T I M E   T O   W I N

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

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

TIME TO WIN for VB 3.0	: TIME2WIN.DLL	: 8.00
TIME TO WIN for VB 4.0 (16-Bit)	: T2WIN-16.DLL	: 7.08
TIME TO WIN for VB 4.0 (32-Bit)	: T2WIN-32.DLL	: 2.10
TIME TO WIN for MSOffice 95	: T2WOFFIC.DLL	: 1.11

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 WINNT35\SYSTEM32 directory.

Registered version :

The files TIME2WIN.DLL, TIME2WIN.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT35\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 WINNT35\SYSTEM32 directory.

Registered version :

The files T2WIN-16.DLL, T2WIN-16.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT35\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 WINNT35\SYSTEM32 directory.

Registered version :

The files T2WIN-32.DLL, T2WIN-32.HLP should be copied in your WIN95\SYSTEM and/or WINNT35\SYSTEM32 directory.
The file T2WIN-32.LIC should be copied in your WIN95 directory.

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.LIC 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 WINNT35\SYSTEM32 directory.

Registered version :

The files T2WOFFIC.DLL, T2WOFFIC.HLP should be copied in your WINDOWS\SYSTEM or WIN95\SYSTEM and/or WINNT35\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 CompuServe :

MSBASIC forum
VBPJ forum
MSACCESS forum

On Internet :

TIME2WIN.ZIP	(ftp.winsite.com/pub/pc/win3/programr/vbasic)
T2WIN-16.ZIP	(ftp.winsite.com/pub/pc/win3/programr/vbasic)
T2WIN-32.ZIP	(ftp.winsite.com/pub/pc/win95/programr/vbasic)
MCVBEHTP.ZIP	(ftp.winsite.com/pub/pc/win95/programr/vbasic)
MCSECURE.ZIP	(ftp.winsite.com/pub/pc/win95/programr/vbasic)

CompuServe Mail:

Name : Michaël 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: Michaël 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 [TIME TO WIN.LIC] 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 [TIME TO WIN.LIC] 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
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.LIC
TIME TO WIN for MSOffice 95	: T2WOFFIC.LIC

Acknowledgement

Thanks to Andreas Thoele for some translations in German language.
Thanks to Silvio Sorrentino for some translations in Italian language.
Thanks to Manuel Tobarra Narro for some translations in Spanish language.
Thanks to Pawel Mandalian for some translations in Polish language.
Thanks to Joan Ludevid for some translations in Catalan language.

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).

This help has been writed by using ForeHelp v1.04 from ForeFront, Inc.

For TIME TO WIN (32-Bit), special thanks for registered user who have asked me some new functions :

Guillermo Kunst for cEnumPrinterJobs.
Norm Zastre for c3DWeightAverage, cFProcessAsciiFile, cFGotoRecord.

Other products

Basis products :

1) TIME TO WIN (VB 3.0 or VB 4.0 (16-Bit))

   This product is a powerfull 16-Bit DLL with more than 640 routines for VB 3.0 and VB 4.0 (16-Bit) application.
   You can register thru CompuServe SWREG #4045 for $61.00
   You can download a demo called TIME2WIN.ZIP for VB 3.0 and T2WIN-16.ZIP for VB 4.0 (16-Bit), either in MSBASIC and VBPJ forum.
   On Internet : TIME2WIN.ZIP (ftp.winsite.com/pub/pc/win3/programr/vbasic)
   T2WIN-16.ZIP (ftp.winsite.com/pub/pc/win3/programr/vbasic)

2) TIME TO WIN (VB 4.0 (32-Bit))

   This product is a powerfull 32-Bit DLL with more than 642 routines for VB 4.0 (32-Bit) application.
   You can register thru CompuServe SWREG #7516 for $52.00
   You can download a demo called T2WIN-32.ZIP for VB 4.0 (32-Bit), either in MSBASIC and VBPJ forum.
   On Internet : T2WIN-32.ZIP (ftp.winsite.com/pub/pc/win95/programr/vbasic)

3) TIME TO WIN for PowerBuilder 4.0

   This product is a powerfull 16-Bit DLL with more than 250 routines for PowerBuilder 4.0 application.
   You can register thru CompuServe SWREG #9095 for $38.00
   You can download a demo called T2WPB-16.ZIP for PowerBuilder, in POWERBUILDER forum.

4) TIME TO WIN for MS Office 95

   This product is a powerfull 32-Bit DLL with more than 200 routines for Access 95, Excel 95 and Word 95.
   You can register thru CompuServe SWREG #10355 for $25.00
   You can download a demo called T2WOFFIC.ZIP for Access 7.0, in MSACCESS forum.

5) 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 register thru CompuServe SWREG #4380 for $25.00
   You can download a demo called MCVBEHTP.ZIP for the languages, either in MSBASIC and VBPJ forum.
   On Internet : MCVBEHTP.ZIP (ftp.winsite.com/pub/pc/win95/programr/vbasic)

6) MC SECURITY for VB 4.0 (16/32 Bit)

   This product is a powerfull 16/32-Bit DLL with 16 routines for VB 4.0 (16/32 Bit) application.
   This product cover many aspect of how to protect your application.
   You can register thru CompuServe SWREG #8536 for $10.00
   You can download a demo called MCSECURE.ZIP for VB 4.0 (16/32 Bit), either in MSBASIC and VBPJ forum.
   On Internet : MCSECURE.ZIP (ftp.winsite.com/pub/pc/win95/programr/vbasic)

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))

