition%, sValue#)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 8), 8) = sValue#
Call cPutMKI(Value\$, iPosition%, iValue%)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 2), 2) = iValue%
Call cPutMKL(Value\$, iPosition%, lValue&)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 4), 4) = lValue&
Call cPutMKS(Value\$, iPosition%, sValue!)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 4), 4) = sValue!
rs("Field1") = cPutMKBs(rs("Field1"), iPosition%, bValue)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 1), 1) = bValue
rs("Field1") = cPutMKCs(rs("Field1"), iPosition%, cValue@)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 8), 8) = cValue@
rs("Field1") = cPutMKDs(rs("Field1"), iPosition%, sValue#)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 8), 8) = sValue#
rs("Field1") = cPutMKIs(rs("Field1"), iPosition%, iValue%)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 2), 2) = iValue%
rs("Field1") = cPutMKLs(rs("Field1"), iPosition%, lValue&)	'equivalent to Mid\$(Value\$, 1 + ((iPosition% - 1) * 4), 4) = lValue&

```
1) * 4), 4) = lValue&  
rs("Field1") = cPutMKSS(rs("Field1"), iPosition%, sValue!)  
1) * 4), 4) = sValue!
```

'equivalent to Mid\$(Value\$, 1 + ((iPosition% -

**Where :**

Value\$	receives the modification.
rs("Field1")	is a field in a recordset
iPosition%	is the position of the IEEE string in Value\$
xValue	the new value

**Comments :**

**See also :** [GetCVx](#)



```
Public Enum mcGradientStyleEnum
    mcNoneGradient = 0
    mcLeftHGradient = 1
    mcRightHGradient = 2
    mcTopVGradient = 3
    mcBottomVGradient = 4
    mcInnerHGradient = 5
    mcOuterHGradient = 6
    mcInnerVGradient = 7
    mcOuterVGradient = 8
End Enum
```

# IsCapsLockOn, IsNumLockOn, IsScrollLockOn

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[IsCapsLockOn](#) verifies if the CAPS LOCK key is On.

[IsNumLockOn](#) verifies if the NUM LOCK key is On.

[IsInsertOn](#) verifies if the INSERT key is On.

[IsScrollLockOn](#) verifies if the SCROLL LOCK key is On.

## **Declare Syntax :**

```
Declare Function clsCapsLockOn Lib "time2win.dll" () As Integer
```

```
Declare Function clsInsertOn Lib "time2win.dll" () As Integer
```

```
Declare Function clsNumLockOn Lib "time2win.dll" () As Integer
```

```
Declare Function clsScrollLockOn Lib "time2win.dll" () As Integer
```

## **Call Syntax :**

```
test% = clsCapsLockOn()
```

```
test% = clsInsertOn()
```

```
test% = clsNumLockOn()
```

```
test% = clsScrollLockOn()
```

## **Where :**

test%	TRUE if the key is On
	FALSE if the key is Off

## **Comments :**

## **Examples :**

**See also :** [Windows 95](#)

# FilePartAppend, FilePartCopy

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FilePartAppend](#) appends part of one file to another file starting at or until Offset.

[FilePartCopy](#) copies part of one file to another file starting at or until Offset.

## Declare Syntax :

```
Declare Function cFilePartAppend Lib "time2win.dll" (ByVal SourceFile As String, ByVal TargetFile As String, ByVal Offset As Long, ByVal FirstPart As Integer) As Long
```

```
Declare Function cFilePartCopy Lib "time2win.dll" (ByVal SourceFile As String, ByVal TargetFile As String, ByVal Offset As Long, ByVal FirstPart As Integer) As Long
```

## Call Syntax :

```
test& = cFilePartAppend(SourceFile$, TargetFile$, Offset&, FirstPart%)
```

```
test& = cFilePartCopy(SourceFile$, TargetFile$, Offset&, FirstPart%)
```

## Where :

SourceFiles\$	is the source file.
TargetFile\$	is the target filename to append/copy.
Offset&	is the position in the source file.
FirstPart%	is the part of the source file : to Offset or from Offset.
test&	> 0 if all is OK (the returned value is the size of the part copied), < 0 if an error has occurred.

## Comments :

The returned value can be negative and have the following value :

```
Public Const COPY_BAD_POSITION = -1
Public Const COPY_BAD_SOURCE_FILENAME = -2
Public Const COPY_BAD_TARGET_FILENAME = -3
Public Const COPY_POSITION_TOO_BIG = -4
Public Const COPY_CANT_OPEN_SOURCE = -5
Public Const COPY_CANT_CREATE_TARGET = -6
```

-32720 the number of chars in a block for writing differs from the number of chars for reading.

-32730 reading error for SourceFile.

-32740 writing error for TargetFiles.

-32760 allocation error for memory buffer.

## Examples :

```
Debug.Print cFileSize("d:\win95\system\vba2.dll")
```

```
Debug.Print cFilePartCopy("d:\win95\system\vba2.dll", "d:\win95\temp\vbleft.hlp", 1234567, True)
Debug.Print cFilePartCopy("d:\win95\system\vba2.dll", "d:\win95\temp\vbright.hlp", 1234567, False)
```

```
Debug.Print cFilePartAppend("d:\win95\system\vba2.dll", "d:\win95\temp\vbleft.hlp", 1234567, True)
Debug.Print cFilePartAppend("d:\win95\system\vba2.dll", "d:\win95\temp\vbright.hlp", 1234567, False)
```

```
Debug.Print cFileSize("d:\win95\temp\vbleft.hlp")
Debug.Print cFileSize("d:\win95\temp\vbright.hlp")
```

Result is :

1378560

1234567  
143993

1234567  
143993

2469134  
287986

**See also :** [File](#)

# IsWin95OSR2

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[IsWin95OSR2](#) determines if the current version of Win95 is the OSR2 or not.

## **Declare Syntax :**

Declare Function clsWin95OSR2 Lib "[time2win.dll](#)" () As Integer

## **Call Syntax :**

test% = clsWin95OSR2()

## **Where :**

test%	TRUE if it's Win95OSR2
	FALSE if not

## **Comments :**

## **Examples :**

**See also :** [Windows 95](#)

RoundNearest

# Round

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[Round](#) rounds a number with a precision.

[RoundNearest](#) rounds a number to the nearest value.

## Declare Syntax :

Declare Function cRound Lib "[time2win.dll](#)" (ByVal x As Double, ByVal Precision As Integer) As Double

Declare Function cRoundNearest Lib "[time2win.dll](#)" (ByVal x As Double, ByVal NearestValue As Long) As Double

## Call Syntax :

test# = cRound(x#, Precision%)

test# = cRoundNearest(x#, NearestValue&)

## Where :

test#	the rounded value
x#	the value to be rounded
Precision%	the number of decimal for the precision
NearestValue&	the nearest value

## Comments :

## Examples :

Debug.Print cRound(3.56376, 3)	'3,564
Debug.Print cRound(3.56376, 1)	'3,6
Debug.Print cRound(3.56376, 0)	'4
Debug.Print cRound(3.56376, 2)	'3,56
Debug.Print cRound(1.4999, 3)	'1,5
Debug.Print cRound(1.4899, 2)	'1,49
Debug.Print cRoundNearest(12.4, 5)	'10
Debug.Print cRoundNearest(12.5, 5)	'15
Debug.Print cRoundNearest(12.6, 5)	'15
Debug.Print cRoundNearest(18.1, 5)	'20
Debug.Print cRoundNearest(0.1, 5)	'0
Debug.Print cRoundNearest(2.6, 5)	'5

**See also :** [Math](#)

# FileScanHeader, FileScanHeaderForRecipients

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FileScanHeader](#) searches a string in a given HEADER file starting at a certain line number.

[FileScanHeaderForRecipients](#) searches a string (recipients like "to: ", "cc: ", "bcc: ") in a given HEADER file starting at a certain line number.

## Declare Syntax :

Declare Function cFileScanHeader Lib "time2win.dll" (ByVal nFileName As String, ByVal Search As String, ByVal FieldSep As String, StartLine As Long, ByVal Sensitivity As Integer) As String

Declare Function cFileScanHeaderForRecipients Lib "time2win.dll" (ByVal nFileName As String, ByVal Search As String, ByVal FieldSep As String, ByVal ExtractOnlyAddress As Integer, ByVal ExtractAddressSep As String, StartLine As Long, ByVal Sensitivity As Integer) As String

## Call Syntax :

```
test$ = cFileScanHeader(nFilename$, Search$, FieldSep$, StartLine&, Sensitivity%)
```

```
test$ = cFileScanHeaderForRecipients(nFileName$, Search$, FieldSep$, ExtractOnlyAddress%,  
ExtractAddressSep$, Sensitivity%)
```

## Where :

nFilename\$	the ASCII file.
Search\$	the string to be searched.
FieldSep\$	field separator (any number of chars).
ExtractOnlyAddress%	True if the e-mail address must be only returned False if the e-mail address and the name must be returned
ExtractAddressSep\$	If empty string the following separators are used "<>" & Chr\$(34) & Chr\$(34) & vbTab if not empty string : 1. the first two characters are the delimiters for the e-mail address; 2. the next two characters are the delimiters for the alias name; 3. the fifth character are the separator between multi e-mail address.
StartLine&	the line number to start the search.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
test\$	<> "" if success; = "" if an error occurs.

## Comments :

cFileScanHeader and cFileScanHeaderForRecipients can handle lines with a maximum of 4096 chars.

If the nFilename string is an EMPTY string, the returned value is "".

If the Search string is an EMPTY string, the returned value is "".

The StartLine can be negative (if an error occurs) and have the following value :

- 1 the string to be searched can't be found.
- 32730 reading error for file 1.
- 32750 opening error for file 1.

## Examples :

File "Message Subject.eml" :

```
From: "Alpha Beta" <alpha@beta.edu>  
To: "Gamma Theta" <gamma@beta.edu>,
```



<epsilon@beta.edu>,  
"rho@beta.edu",  
Subject: Message Subject  
Date: Sun, 29 Mar 1998 22:34:04 +0200  
MIME-Version: 1.0

StartLine = 0  
Debug.Print cFileScanHeaderForRecipients("c:\cserve\cs3\download\Message Subject.eml", "to: ", "", False, "",  
StartLine, False)  
"Gamma Theta" <gamma@beta.edu> <epsilon@beta.edu> "rho@beta.edu"

StartLine = 0  
Debug.Print cFileScanHeaderForRecipients("c:\cserve\cs3\download\Message Subject.eml", "to: ", "", True, "",  
StartLine, False)  
gamma@beta.edu epsilon@beta.edu rho@beta.edu

StartLine = 0  
Debug.Print cFileScanHeaderForRecipients("c:\cserve\cs3\download\Message Subject.eml", "to: ", "", True, "<>"  
& Chr\$(34) & Chr\$(34) & vbCr, StartLine, False)  
gamma@beta.edu  
epsilon@beta.edu  
rho@beta.edu

**See also :** [File](#)

# IsPrime

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

These routines checks if the specified value is :

[IsPrime](#)            prime

## Declare Syntax :

Declare Function clsPrime Lib "t2win-32.dll" (ByVal Value As Long) As Integer

## Call Syntax :

```
test = clsPrime(Value&)
```

## Where :

Value&	the value to proceed
test	TRUE if test is OK FALSE if the test fails

## Comments :

## Examples :

```
Value& = 1234
```

```
test = clsPrime(Value&)                            ' False
```

```
Value& = 1231
```

```
test = clsPrime(Value&)                            ' True
```

**See also :** [Is](#)

# PrinterHeight, PrinterOffsetLeft, PrinterOffsetTop, PrinterWidth

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[PrinterHeight](#) returns the height of the printer in inch.

[PrinterOffsetLeft](#) returns the left offset of the printer in inch (begin of the printable area).

[PrinterOffsetTop](#) returns the top offset of the printer in inch (begin of the printable area).

[PrinterWidth](#) returns the height of the printer in inch.