   This product is an update for registered user of 'TIME TO WIN' which want register the 'TIME TO WIN (32-Bit)'.
   You can register thru CompuServe SWREG #7517 for $29.00
   You can download a demo called T2WIN-32.ZIP for VB 4.0 (32-Bit), either in MSBASIC and VBPJ forum.
   On Internet : T2WIN-32.ZIP (ftp.winsite.com/pub/pc/win95/programr/vbasic)

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 register thru CompuServe SWREG #4379 for $16.00
You can download a demo called MCVBEHTP.ZIP for these languages, either in MSBASIC and VBPJ forum.
On Internet : MCVBEHTP.ZIP (ftp.winsite.com/pub/pc/win95/programr/vbasic)

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

See also : Revision History

Version	Comments
8.08	*no new features.*
7.07	Conversion of a binary string into an integer variable. cB2I Conversion of a binary string into a long variable. cB2L Conversion of a hexa string into an integer variable. cH2I Conversion of a hexa string into a long variable. cH2L Access of method (by position) of OCX custom controls. cObjectMethodByPos Access of method (by name) of OCX custom controls. cObjectMethodByName Reads data in properties (by position) from OCX custom controls. cObjectGetPropertyByPos Reads data in properties (by name) from OCX custom controls. cObjectGetPropertyByName Writes data in properties (by position) in OCX custom controls. cObjectPutPropertyByPos Writes data in properties (by name) from OCX custom controls. cObjectPutPropertyByName
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)
TIME TO WIN for MSOffice 95

TIME TO WIN for VB 3.0 : New features

See also : Revision History

Version	Comments
8.08	*no new features.*
7.01	Adds new functionnalities 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 array. cCountI Counts a specific value in a Long array. cCountL Counts a specific value in a Single array. cCountS Counts a specific value in a Double array. cCountD Searchs a specific value in an Integer array. cSearchI Searchs a specific value in a Long array. cSearchL Searchs a specific value in a Single array. cSearchS Searchs a specific value in a Double array. cSearchD
6.00	Truncates a long path with filename. cTruncatePath Searchs 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 random number. cRndI Returns a long random number. cRndL Returns a single random number. cRndS Returns a double random number. cRndD
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, scalar date, scalar 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.
Changes all chars in a char set by a new char set in a file (text or binary).
cFileChangeChars

5.02 Adds a 3D Meter (rectangle, triangle, trapezium, ellipse, bar) from a Picture Box. c3DMeter

5.00 Adds a 3D visibility to a VB standard control or VBX (custom colors). cCtl3D
Adds a 3D visibility to a VB standard control or VBX (fixed colors). c3D
Returns the Left, Top, Right, Bottom value of a control in Pixels.
cGetCtlRect
Returns the Left, Top, Right, Bottom value of a control in Twips. cGetCtlRectTwips
Center a form on the screen.
cCenterWindow
Explode a window before show.
cShowWindow
Calculates a scalar (long) from a time.
cTimeToScalar
Decomposes a scalar into time parts.
cScalarToTime

4.57 Transfers the contents of an string array to a List Box. cArrayToListBox
Transfers the contents of an string array to a Combo Box.
cArrayToComboBox

4.50 Create a Huge Array.
cHMACreate
Free a Huge Array.
cHMAFree
Read an element from a Huge Array.
cHMAGet
Read a type'd variable from a Huge Array.
cHMAGetType
Save an element to a Huge Array.
cHMAPut
Save a type'd variable to a Huge Array.
cHMAPutType
Clear a Huge Array (fill it with chr$(0) or chr$(32) (for string array)).
cHMAClear
Clear a single Sheet in a Huge Array (fill it with chr$(0) or chr$(32) (for string array)).
cHMAClearSheet
Clear a single Col on on one Sheet or on all sheets in a Huge Array (see above). cHMAClearCol
Clear a single Row on one Sheet or on all Sheets in a Huge Array (see above).
cHMAClearRow
Clear a single Col in a Huge Array with only one sheet.
cHMAsClearCol
Clear a single Row in a Huge Array with only one sheet.
cHMAsClearRow
Read an element from a Huge Array with only one sheet.
cHMAsGet
Read a type'd variable from a Huge Array with only one sheet.
cHMAsGetType
Save an element from a Huge Array with only one sheet.
cHMAsPut
Save a type'd variable from a Huge Array with only one sheet.
cHMAsPutType
Read an element from a Huge Array with only one sheet and one row.
cHMArGet
Read a type'd variable from a Huge Array with only one sheet and one row.
cHMArGetType
Save an element from a Huge Array with only one sheet and one row.
cHMArPut

Save a type'd variable from a Huge Array with only one sheet and one row.
cHMArPutType
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 gived string. cGetInR
Extracts the first/second part from the left of a gived string.
cGetInPart
Extracts the first/second part from the right of a gived 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 specials 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 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.
cFileExpand
Compress a string.
cStringCompress
Expands a string compressed by cStringCompress.
cStringExpand
Fills an array by starting value and increment value. FillIncr
Calculates the determinant of a square matrix.
cMatrixDet
Calculates the cofactor of an element in a square matrix.
cMatrixCoFactor
Calculates the minor of an element in a square matrix. cMatrixMinor
Fills a square matrix.
cMatrixFill
Inverts a square matrix.
cMatrixInv
Creates a symmetrical Toeplitz square matrix.
cMatrixSymToeplitz

1.60 2-D Geometry calculations (14 functions). 2-D Geometry
3-D Geometry calculations (14 functions). 3-D Geometry
Adds two square matrix.
cMatrixAdd
Compares two square matrix.
cMatrixCompare
Copy a square matrix.
cMatrixCopy
Multiply two square matrix. cMatrixMul
Substract two square matrix.
cMatrixSub
Transpose a square matrix.
cMatrixTranspose