## Declare Syntax :

```
Declare Function cPrinterWidth Lib "time2win.dll" (ByVal hDC As Long) As Single
```

```
Declare Function cPrinterHeight Lib "time2win.dll" (ByVal hDC As Long) As Single
```

```
Declare Function cPrinterOffsetLeft Lib "time2win.dll" (ByVal hDC As Long) As Single
```

```
Declare Function cPrinterOffsetTop Lib "time2win.dll" (ByVal hDC As Long) As Single
```

## Call Syntax :

```
sngResult! = cPrinterWidth(hDC&)
```

```
sngResult! = cPrinterHeight(hDC&)
```

```
sngResult! = cPrinterOffsetLeft(hDC&)
```

```
sngResult! = cPrinterOffsetTop(hDC&)
```

## Where :

hDC& is the .hDC property of the printer object (Printer.hDC);

sngResult! is the returned value.

## Comments :

## Examples :

```
Dim strDisplay As String
```

```
strDisplay = ""
```

```
strDisplay = strDisplay + "Printer width : " & cPrinterWidth(Printer.hDC) & vbCrLf
```

```
strDisplay = strDisplay + "Printer height : " & cPrinterHeight(Printer.hDC) & vbCrLf
```

```
strDisplay = strDisplay + "Printer offset left : " & cPrinterOffsetLeft(Printer.hDC) & vbCrLf
```

```
strDisplay = strDisplay + "Printer offset top : " & cPrinterOffsetTop(Printer.hDC) & vbCrLf
```

```
Debug.Print strDisplay
```

```
' Printer width : 8,27
```

```
' Printer height : 11,69
```

```
' Printer offset left : 0,2166667
```

```
' Printer offset top : 0,1666667
```

## See also :

# SearchStr

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SearchStr](#) searches a [String](#) array for a given string.

## Declare Syntax :

Declare Function cSearchStr Lib "[time2win.dll](#)" (Strarray() As String, ByVal Value As String, ByVal Sensitivity As Integer) As Long

## Call Syntax :

cnt& = cSearchStr(Strarray(), Value\$, Sensitivity%)

## Where :

Strarray()	is the string array.
Value\$	is the string to search.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
cnt&	> 0 : the position of the searched value; = -1 : the searched value is not found.

## Comments :

## Examples :

**See Also :** [Array](#)

# FileSearchFromFile

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[FileSearchFromLine](#) searches a string in a given TEXT file starting at a certain line number.

[FileSearchPatternFromLine](#) searches a pattern string in a given TEXT file starting at a certain line number.

## Declare Syntax :

Declare Function cFileSearchFromLine Lib "[time2win.dll](#)" (ByVal nFileName As String, ByVal Search As String, StartLine As Long, ByVal Sensitivity As Integer) As String

Declare Function cFileSearchPatternFromLine Lib "[time2win.dll](#)" (ByVal nFileName As String, ByVal Search As String, StartLine As Long, ByVal Sensitivity As Integer) As String

## Call Syntax :

```
test$ = cFileSearchFromLine(nFilename$, Search$, StartLine&, Sensitivity%)
```

```
test$ = cFileSearchPatternFromLine(nFilename$, Search$, StartLine&, Sensitivity%)
```

## Where :

nFilename\$	the ASCII file.
Search\$	the string or the pattern string to be searched
StartLine&	the line number to start the search.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
test\$	<> "" if success; = "" if an error occurs.

## Comments :

cFileSearchFromLine can handle lines with a maximum of 4096 chars.

cFileSearchPatternFromLine can handle lines with a maximum of 4096 chars.

If the nFilename string is an EMPTY string, the returned value is "".

If the search string is an EMPTY string, the returned value is "".

The StartLine can be negative (if an error occurs) and have the following value :

-1	the string to be searched can't be found.
-32730	reading error for file 1.
-32750	opening error for file 1.

For cFileSearchPatternFromLine : see [PatternExtMatch](#)

## Examples :

```
Dim StartLine      As Long
Dim strResult      As String
Dim strDisplay     As String
Dim File1          As String
Dim Search1        As String
```

```
File1 = "c:\autoexec.bat"
Search1 = "rem"
```

```
strDisplay = strDisplay & "File Search (all lines occurrence, sensitive) : " & Search1 & " in " & File1 & " is " &
vbCrLf
```

```
StartLine = 0
```

```
strResult = cFileSearchFromLine(File1, Search1, StartLine, True)
While (StartLine > 0)
    strDisplay = strDisplay & "    line (" & StartLine & ") : " & strResult & vbCrLf
    strResult = cFileSearchFromLine(File1, Search1, StartLine, True)
Wend
strDisplay = strDisplay & vbCrLf

Debug.Print strDisplay
```

On my system :

```
line (9) : rem - By Windows 95 Network - D:\WIN95\net start
line (15) : rem - By Windows Setup - LH /L:1,40352 D:\WIN95\COMMAND\MSCDEX.EXE /D:ASPICD0 /M:12 /L:R
```

**See also :** [File](#)

# GreatCommonDivisor

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[GreatCommonDivisor](#) returns the Greatest Common Divisor of two numbers.

## Declare Syntax :

Declare Function cGreatCommonDivisor Lib "t2win-32.dll" (ByVal Value1 As Long, ByVal Value2 As Long) As Long

## Call Syntax :

test& = cGreatCommonDivisor(Value1&, Value2&)

## Where :

Value1&	is the first number to be proceeded
Value2&	is the second number to be proceeded
test&	is the great common divisor

## Comments :

## Examples :

Debug.Print cGreatCommonDivisor(15, 9)	'-> 3
Debug.Print cGreatCommonDivisor(123698745, 147896325)	'-> 45

**See also** : [Math](#)

# Rotate

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[Rotate](#) returns the Greatest Common Divisor of two numbers.

## **Declare Syntax :**

Declare Function cRotate Lib "t2win-32.dll" (ByVal Value As Long, ByVal Rotation As Integer) As Long

## **Call Syntax :**

```
test& = cRotate(Value&)
```

## **Where :**

Value&	is the number to be proceeded
test&	is the rotated number

## **Comments :**

## **Examples :**

```
Debug.Print cRotate(1234567, 3)           '-> 5671234
```

**See also :** [Math](#)



# SortDigits

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[SortDigits](#) sorts the digits of a number.