1.52 Converts the first letter of each word separated by a space in a string to upper case.
cProperName

1.50 Functions for calculating interest rate (12 functions).
Financial

1.42 Performs the hash algorithm (MD5) to a specified string.
cHashMD5
Adds registration key management. cRegistrationKey
Removes a serialization information (descriptions and number) from a serialized file.
cSerialRmv
Sorts an ASCII file or a BINARY file in ascending or descending order with case sensitive or not. cFileSort
Computes the number of combinations of n items, taken m at a time.
cCombination
Converts an ASCII string into an EBCDIC string.
cCnvASCIItoEBCDIC
Converts an EBCDIC string into an ASCII string.
cCnvEBCDICtoASCII
Opens a file for I/O cFopen
Closes an open stream. cFclose
Reads a single character from a stream. cFgetc
Writes a single character to a stream. cFputc
Writes a line of characters to a stream. cFputs
Reads a line of characters from a stream. cFgets
Writes an arbitrary number of characters to a stream. cFwrite

Reads an arbitrary number of characters from a stream. cFread
Closes all files opened.
cFcloseall
Flushes buffered I/O to a particular stream to disk. cFflush
Flushes buffered I/O for all open streams to disk.
cFflushall
Tests for end-of-file on a stream. cFeof
Tests for an error on a stream. cFerror
Resets the error indicator for a stream.
cFclearerr
Moves the file pointer to a specified location. cFseek
Gets the current position of a file pointer. cFtell
Moves the file pointer to the beginning of a file.
cFrewind

1.36 Arrange all desktop icons.
cArrangeDesktopIcons
Put/Get full variable string array (one dimension) on/from disk.
cArrayStringOnDisk
Put/Get full array (any dimension) on/from disk (very fast routine).
cArrayOnDisk
Extract a sub-string delimited by a separator's list in a gived string.
cTokenIn
Align a string in left, center, right position cAlign
New timer for more accuracy (1 ms in place of 55 ms) cTimer.x.
Increment the serialized number of a serialized file by a value (positive or negative).
cSerialInc
Check if a file is serialized. cIsSerial
Put or Modify a serialization information (descriptions and number) to a file.
cSerialPut
Get a serialization information (descriptions and number) from a file.
cSerialGet
Walk thru the window's list.
cWalkThruWindow
UnHide all edit forms in the VB design environnement.
cUnHideAllEditForm
Hide all edit forms in the VB design environnement.
cHideAllEditForm
UnHide debug form in the VB design environnement.
cUnHideDebugForm
Hide debug form in the VB design environnement.
cHideDebugForm
Multiple AND 'InStr' in one call.
cAndToken, cAndTokenIn
Multiple OR 'InStr' in one call.
cOrToken, cOrTokenIn

1.33 Close all edit forms in the VB design environnement.
cCloseAllEditForm
Create a multiple directory in one call.
cMakeMultipleDir

1.30 Clear a single Sheet in a big sized array (fill it with chr$(0) or chr$(32) (for string array))
cDAClearSheet
Clear a single Col on on one Sheet or on all sheets in a big sized array (see above).
cDAClearCol
Clear a single Row on one Sheet or on all Sheets in a big sized array (see above).
cDAClearRow
Clear a single Col in a big sized array with only one sheet.
cDAsClearCol
Clear a single Row in a big sized array with only one sheet.
cDAsClearRow

Read an element from a big sized array on disk with only one sheet. cDAsGet
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 occurence of a string in a file (search can be case-sensitive or not).
cFileSearchCount
Check the specified ISBN (International Standard Book Numbers). cIsISBN
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 gived 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
cIsFilenameValid

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.
cIsFormEnabled
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) : New features

See also : Revision History

Version	**Comments**
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
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**
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 functionnalities 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 functionnality 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 (.nIsTyped) 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 occured all times.

1.28 Adds TimeOut functionnality (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 cIsFilenameValid.
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 cIsFilenameValid (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)
TIME TO WIN for MSOffice 95

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

See also : New Features

Version	Comments
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

FileCompress	compress a file into a compressed format.
FileExpand	expand a compressed file into a normal format.
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) : Revision history

See also : New Features

Version	Comments
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 usen in the International Message Box and International Input Box are now the icons usen 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) {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 Constants and 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

Version	**Comments**
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) {Win95/WinNT}, MSOffice 95

Purpose :

EnumPrinterJobs enumerate 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 ame 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 'lStatus' 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 + "lJobId : " & JI.lJobId & vbCrLf
        strDisplay = strDisplay + "lStatus : " & JI.lStatus & vbCrLf
        strDisplay = strDisplay + "lPriority : " & JI.lPriority & vbCrLf
        strDisplay = strDisplay + "lPosition : " & JI.lPosition & vbCrLf
        strDisplay = strDisplay + "lStartTime : " & JI.lStartTime & vbCrLf
        strDisplay = strDisplay + "lUntilTime : " & JI.lUntilTime & vbCrLf
        strDisplay = strDisplay + "lTotalPages : " & JI.lTotalPages & vbCrLf
        strDisplay = strDisplay + "lPagesPrinted : " & JI.lPagesPrinted & vbCrLf
        strDisplay = strDisplay + "lSize : " & JI.lSize & vbCrLf
        strDisplay = strDisplay + "lTime : " & JI.lTime & 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)

- 1) On CompuServe GO SWREG
- 2) Choose Register Shareware.
- 3) 'TIME TO WIN' SWREG ID is : #4045. (price is $**61.00**)

As soon as I receive notification of your registration (usually 1 - 3 days) I will send you out via e-Mail the latest version and a license file for one site (only if lastest version is available (not currently in test)) if not you receive the license file for one site.

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)

- 1) On CompuServe GO SWREG
- 2) Choose Register Shareware.
- 3) 'TIME TO WIN (16-Bit)' SWREG ID is : #4045. (price is $**61.00**)

As soon as I receive notification of your registration (usually 1 - 3 days) I will send you out via e-Mail the latest version and a license file for one site (only if lastest version is available (not currently in test)) if not you receive the license file for one site.

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 3.0 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)

- 1) On CompuServe GO SWREG
- 2) Choose Register Shareware.
- 3) 'TIME TO WIN (32-Bit)' SWREG ID is : #7516. (price is $**52.00**)

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

- 1) On CompuServe GO SWREG
- 2) Choose Register Shareware.
- 3) 'UPDATE T2WIN -> T2WIN (32-Bit)' SWREG ID is : #7517. (price is $**29.00**)

As soon as I receive notification of your registration (usually 1 - 3 days) I will send you out via e-Mail the latest version and a license file for one site (only if lastest version is available (not currently in test)) if not you receive the license file for one site.

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++ 4.00.
The code has been optimized for 80486 use with the 'maximize speed' option.

'TIME TO WIN (32-Bit)' can only be used with Visual Basic 4.0 (32-Bit Edition) 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)

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

As soon as I receive notification of your registration (usually 1 - 3 days) I will send you out via e-Mail the latest version and a license file for one site (only if lastest version is available (not currently in test)) if not you receive the license file for one site.

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) {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 substract (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 and Visual Basic 4.0 (16/32-Bit).

I'm an Engineer in Electricity and Electronic and I've writed '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 very limited.

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

'TIME TO WIN' contains more over ***645*** functions or subroutines.
You can find functions or routines over the following sections :

ArrayOnDisk

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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

ArrayOnDisk, ArrayStringOnDisk	Put/Get full array on/from disk.
AddD, AddI, AddL, AddS	Adding a value to all elements in an array
ArrayPrm	Read the configuration of an array.
CountD, CountI, CountL, CountS	Count a specific value in an array.
DeviationD, DeviationI, DeviationL, DeviationS	Calculating the standard deviation from all elements in an array
FillD, FillI, FillL, FillS	Filling an array with a value incremented by one for any element.
FillIncrD, FillIncrI, FllIncrL, FillIncrS	Filling an array with a value incremented by an increment for any element.
MaxD, MaxI, MaxL, MaxS	Finding the maximum value in an array.
MeanD, MeanI, MeanL, MeanS	Calculating the mean from all elements in an array.
MinD, MinI, MinL, MinS	Finding the minimum value in an array.
ReverseSortD, ReverseSortI, ReverseSortL, ReverseSortS	Sort an array in descending order.
ReverseSortStr	Sort, in descending order, a string.
SearchD, SearchI, SearchL, SearchS	Search a specific value in an array.
SetD, SetI, SetL, SetS	Setting all elements in an array with the same value.
SortD, SortI, SortL, SortS	Sort an array in ascending order.
SortStr	Sort, in ascending order, a string.
SumD, SumI, SumL, SumS	Sum all elements from an array.

DeviationD, DeviationI, DeviationL, DeviationS

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

Purpose :

ArrayPrm retrieve the definition of a gived 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 gived 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
Dim status                              As Integer

status = cArrayPrm(array(), arrayDef)

        arrayDef.Bounds               is 458753
        arrayDef.LBound  is 1
        arrayDef.UBound               is 7
        arrayDef.ElemSize             is 4 (LONG)
        arrayDef.IndexCount           is 2
        arrayDef.TotalElem            is 7
```

See also : Array

FillD, FillI, FillL, FillS

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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, FIlIncrL, FillIncrS

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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) {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) {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) {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 :

See Also : Array

SetD, SetI, SetL, SetS

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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 :

See Also : Array

SumD, SumI, SumL, SumS

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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) {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 :

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

ErrCode = cDAClearSheet(DA, 2)                          'clear the Sheet 2 in the big sized array on
disk

ErrCode = cDAClearCol(DA, DA.nCols, 2)                  'clear the last Col in Sheet 2 in the big sized
array on disk
ErrCode = cDAsClearCol(DA, DA.nCols)                    'clear the last Col in Sheet 1 in the big sized
array on disk

ErrCode = cDAClearRow(DA, DA.nRows, 2)                  'clear the last Row in Sheet 2 in the big sized
array on disk
ErrCode = cDAsClearRow(DA, DA.nRows)                    '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) {Win95/WinNT}, MSOffice 95

Purpose :

DAGet read an element from a big sized array on disk.
DArGet have the same functionnality but with a big sized array with only one sheet and only one row.
DAsGet have the same functionnality 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) {Win95/WinNT}, MSOffice 95

Purpose :

DAGetType read a type'd variable from a big sized array on disk.
DArGetType have the same functionnality but with a big sized array with only one sheet and only one row.
DAsGetType have the same functionnality 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\datype.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 = cDACreate(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))                        'read the type'd variable in Row 500, Col 1,
Sheet 1
Call cDAGetType(DA, DA.nRows, DA.nCols, 1, TE(4))           '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) {Win95/WinNT}, MSOffice 95

Purpose :

DAPut save an element to a big sized array on disk.
DArPut have the same functionnality but with a big sized array with only one sheet and only one row.
DAsPut have the same functionnality 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) {Win95/WinNT}, MSOffice 95

Purpose :

DAPutType save a type'd variable from a big sized array on disk.
DArPutType have the same functionnality but with a big sized array with only one sheet and only one row.
DAsPutType have the same functionnality 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\datype.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
```

```
ErrCode = cTasks(TE, False)
Call cDAPutType(DA, 1, DA.nCols, 1, TE)                    'save the type'd variable in Row 1, Col 500,
Sheet 1
ErrCode = cTasks(TE, False)
Call cDAPutType(DA, DA.nRows, 1, 1, TE)                    'save the type'd variable in Row 500, Col 1,
Sheet 1
ErrCode = cTasks(TE, False)
Call cDAPutType(DA, DA.nRows, DA.nCols, 1, TE)             'save the type'd variable in Row 500, Col
500, Sheet 1
```