## **Declare Syntax :**

```
Declare Function cSortDigits Lib "t2win-32.dll" (ByVal Value As Long) As Long
```

## **Call Syntax :**

```
test& = cSortDigits(Value&)
```

## **Where :**

Value&	is the number to be proceeded
test&	is the sorted number

## **Comments :**

## **Examples :**

```
Debug.Print cRotate(43218765)           '-> 12345678
```

**See also :** [Math](#)

# SumDigits, SumDigitsAlt, SumDivisors

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

**SumDigits** returns the sum of the digits of a number.

**SumDigitsAlt** returns the alternating sum of the digits of a number.

**SumDivisors** returns the sum of all possible divisors of a number, number not included.

## Declare Syntax :

```
Declare Function cSumDigits Lib "t2win-32.dll" (ByVal Value As Long) As Long
```

```
Declare Function cSumDigitsAlt Lib "t2win-32.dll" (ByVal Value As Long) As Long
```

```
Declare Function cSumDivisors Lib "t2win-32.dll" (ByVal Value As Long) As Long
```

## Call Syntax :

```
test& = cSumDigits(Value&)
```

```
test& = cSumDigitsAlt(Value&)
```

```
test& = cSumDivisors(Value&)
```

## Where :

Value&                    is the number to be proceeded

test&                     is the result

## Comments :

For SumDigitsAlt : even digits are subtracted, odd digits are added.

## Examples :

```
Debug.Print cSumDigits(132546576)            '-> 39  
Debug.Print cSumDigitsAlt(132546576)        '-> -3  
Debug.Print cSumDivisors(12)                '-> 16
```

**See also** : [Math](#)

# WrapLine

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[WrapLine](#) wraps a line in multiple lines with a maximum length by line.

## Declare Syntax :

Declare Function cWrapLine Lib "[time2win.dll](#)" (ByVal Text As String, ByVal Length As Integer, ByVal Separator As String, ByVal EndOfLine As String) As String

## Call Syntax :

```
test$ = cWrapLine(Text$, Length%, Separator$, EndOfLine$)
```

## Where :

Text\$	is the string to be wrapped.
Length%	is the maximum length by lines.
Separator\$	is the word separator.
EndOfLine\$	is the end of line characters on each line.
Test\$	is the wrapped string.

## Comments :

The size of the string must be greater than 1.

## Examples :

```
Dim Str1 As String
```

```
Str1 = "This is a line with some text, which is to long "  
Str1 = Str1 & "for the purpose we need it for. This "  
Str1 = Str1 & "means we have to do some word wrapping. "  
Str1 = Str1 & "That's why we need the cWrapLine() function."
```

```
Debug.Print cWrapLine(Str1, 30, " ", vbCrLf)
```

This is a line with some text,  
which is to long for the  
purpose we need it for. This  
means we have to do some word  
wrapping. That's why we need  
the cWrapLine() function.

**See also :** [String](#)

# SetCapsLock, SetNumLock, SetScrollLock

**QuickInfo** : [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[SetCapsLock](#) sets the CAPS LOCK key On or Off.

[SetInsert](#) sets the NUM LOCK key On or Off.

[SetNumLock](#) sets the NUM LOCK key On or Off.

[SetScrollLock](#) sets the SCROLL LOCK key On or Off.

## Declare Syntax :

```
Declare Function cSetCapsLock Lib "time2win.dll" (ByVal iOnOff As Integer) As Integer
```

```
Declare Function cSetInsert Lib "time2win.dll" (ByVal iOnOff As Integer) As Integer
```

```
Declare Function cSetNumLock Lib "time2win.dll" (ByVal iOnOff As Integer) As Integer
```

```
Declare Function cSetScrollLock Lib "time2win.dll" (ByVal iOnOff As Integer) As Integer
```

## Call Syntax :

```
test% = cSetCapsLock(iOnOff%)
```

```
test% = cSetInsert(iOnOff%)
```

```
test% = cSetNumLock(iOnOff%)
```

```
test% = cSetScrollLock(iOnOff%)
```

## Where :

iOnOff%	True to set the key On False to set the key Off
test%	TRUE if no error FALSE if error

## Comments :

## Examples :

**See also** : [Windows 95](#)

' definition for error type for FileWrapLine

```
Public Const WRAP_SEPARATOR_IS_EMPTY = -1
Public Const WRAP_EOL_IS_EMPTY = -2
Public Const WRAP_BAD_LENGTH_OF_LINE = -3
Public Const WRAP_CANT_INITIALIZE_IN_LINE_BUFFER = -4
Public Const WRAP_CANT_INITIALIZE_OUT_LINE_BUFFER = -5
Public Const WRAP_CANT_INITIALIZE_WORD_BUFFER = -6
Public Const WRAP_CANT_INITIALIZE_BUFFER_1 = -7
Public Const WRAP_CANT_OPEN_SOURCE = -8
Public Const WRAP_CANT_CREATE_TARGET = -9
Public Const WRAP_CANT_READ_LINE = -10
Public Const WRAP_CANT_WRITE_LINE = -11
```

# TrashFile

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## **Purpose :**

[TrashFile](#) sends a file to the trash with persistence and confirmation.

## **Declare Syntax :**

Declare Function cTrashFile Lib "[time2win.dll](#)" (ByVal sFileName As String, ByVal iPermanent As Integer, ByVal iConfirm As Integer) As Integer

## **Call Syntax :**

test% = cTrashFile(sFileName\$, iPermanent%, iConfirm%)

## **Where :**

sFileName\$	the filename to proceed
iPermanent%	= False : preserves undo information, if possible; = True : doesn't preserve undo information.
iConfirm%	= False : responds with "yes to all" for any dialog box that is displayed; = True : doesn't respond "yes to all"
test%	TRUE if all is OK <> TRUE if an error has occurred

## **Comments :**

**See also :** [Windows](#)

' error codes for GetAllSettings

```
Public Const GET_ALL_BAD_DIMENSION = 0  
Public Const GET_ALL_CANT_OPEN_SECTION = -1  
Public Const GET_ALL_NOT_ENOUGH_ELEMENT = -2
```

# ArrayLookup

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[ArrayLookup](#) searches an element in the first dimension of a two-dimensionnal string array and returns value in the second dimension.