See also : Disk Array

DACreate

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 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 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 Constants and Types declaration. (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"                    '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

Call cDAPut(DA, 1, 1, 2, "D:2, 1234567890")                 'save the string in Row 1, Col 1, Sheet 2
Call cDAPut(DA, 1, DA.nCols, 2, "D:2, 0987654321")          'save the string in Row 1, Col 500, Sheet 2
Call cDAPut(DA, DA.nRows, 1, 2, "D:2, 12345ABCDE")          'save the string in Row 500, Col 1, Sheet 2
Call cDAPut(DA, DA.nRows, DA.nCols, 2, "D:2, VWXYZ54321")   'save the string in Row 500, Col 500, Sheet 2

Var(1) = cDAGet(DA, 1, 1, 1)                                'read the string in Row 1, Col 1, Sheet 1
Var(2) = cDAGet(DA, 1, DA.nCols, 1")                        'read the string in Row 1, Col 500, Sheet 1
Var(3) = cDAGet(DA, DA.nRows, 1, 1)                         'read the string in Row 500, Col 1, Sheet 1
Var(4) = cDAGet(DA, DA.nRows, DA.nCols, 1)                  'read the string in Row 500, Col 500, Sheet 1

Var(5) = cDAGet(DA, 1, 1, 2)                                'read the string in Row 1, Col 1, Sheet 2
Var(6) = cDAGet(DA, 1, DA.nCols, 2)                         'read the string in Row 1, Col 500, Sheet 2
Var(7) = cDAGet(DA, DA.nRows, 1, 2)                         'read the string in Row 500, Col 1, Sheet 2
Var(8) = cDAGet(DA, DA.nRows, DA.nCols, 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                                             'internal header size
DA.Signature   = "MCR_347"                                  'internal signature
DA.nFilename = "c:\t2w_tmp\dastring.tmp"                    'name fo the file
DA.nType = 50                                               'fixed string of 50 chars
DA.nRows = 500                                              '500 rows
DA.nCols = 500                                              '500 cols
DA.nSheets = 2                                              '2 sheets
DA.rHandle = 0                                              'internal handle
DA.rElementSize = 50                                        'internal size of a element
DA.rFileSize = 25000128                                     'internal size of the file
DA.rParts = 762                                             'internal number of parts (block of 32768
chars)
DA.rRemain = 30784                                          'internal remain chars
DA.rSheetSize = 250000                                      'internal size of one sheet
DA.rTime = 26639                                            '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) {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) {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" (MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal DeleteFile As Integer)

Call Syntax :

Call cMDAClose(MULTIPLEDISKARRAY, DeleteFile%)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
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) {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 functionnality 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 functionnality 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 error code.

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 functionnality 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)                         'clear the last Col in Sheet 2 in the
big sized array on disk
ErrCode = cMDAsClearCol(1, MDA, MDA.nCols(1))                           'clear the last Col in Sheet 1 in the
big sized array on disk

ErrCode = cMDAClearRow(1, MDA, MDA.nRows(1), 2)                         'clear the last Row in Sheet 2 in the
big sized array on disk
ErrCode = cMDAsClearRow(1, MDA, MDA.nRows(1), 1)                        'clear the last Row in Sheet 1 in the
big sized array on disk
```

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 :

MDAClear	clear a multiple big sized array.
MDAClearCol	clear a single col on on a sheet in a multiple big sized array.
MDAClearRow	clear a single row on a sheet in a multiple big sized array.
MDAClearSheet	clear a single sheet in a multiple big sized array.
MDAClose	close a big sized array and keep it or close a multiple big sized array and destroy it.
MDACreate	create a new big sized array on disk or use an existing multiple big sized array on disk.
MDAGet	read an element from a multiple big sized array on disk.
MDAGetType	read a type'd variable from a multiple big sized array on disk.
MDAPut	save an element to a multiple big sized array on disk.
MDAPutType	save a type'd variable to a multiple big sized array on disk.
MDArGet	read an element from a multiple big sized array on disk with only one sheet and one row.
MDArGetType	read a type'd variable from a big sized array on disk with only one sheet and one row.
MDArPut	save an element to a multiple big sized array on disk with only one sheet and one row.
MDArPutType	save a type'd variable to a multiple big sized array on disk with only one sheet and one row.
MDAsClearCol	clear a single col on on a sheet in a multiple big sized array with only one sheet.
MDAsClearRow	clear a single row on a sheet in a multiple big sized array with only one sheet.
MDAsGet	read an element from a multiple big sized array on disk with only one sheet.
MDAsGetType	read a type'd variable from a multiple big sized array on disk with only one sheet.
MDAsPut	save an element to a multiple big sized array on disk with only one sheet.
MDAsPutType	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 MDAGetType, 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) {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" (MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal CreateOrUse As Integer) As Integer

Call Syntax :

ErrCode% = cMDACreate(MDA, CreateOrUse%)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
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 Constants and Types declaration. (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 Constants and Types declaration. (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 .nIsTyped 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.nIsTyped(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.nIsTyped(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)                'read the string in Row 500, Col 1,
Sheet 1, Array 1
Var(4) = cMDAGet(1, MDA, MDA.nRows(1), MDA.nCols(1), 1)     'read the string in Row 500, Col
500, Sheet 1, Array 1

Var(5) = cMDAGet(9, MDA, 1, 1, 5)                           'read the string in Row 1, Col 1,
Sheet 5, Array 9
Var(6) = cMDAGet(9, MDA, 1, MDA.nCols(9), 5)                'read the string in Row 1, Col 100,
Sheet 5, Array 9
Var(7) = cMDAGet(9, MDA, MDA.nRows(9), 1, 5)                'read the string in Row 100, Col 1,
Sheet 5, Array 9
Var(8) = cMDAGet(9, MDA, MDA.nRows(9), MDA.nCols(9), 5)     'read the string in Row 100, Col
100, Sheet 5, Array 9

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) {Win95/WinNT}, MSOffice 95

Purpose :

MDAGet read an element from a multiple big sized array on disk.
MDArGet have the same functionnality but with a multiple big sized array with only one sheet and only one row.
MDAsGet have the same functionnality 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) {Win95/WinNT}, MSOffice 95

Purpose :

MDAPut save an element to a big sized array on disk.
MDArPut have the same functionnality but with a multiple big sized array with only one sheet and only one row.
MDAsPut have the same functionnality 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) {Win95/WinNT}, MSOffice 95

Purpose :

MDAGetType read a type'd variable from a multiple big sized array on disk.
MDArGetType have the same functionnality but with a multiple big sized array with only one sheet and only one row.
MDAsGetType have the same functionnality 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\datype.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)                              'use a created multiple big sized array on
disk

Call cDAGetType(1, MDA, 1, 1, 1, TE(1))                       'read the type'd variable in Row 1, Col 1,
Sheet 1, Array 1.
Call cDAGetType(1, MDA, 1, DA.nCols(1), 1, TE(2))             'read the type'd variable in Row 1, Col 500,
Sheet 1, Array 1.
Call cDAGetType(1, MDA, MDA.nRows(1), 1, 1, TE(3))            'read the type'd variable in Row 500, Col 1,
Sheet 1, Array 1.
Call cDAGetType(1, MDA, MDA.nRows(1), MDA.nCols(1), 1, TE(4)) 'read the type'd variable in Row 500, Col
500, Sheet 1, Array 1.
```

See also : Multiple Disk Array, MDAPutType

MDAPutType, MDAsPutType, MDArPutType

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

Purpose :

MDAPutType save a type'd variable from a big sized array on disk.
MDArPutType have the same functionnality but with a big sized array with only one sheet and only one row.
MDAsPutType have the same functionnality 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) {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, 1representation

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")                             -> "cmccmmcmcmccccmm"

test$ = cFromBinary("0100110101000011")                   -> "MC"
test$ = cFromBinary2("cmccmmcmcmccccmm","mc")  -> "MC"
```

See also : Binary

2-D Geometry : Overview

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

Purpose :

V2Add add two 2D vectors.
V2Sub substract 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 cV2LinearIp 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) {Win95/WinNT}, MSOffice 95

Purpose :

V3Add add two 3D vectors.
V3Sub substract 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 cV3LinearIp 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" (ul3D As tagVECTOR3, ll3D As tagVECTOR3, lr3D 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 lRight3D                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

    lRight3D.x = 17
    lRight3D.y = 15
    lRight3D.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, lRight3D, uRight3D, ptToLocate3D)          '->
    24,0609108270454

    ptToLocate3D.x = 15
    ptToLocate3D.y = 19
    ptToLocate3D.z = 0

    Debug.Print c3DWeightAverage(uLeft3D, lLeft3D, lRight3D, 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) {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.

CheckWait	check if the specified timer has reached the time to wait.
ReadBasisTimer	read the value of the default timer.
ReadTimer	read the value of the specified timer.
SetWait	set the time to wait in a specified timer.
Sleep	suspend the current execution of a routine for a gived delay.
StartBasisTimer	start the default timer.
StartTimer	start the specified timer.
StartWait	start the specified timer.
StopBasisTimer	stop the value of the default timer.
StopTimer	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.

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.

Sleep

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

Purpose :

Sleep suspend the current execution of a routine for a gived 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) {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) {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) {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 is 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
```

IEEEnum : Overview

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.

CVB
CVC
CVD
CVI
CVL
CVS

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.

MKB
MKC
MKD
MKI
MKL
MKN
MKS

Binary : Overview

B2I	convert a binary string into an integer variable.
B2L	convert a binary string into a long variable.
CreateBits	create a string which containes how many bits specified by a number.
FindBitReset	find the first bit Reset starting at the position gived for a a gived string.
FindBitSet	find the first bit Set starting at the position gived for a a gived string.
FromBinary	convert a binary string (0, 1) to a string
FromBinary2	convert a binary string (custom letters) to a string
FromHexa	convert a hexa string to an ascii string.
GetBit	return if a gived bit in a gived string if Set or Reset.
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.
H2I	convert a hexa string into an integer variable.
H2L	convert a hexa string into a long variable.
IsBitPalindrome	check if a string is Bit palindrome.
ReverseAllBits	reverse all bits in a gived string.
ReverseAllBitsByChar	reverse all bits by each char in a gived string.
SetAllBits	set all bits of a gived string to Set state or Reset state.
SetBit	set a gived bit in a gived string to Set state or Reset state.
SetBitToFalse	set a gived bit in a gived string to Reset state.
SetBitToTrue	set a gived bit in a gived string to Set state.
ToBinary	convert a string to a binary representation with 0, 1
ToBinary2	convert a string to a binary representation with two custom letters for 0, 1representation
ToggleAllBits	toggle all bits in a gived 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 gived bit in a gived 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.
ToHexa	convert a ascii string to hexa string.