## Declare Syntax :

Declare Function cArrayLookup Lib "[time2win.dll](#)" (Strarray() As String, ByVal Value As String, ByVal Sensitivity As Integer) As String

## Call Syntax :

```
test$ = cArrayLookup(Strarray(), Value$, Sensitivity%)
```

## Where :

Strarray()	is the string array (2 dimension).
Value\$	is the string to search.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
test\$	the returned value corresponding to Value\$

## Comments :

This function can be used with [cGetAllSettings](#) to search the value for a specified key.

## Examples :

```
Dim StrArray(1 TO 5, 1 TO 2) As String
```

```
StrArray(1,1) = "Top": StrArray(1,2) = "75 "  
StrArray(1,1) = "Left": StrArray(1,2) = "50"  
StrArray(1,1) = "Test": StrArray(1,2) = "1234567890"
```

```
Debug.Print "> ", cArrayLookup(Strarray(), "test", False)  
Debug.Print "> ", cArrayLookup(Strarray(), "top", False)  
Debug.Print "> ", cArrayLookup(Strarray(), "test", True)  
Debug.Print "> ", cArrayLookup(Strarray(), "top", True)
```

```
>          1234567890  
>          75  
>  
>
```

**See Also :** [Array](#)



# GetShortPathName

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT4.0x}, MSOffice 95

## Purpose :

[GetShortPathName](#) retrieves the short path form of a specified input path.

## Declare Syntax :

```
Declare Function cGetShortPathName Lib "time2win.dll" (ByVal sLongPath As String) As String
```

## Call Syntax :

```
test$ = cGetShortPathName(sLongPath$)
```

## Where :

sLongPath\$	file to be used to find the short path name.
test\$	the short path name else an empty string.

## Comments :

The sLongPath\$ can be a file or a directory.

## Examples :

```
Debug.Print cGetShortPathName("D:\Program Files\Publication Web\Web Publisher Wizard.exe") ' D:\PROGRA~1\
PUBLIC~1\WEBPUB~1.EXE
Debug.Print cGetShortPathName("D:\Program Files\Publication Web") ' D:\
PROGRA~1\PUBLIC~1
```

**See also** : [Windows 95](#)

# FileWrapLine

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[FileWrapLine](#) wraps a file in multiple lines with a maximum length by line.

## Declare Syntax :

Declare Function cFileWrapLine Lib "[time2win.dll](#)" (ByVal sSourceFile As String, ByVal sTargetFile As String, ByVal Length As Long, ByVal Separator As String, ByVal EndOfLine As String) As Long

## Call Syntax :

Test& = cFileWrapLine(sSourceFile\$, sTargetFile\$, Length%, Separator\$, EndOfLine\$)

## Where :

sSourceFile\$	is the source file to be wrapped.
sTargetFile\$	is the target file wrapped.
Length%	is the maximum length by lines.
Separator\$	is the word separator.
EndOfLine\$	is the end of line characters on each line.
Test%	> 0 : the size of the target file. < 0 : see <a href="#">error code</a> .

## Comments :

## Examples :

Test& = cFileWrapLine("c:\autoexec.bat", "c:\autoexec.wrap", 78, " ", vbCrLf)

**See also :** [File](#)

# GetPrinterX, SetPrinterX

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[GetPrinterCopies](#) gets the number of copies of a printer.  
[GetPrinterDefaultSource](#) gets the default source of a printer.  
[GetPrinterDitherType](#) gets the dither type of pictures of a printer.  
[GetPrinterOrientation](#) gets the paper orientation of a printer.  
[GetPrinterPaper](#) gets the paper size of a printer.  
[GetPrinterQuality](#) gets the quality for a printer.

[SetPrinterCopies](#) sets the number of copies for a printer.  
[SetPrinterDefault](#) sets the default printer.  
[SetPrinterDefaultSource](#) sets the default source for a printer.  
[SetPrinterDitherType](#) sets the dither type of pictures for a printer.  
[SetPrinterOrientation](#) sets the paper orientation for a printer.  
[SetPrinterQuality](#) sets the quality for a printer.

## Declare Syntax :

```
Declare Function cGetPrinterCopies Lib "time2win.dll" (ByVal sPrinterName As String, iCopies As Integer) As Integer
Declare Function cGetPrinterDefaultSource Lib "time2win.dll" (ByVal sPrinterName As String, iDefaultSource As Integer) As Integer
Declare Function cGetPrinterDitherType Lib "time2win.dll" (ByVal sPrinterName As String, iDitherType As Integer) As Integer
Declare Function cGetPrinterOrientation Lib "time2win.dll" (ByVal sPrinterName As String, iOrientation As Integer) As Integer
Declare Function cGetPrinterQuality Lib "time2win.dll" (ByVal sPrinterName As String, iQuality As Integer) As Integer
Declare Function cGetPrinterPaper Lib "time2win.dll" (ByVal sPrinterName As String, iSize As Integer) As Integer

Declare Function cSetPrinterCopies Lib "time2win.dll" (ByVal sPrinterName As String, ByVal iCopies As Integer) As Integer
Declare Function cSetPrinterDefault Lib "time2win.dll" (ByVal sPrinterName As String) As Integer
Declare Function cSetPrinterDefaultSource Lib "time2win.dll" (ByVal sPrinterName As String, ByVal iDefaultSource As Integer) As Integer
Declare Function cSetPrinterDitherType Lib "time2win.dll" (ByVal sPrinterName As String, ByVal iDitherType As Integer) As Integer
Declare Function cSetPrinterOrientation Lib "time2win.dll" (ByVal sPrinterName As String, ByVal iOrientation As Integer) As Integer
Declare Function cSetPrinterQuality Lib "time2win.dll" (ByVal sPrinterName As String, ByVal iQuality As Integer) As Integer
```

## Call Syntax :

```
intResult% = cGetPrinterCopies(sPrinterName$, iCopies%)
intResult% = cGetPrinterDefaultSource(sPrinterName$, iDefaultSource%)
intResult% = cGetPrinterDitherType(sPrinterName$, iDitherType%)
intResult% = cGetPrinterOrientation(sPrinterName$, iOrientation%)
intResult% = cGetPrinterPaper(sPrinterName$, iSize%)
intResult% = cGetPrinterQuality(sPrinterName$, iQuality%)

intResult% = cSetPrinterCopies(sPrinterName$, iCopies%)
intResult% = cSetPrinterDefault(sPrinterName$)
intResult% = cSetPrinterDefaultSource(sPrinterName$, iDefaultSource%)
intResult% = cSetPrinterDitherType(sPrinterName$, iDitherType%)
intResult% = cSetPrinterOrientation(sPrinterName$, iOrientation%)
intResult% = cSetPrinterQuality(sPrinterName$, iQuality%)
```

## Where :

sPrinterName\$ is the name of the printer to be proceeded (can be "LPTx:" or the name of the printer);  
intResult% see [error codes](#)

**Comments :**

To get the default printer, use [GetDefaultPrinter](#).