CVx

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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.

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

Call Syntax :

test% = cCVB(Value$)
test@ = cCVC(Value$)
test# = cCVD(Value$)
test% = cCVI(Value$)
test& = cCVL(Value$)
test! = cCVS(Value$)

Where :

test? receives the value represented by the IEEE string held in Value$

Comments :

See also : MKx

MKx

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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.

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 cMKN Lib "time2win.dll" (ByVal Value As String) As String

Call Syntax :

Nm$ = cMKB(Value%)
Nm$ = cMKC(Value@)
Nm$ = cMKD(Value#)
Nm$ = cMKI(ValueM)
Nm$ = cMKL(Value&)
Nm$ = cMKS(Value!)

Nm$ = cMKN(Value$)

Where :

Nm$ receives the IEEE 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) {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("ABCDEFG")                    -> "41424344454647"
test$ = cFromHexa("47464544434241")           -> "GFEDCBA"
```

See also : Binary

B2I, B2L, H2I, H2L

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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.

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
```

Call Syntax :

```
Test% = cB2I(txtBinary$)
Test& = cB2L(txtBinary$)
Test% = cH2I(txtHexa$)
Test& = cH2L(txtHexa$)
```

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)

Comments :

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("1111111111111111")                   ' -> -1
Debug.Print cB2I("0101010101010101")                   ' -> 21845
Debug.Print cB2I("1010101010101010")                   ' -> -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 cH2I("0")                                  ' -> 0
Debug.Print cH2I("A1")                                 ' -> 161
Debug.Print cH2I("A1B")                                ' -> 2587
Debug.Print cH2I("7FFF")                               ' -> 32767
Debug.Print cH2I("A1B2")                               ' -> -24142
Debug.Print cH2I("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
```

See also : Binary

SetAllBits, SetBit, SetBitToFalse, SetBitToTrue

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

Purpose :

SetAllBits set all bits of a gived string to Set state or Reset state.
SetBit set a gived bit in a gived string to Set state or Reset state.
SetBitToFalse set a gived bit in a gived string to Reset state.
SetBitToTrue set a gived bit in a gived 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) {Win95/WinNT}, MSOffice 95

Purpose :

FindBitReset find the first bit Reset starting at the position gived for a a gived string.
FindBitSet find the first bit Set starting at the position gived for a a gived 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) {Win95/WinNT}, MSOffice 95

Purpose :

ToggleAllBits toggle all bits in a gived 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 gived bit in a gived 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) {Win95/WinNT}, MSOffice 95

Purpose :

ReverseAllBits reverse all bits in a gived string.
ReverseAllBitsByChar reverse all bits by each char in a gived 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) {Win95/WinNT}, MSOffice 95

Purpose :

IsBitPalindrome check if a string is Bit palindrome.

Declare Syntax :

Declare Function cIsBitPalindrome Lib "time2win.dll" (Txt As String) As Integer

Call Syntax :

test = cIsBitPalindrome(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) {Win95/WinNT}, MSOffice 95

Purpose :

CreateBits create a string which containes 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) {Win95/WinNT}, MSOffice 95

Purpose :

GetBit return if a gived bit in a gived string if 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) {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) {Win95/WinNT}, MSOffice 95

Purpose :

Get extract a sub-string delimited by **'|'** in a gived string.
GetBlock read a block of n chars starting at a gived block in a gived string.
GetIn extract a left sub-string delimited by a separator in a gived string.
GetInPart extract the first left sub-string or the rest after the first sub-string delimited by a separator in a gived string.
GetInPartR extract the first right sub-string or the rest after the first sub-string delimited by a separator in a gived string.
GetInR extract a right sub-string delimited by a separator in a gived string.
TokenIn extract a sub-string delimited by a separator's list in a gived string.

Declare Syntax :

Declare Function cGet Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cGetBlock Lib "time2win.dll" (Txt As String, ByVal Position As Integer, ByVal Length As Integer) As String
Declare Function cGetIn Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetInPart Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetInPartR Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetInR Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cTokenIn Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Integer) 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 delimitor 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

AddDigit	sum all numerics chars in a gived string.
Align	align a give string (left, center, right) into an another new string.
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.
ArabicToRoman	convert an integer or a long integer into Roman representation.
BlockCharFromLeft	read n chars from the left of a string.
BlockCharFromRight	read n chars from the right of a string.
ChangeChars	change all chars specifien by others chars in a string.
ChangeCharsUntil	change all chars specifien by others chars in a string until a char is encountered.
CheckChars	verify that all chars specifien are present in a string.
CheckNumericity	check if a string is a numeric string.
CnvASCIItoEBCDIC	convert an ASCII string into EBCDIC equivalent.
CnvEBCDICtoASCII	convert an EBCDIC string into ASCII equivalent.
Compact	compact a string composed of numeric chars.
Compress	remove all chr$(0):ASCII NULL, chr$(9):TAB, chr$(32):SPACE from a string.
CompressTab	pack all n space chars into a tab char.
CplAlpha	return the complementary string from a gived string composed with ascii chars.
CplDigit	return the complementary string from a gived string composed with numerics chars.
Count	count the number of a specified char in a string.
CreateAndFill	create a string with the specified size and fill it with some chars.
ExpandTab	unpack all tab chars into n space chars.
Fill	fill a string with some chars.
FilterBlocks	remove one or more sub-string separated by two delimitors in a gived string.
FilterChars	remove some chars specifien in a gived string.
FilterFirstChar	remove some chars beginning at first position of a gived string.
FilterNotChar	remove all chars except speficien chars in a gived string.
Get	extract a sub-string delimited by '|' in a gived string.
GetBlock	read a block of n chars starting at a gived block in a gived string.
GetIn	extract a left sub-string delimited by a separator in a gived string.
GetInPart	extract the first left sub-string or the rest after the first sub-string delimited by a separator in a gived string.
GetInPartR	extract the first right sub-string or the rest after the first sub-string delimited by a separator in a gived string.
GetInR	extract a right sub-string delimited by a separator in a gived string.
InsertBlocks	insert different block of char in a gived string separated by '~'.
InsertBlocksBy	insert different block of char in a gived string separated by a gived separator.
InsertByMask	replace the specified char by a string in a gived string.
InsertChars	insert a string starting at a gived position in a gived string.
Lrc	calculate the LRC of a gived string.
MixChars	will mix all chars in a gived string in a random position.
Morse	convert a string to a morse string.
NumDigit	sum and sums all numerics chars in a gived string to have a maximum value of 9.
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.
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.
PatternExtMatch	search if a gived pattern can be found is a gived string.
PatternMatch	search if a gived pattern can be found is a gived string.
ProperName	convert the first letter of each word separated by a space in a string to upper case.
ProperName2	convert the first letter of some words separated by a space or punctuation in upper letter case.
RemoveBlockChar	remove a block of chars at the specified position in a string.
RemoveOneChar	remove one char at the specified position in a string.
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.
Reverse	reverse all chars in a gived string.
RomanToArabic	convert a Roman string into an integer or a long integer.
ScrollL	scroll one char to the left of a specified string.
ScrollR	scroll one char to the right of a specified string.

StringSAR	search and replace a string by an another in the specified string.
TokenIn	extract a sub-string delimited by a separator's list in a gived string.
Uncompact	uncompact a string composed of numeric chars.

BlockCharFromLeft, BlockCharFromRight, OneCharFromLeft, OneCharFromRight

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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, InsertBlocksBy, InsertByMask, InsertChars

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

Purpose :

InsertBlocks insert different block of char in a gived string separated by '~'.
InsertBlocksBy insert different block of char in a gived string separated by a gived separator.
InsertByMask replace the specified char by a string in a gived string.
InsertChars insert a string starting at a gived position in a gived string.

Declare Syntax :