**Examples :**

```
Debug.Print cSetPrinterCopies("LPT1:", 7)

Debug.Print cSetPrinterDefaultSource("LPT1:", SP_SOURCE_LOWER)

Debug.Print cSetPrinterDitherType("LPT1:", SP_DITHER_LINEART)

Debug.Print cSetPrinterOrientation("LPT1:", SP_ORIENTATION_PORTRAIT)
Debug.Print cSetPrinterOrientation("Desktop Printer", SP_ORIENTATION_PORTRAIT)
Debug.Print cSetPrinterOrientation("Network Printer", SP_ORIENTATION_LANDSCAPE)

Debug.Print cSetPrinterQuality("LPT1:", SP_QUALITY_HIGH)

Dim iValue As Integer

Debug.Print cGetPrinterCopies("LPT1:", iValue), iValue

Debug.Print cGetPrinterDefaultSource("LPT1:", iValue), iValue

Debug.Print cGetPrinterDitherType("LPT1:", iValue), iValue

Debug.Print cGetPrinterOrientation("LPT1:", iValue), iValue
Debug.Print cGetPrinterOrientation("Desktop Printer", iValue), iValue
Debug.Print cGetPrinterOrientation("Network Printer", iValue), iValue

Debug.Print cGetPrinterPaper("LPT1:", iValue), iValue

Debug.Print cGetPrinterQuality("LPT1:", iValue), iValue

Debug.Print cGetDefaultPrinter("Hp LaserJet III")
Debug.Print cSetPrinterDefault("Hp LaserJet III")
```

**See also :** [Printer](#)

' codes for SetPrinterDefaultSource

```
Public Const SP_SOURCE_FIRST = 1
Public Const SP_SOURCE_UPPER = 1
Public Const SP_SOURCE_ONLYONE = 1
Public Const SP_SOURCE_LOWER = 2
Public Const SP_SOURCE_MIDDLE = 3
Public Const SP_SOURCE_MANUAL = 4
Public Const SP_SOURCE_ENVELOPE = 5
Public Const SP_SOURCE_ENVMANUAL = 6
Public Const SP_SOURCE_AUTO = 7
Public Const SP_SOURCE_TRACTOR = 8
Public Const SP_SOURCE_SMALLFMT = 9
Public Const SP_SOURCE_LARGEFORMAT = 10
Public Const SP_SOURCE_LARGEFORMATCAPACITY = 11
Public Const SP_SOURCE_CASSETTE = 14
Public Const SP_SOURCE_FORMSOURCE = 15
Public Const SP_SOURCE_LAST = SP_SOURCE_FORMSOURCE
Public Const SP_SOURCE_USER = 256
```

' device specific bins start here

' error codes for cSetPrinterX

```
Public Const SP_SUCCESS = -1
Public Const SP_PRINTER_NAME_EMPTY = 1
Public Const SP_CANT_OPEN_PRINTER = 2
Public Const SP_CANT_GET_PRINTER_PASS1 = 3
Public Const SP_CANT_ALLOCATE_MEMORY = 4
Public Const SP_CANT_LOCK_MEMORY = 5
Public Const SP_CANT_GET_PRINTER_PASS2 = 6
Public Const SP_CANT_UPDATE_DRIVER = 7
Public Const SP_CANT_SET_PRINTER = 8
Public Const SP_COMMAND_NOT_SUPPORTED = 9
```

' codes for SetPrinterOrientation

```
Public Const SP_ORIENTATION_PORTRAIT = 1    ' Portrait
Public Const SP_ORIENTATION_LANDSCAPE = 2    ' Landscape
```

' codes for SetPrinterQuality

Public Const SP\_QUALITY\_DRAFT = -1

Public Const SP\_QUALITY\_LOW = -2

Public Const SP\_QUALITY\_MEDIUM = -3

Public Const SP\_QUALITY\_HIGH = -4



' codes for SetPrinterDitherType

Public Const SP\_DITHER\_NONE = 1

Public Const SP\_DITHER\_COARSE = 2

Public Const SP\_DITHER\_FINE = 3

Public Const SP\_DITHER\_LINEART = 4

Public Const SP\_DITHER\_GRAYSCALE = 5

Public Const SP\_DITHER\_USER = 256

' No dithering

' Dither with a coarse brush

    ' Dither with a fine brush

' LineArt dithering

' Device does grayscale

' Device-specific dithers start here

' codes for GetPrinterSize

```
Public Const DM_PAPER_FIRST = 1
Public Const DM_PAPER_LETTER = 1
Public Const DM_PAPER_LETTERS SMALL = 2
Public Const DM_PAPER_TABLOID = 3
Public Const DM_PAPER_LEDGER = 4
Public Const DM_PAPER_LEGAL = 5
Public Const DM_PAPER_STATEMENT = 6
Public Const DM_PAPER_EXECUTIVE = 7
Public Const DM_PAPER_A3 = 8
Public Const DM_PAPER_A4 = 9
Public Const DM_PAPER_A4SMALL = 10
Public Const DM_PAPER_A5 = 11
Public Const DM_PAPER_B4 = 12
Public Const DM_PAPER_B5 = 13
Public Const DM_PAPER_FOLIO = 14
Public Const DM_PAPER_QUARTO = 15
Public Const DM_PAPER_10X14 = 16
Public Const DM_PAPER_11X17 = 17
Public Const DM_PAPER_NOTE = 18
Public Const DM_PAPER_ENV_9 = 19
Public Const DM_PAPER_ENV_10 = 20
Public Const DM_PAPER_ENV_11 = 21
Public Const DM_PAPER_ENV_12 = 22
Public Const DM_PAPER_ENV_14 = 23
Public Const DM_PAPER_CSHEET = 24
Public Const DM_PAPER_DSHEET = 25
Public Const DM_PAPER_ESHEET = 26
Public Const DM_PAPER_ENV_DL = 27
Public Const DM_PAPER_ENV_C5 = 28
Public Const DM_PAPER_ENV_C3 = 29
Public Const DM_PAPER_ENV_C4 = 30
Public Const DM_PAPER_ENV_C6 = 31
Public Const DM_PAPER_ENV_C65 = 32
Public Const DM_PAPER_ENV_B4 = 33
Public Const DM_PAPER_ENV_B5 = 34
Public Const DM_PAPER_ENV_B6 = 35
Public Const DM_PAPER_ENV_ITALY = 36
Public Const DM_PAPER_ENV_MONARCH = 37
Public Const DM_PAPER_ENV_PERSONAL = 38
Public Const DM_PAPER_FANFOLD_US = 39
Public Const DM_PAPER_FANFOLD_STD_GERMAN = 40
Public Const DM_PAPER_FANFOLD_LGL_GERMAN = 41
Public Const DM_PAPER_ISO_B4 = 42
Public Const DM_PAPER_JAPANESE_POSTCARD = 43
Public Const DM_PAPER_9X11 = 44
Public Const DM_PAPER_10X11 = 45
Public Const DM_PAPER_15X11 = 46
Public Const DM_PAPER_ENV_INVITE = 47
Public Const DM_PAPER_RESERVED_48 = 48
Public Const DM_PAPER_RESERVED_49 = 49
Public Const DM_PAPER_LETTER_EXTRA = 50
Public Const DM_PAPER_LEGAL_EXTRA = 51
Public Const DM_PAPER_TABLOID_EXTRA = 52
Public Const DM_PAPER_A4_EXTRA = 53
Public Const DM_PAPER_LETTER_TRANSVERSE = 54
Public Const DM_PAPER_A4_TRANSVERSE = 55
Public Const DM_PAPER_LETTER_EXTRA_TRANSVERSE = 56
Public Const DM_PAPER_A_PLUS = 57
Public Const DM_PAPER_B_PLUS = 58
Public Const DM_PAPER_LETTER_PLUS = 59

' Letter 8 1/2 x 11 in
' Letter Small 8 1/2 x 11 in
' Tabloid 11 x 17 in
' Ledger 17 x 11 in
' Legal 8 1/2 x 14 in
' Statement 5 1/2 x 8 1/2 in
' Executive 7 1/4 x 10 1/2 in
' A3 297 x 420 mm
' A4 210 x 297 mm
' A4 Small 210 x 297 mm
' A5 148 x 210 mm
' B4 (JIS) 250 x 354
' B5 (JIS) 182 x 257 mm
' Folio 8 1/2 x 13 in
' Quarto 215 x 275 mm
' 10x14 in
' 11x17 in
' Note 8 1/2 x 11 in
' Envelope #9 3 7/8 x 8 7/8
' Envelope #10 4 1/8 x 9 1/2
' Envelope #11 4 1/2 x 10 3/8
' Envelope #12 4 1/2 x 11
' Envelope #14 5 x 11 1/2
' C size sheet
' D size sheet
' E size sheet
' Envelope DL 110 x 220mm
' Envelope C5 162 x 229 mm
' Envelope C3 324 x 458 mm
' Envelope C4 229 x 324 mm
' Envelope C6 114 x 162 mm
' Envelope C65 114 x 229 mm
' Envelope B4 250 x 353 mm
' Envelope B5 176 x 250 mm
' Envelope B6 176 x 125 mm
' Envelope 110 x 230 mm
' Envelope Monarch 3.875 x 7.5 in
' 6 3/4 Envelope 3 5/8 x 6 1/2 in
' US Std Fanfold 14 7/8 x 11 in
' German Std Fanfold 8 1/2 x 12 in
' German Legal Fanfold 8 1/2 x 13 in
' B4 (ISO) 250 x 353 mm
' Japanese Postcard 100 x 148 mm
' 9 x 11 in
' 10 x 11 in
' 15 x 11 in
' Envelope Invite 220 x 220 mm
' RESERVED--DO NOT USE
' RESERVED--DO NOT USE
' Letter Extra 9 1/2 x 12 in
' Legal Extra 9 1/2 x 15 in
' Tabloid Extra 11.69 x 18 in
' A4 Extra 9.27 x 12.69 in
' Letter Transverse 8 1/2 x 11 in
' A4 Transverse 210 x 297 mm
' Letter Extra Transverse 9 1/2 x 12 in
' SuperA/SuperA/A4 227 x 356 mm
' SuperB/SuperB/A3 305 x 487 mm
' Letter Plus 8.5 x 12.69 in
```

Public Const DM_PAPER_A4_PLUS = 60	' A4 Plus 210 x 330 mm
Public Const DM_PAPER_A5_TRANSVERSE = 61	' A5 Transverse 148 x 210 mm
Public Const DM_PAPER_B5_TRANSVERSE = 62	' B5 (JIS) Transverse 182 x 257 mm
Public Const DM_PAPER_A3_EXTRA = 63	' A3 Extra 322 x 445 mm
Public Const DM_PAPER_A5_EXTRA = 64	' A5 Extra 174 x 235 mm
Public Const DM_PAPER_B5_EXTRA = 65	' B5 (ISO) Extra 201 x 276 mm
Public Const DM_PAPER_A2 = 66	' A2 420 x 594 mm
Public Const DM_PAPER_A3_TRANSVERSE = 67	' A3 Transverse 297 x 420 mm
Public Const DM_PAPER_A3_EXTRA_TRANSVERSE = 68	' A3 Extra Transverse 322 x 445 mm
Public Const DM_PAPER_LAST = DM_PAPER_A3_EXTRA_TRANSVERSE	
Public Const DM_PAPER_USER = 256	

# RasGetCountryInfo

**QuickInfo** : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[RasGetCountryInfo](#) retrieves country-specific dialing information from the Windows Telephony list of countries.

## Declare Syntax :

Declare Function cRasGetCountryInfo Lib "time2win.dll" (ByVal iCountry As Integer, COUNTRYINFO As tagCOUNTRYINFO, ByVal iFirstNext As Integer) As Integer

## Call Syntax :

```
intResult% = cRasGetCountryInfo(iCountry%, COUNTRYINFO, iFirstNext%)
```

## Where :

iCountry%	= 0 : can enumerate all country code; <>0 : retrieves information about a specific country code
COUNTRYINFO	is the type'd tagCOUNTRYINFO;
iFirstNext%	TRUE : begin the enumeration and return the first country; FALSE : continue the enumeration and return the next country;
intResult%	RAS_SUCCESS : all is ok RAS_CANT_ACCESS_COUNTRY: can't access the specified country RAS_NO_MORE_COUNTRY : no more country

## Comments :

## Examples :

```
Dim CI As tagCOUNTRYINFO
intResult = cRasGetCountryInfo(0, CI, True)
Do While (intResult = RAS_SUCCESS)
    Debug.Print CI.iCountryCode, CI.iCountryID, CI.sCountryName
    intResult = cRasGetCountryInfo(0, CI, False)
Loop
```

**See also** : [Remote Access Service](#)

# Remote Access Service : Overview

[RasGetCountryInfo](#)  
countries.

retrieves country-specific dialing information from the Windows Telephony list of

# EnumPrintersX

**QuickInfo :** VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) | VB 5.0 | VBA 5.0 {Win95/WinNT}, MSOffice 95

## Purpose :

[EnumPrinters1](#) enumerates all printers (data returned are description, printer name, comment).

[EnumPrinters2](#) enumerates all printers (data returned are for remote printers).

[EnumPrinters5](#) enumerates all printers (data returned are printer name, port name).

## Declare Syntax :

Declare Function cEnumPrinters1 Lib "time2win.dll" (PRINTER1 As tagPRINTER1, ByVal iFirstNext As Integer) As Integer

Declare Function cEnumPrinters2 Lib "time2win.dll" (ByVal sServerName As String, PRINTER2 As tagPRINTER2, ByVal iFirstNext As Integer) As Integer

Declare Function cEnumPrinters5 Lib "time2win.dll" (PRINTER5 As tagPRINTER5, ByVal iFirstNext As Integer) As Integer

## Call Syntax :

```
intResult% = cEnumPrinters1(PRINTER1, FirstNext%)
```

```
intResult% = cEnumPrinters2(PRINTER2, FirstNext%)
```

```
intResult% = cEnumPrinters5(PRINTER5, FirstNext%)
```

## Where :

sServerName is the name of the server for enumerate the printer (**CAN'T BE THE SERVER ITSELF**);

PRINTER1 is the type'd tagPRINTER1;

PRINTER2 is the type'd tagPRINTER2;

PRINTER5 is the type'd tagPRINTER5;

FirstNext% TRUE : begin the enumeration and return the first job;

FALSE : continue the enumeration and return the next job;

intResult% EP\_SUCCESS : all is ok

EP\_STRANGE\_ERROR : unknow error when accessing the enumeration

EP\_CANT\_ENUMERATE\_MORE\_PRINTERS : no more printers

## Comments :

For cEnumPrinters1 :

For cEnumPrinters2 :

For cEnumPrinters5 :

## Examples :

```
Dim intResult As Integer
```

```
Dim P1 As tagPRINTER1
```

```
Dim P2 As tagPRINTER2
```

```
Dim P5 As tagPRINTER5
```

```
Debug.Print "EnumPrinters1" & vbCrLf
```

```
intResult = cEnumPrinters1(P1, True)
```

```
Do While (intResult = EP_SUCCESS)
    Debug.Print P1.sDescription, P1.sPrinterName, P1.sComment
    intResult = cEnumPrinters1(P1, False)
Loop
Debug.Print "EnumPrinters2" & vbCrLf
intResult = cEnumPrinters2(P2, True)
Do While (intResult = EP_SUCCESS)
    Debug.Print P2.sServerName, P2.sPrinterName, P2.sPortName, P2.sShareName, P2.sDriverName,
P2.sComment, P2.sLocation
    intResult = cEnumPrinters2(P2, False)
Loop
Debug.Print "EnumPrinters5" & vbCrLf
intResult = cEnumPrinters5(P5, True)
Do While (intResult = EP_SUCCESS)
    Debug.Print P5.sPrinterName, P5.sPortName
    intResult = cEnumPrinters5(P5, False)
Loop
```

**See also :** [Printer](#)

' constant for RunFile

```
Public Const SW_HIDE = 0
Public Const SW_SHOWNORMAL = 1
Public Const SW_NORMAL = 1
Public Const SW_SHOWMINIMIZED = 2
Public Const SW_SHOWMAXIMIZED = 3
Public Const SW_MAXIMIZE = 3
Public Const SW_SHOWNOACTIVATE = 4
Public Const SW_SHOW = 5
Public Const SW_MINIMIZE = 6
Public Const SW_SHOWMINNOACTIVE = 7
Public Const SW_SHOWNA = 8
Public Const SW_RESTORE = 9
Public Const SW_SHOWDEFAULT = 10
Public Const SW_MAX = 10
```



# RunFile

**QuickInfo :** [VB 3.0](#), [VB 4.0 \(16-Bit\)](#), [VB 4.0 \(32-Bit\)](#) | [VB 5.0](#) | [VBA 5.0 {Win95/WinNT}](#), [MSOffice 95](#)

## Purpose :

[RunFile](#) opens a specified executable or document file.

## Declare Syntax :

Declare Function cRunFile Lib "[time2win.dll](#)" (ByVal sFileName As String, ByVal sFileParameter As String, ByVal sFilePath As String, ByVal iRunStyle As Integer) As Long

## Call Syntax :

lngResult& = cRunFile(sFileName\$, sFileParameter\$, sFilePath\$, iRunStyle%)

## Where :

sFileName\$	is the filename to be executed.
sFileParameter\$	is the parameter list to be passed to the filename.
sFilePath\$	is the file path to be the default directory.
iRunStyle%	constant.
constant.	specifies how the application is to be shown when it is opened. Use one the following
lngResult&	<= 32 : an error has occurred
	> 32 : is the instance handle of the application that was run.

## Comments :

## Examples :

```
Debug.Print cRunFile("notepad.exe", "c:\temp\shortcut.test", "c:\temp", SW_NORMAL)
```

**See also :** [Windows](#)