```
Declare Function cInsertBlocks Lib "time2win.dll" (Txt As String, Insert As String) As String
Declare Function cInsertBlocksBy Lib "time2win.dll" (Txt As String, Insert As String, Delimitor 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$ = cInsertBlocksBy(Txt, Insert, Delimitor)
test$ = cInsertByMask(Txt, Mask, Insert)
test$ = cInsertChars(Txt, Position, Insert)
```

Where :

Txt	the string to proceed
Insert	the string to insert
Delimitor	the delimitor 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 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$ = cInsertBlocksBy("U/VW/XYZ", "a/bc", "/")             ' "UaVWbcXYZ"

test$ = cInsertByMask("Nr ## Price $###.##", "#", "0705200") ' "Nr 07 Price $052.00"

test$ = cInsertChars("ABCDEFG", 3, "wxyz")                   ' "ABCwxyzDEFG"
test$ = cInsertChars("ABCDEFG", 90, "wxyz")                  ' "ABCDEFGwxyz"
test$ = cInsertChars("ABCDEFG", 0, "wxyz")                   ' "wxyzABCDEFG"
```

See also : String

AndToken, AndTokenIn, OrToken, OrTokenIn

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

Purpose :

FilterBlocks remove one or more sub-string separated by two delimitors in a gived string.
FilterChars remove some chars specifien in a gived string.
FilterFirstChar remove some chars beginning at first position of a gived string.
FilterNotChar remove all chars except speficien chars in a gived string.

Declare Syntax :

Declare Function cFilterBlocks Lib "time2win.dll" (Txt As String, Delimitor 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, Delimitor)
test = cFilterChars(Txt, charSet)
test = cFilterFirstChars(Txt, charSet)
test = cFilterNotChars(Txt, charSet)

Where :

Txt	the string to proceed
Delimitor	two chars for filter the string
charSet	the chars for filter the string
test	the result

Comments :

Examples :

```
Txt = "A/BC/DEF/GHIJ"
Delimitor = "//"
test = cFilterBlocks(Txt, Delimitor)
        ' test = "ADEF"
```

```
Txt = "A/BC/DEF/GHIJ"
Delimitor = "BI"
test = cFilterBlocks(Txt, Delimitor)
        ' test = "A/J"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "B/"
test = cFilterChars(Txt, charSet)
        ' test = "ACDEFGHIJ"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "AF/"
test = cFilterChars(Txt, charSet)
        ' test = "BCDEGHIJ"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = A/"
test = cFilterFirstChars(Txt, charSet)
        ' test = "BC/DEF/GHIJ"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "A/BC/"
test = cFilterFirstChars(Txt, charSet)
        ' test = "DEF/GHIJ"
```

```
Txt = "A/BC/DEF/GHIJ"
charSet = "B/"
test = cFilterNotChars(Txt, charSet)
        ' test = "/B//"
```

```
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) {Win95/WinNT}, MSOffice 95

Purpose :

AddDigit sum all numerics chars in a gived string.
CplDigit return the complementary string from a gived string composed with numerics chars.
NumDigit sum and sums all numerics chars in a gived string to have a maximum value of 9.
CplAlpha return the complementary string from a gived 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("12345678909876543217123456789098765432l7") ' 194
test% = cNumDigit("1234567890987654321712345678909876543217")' 5

test$ = cCplDigit("1234567890987654321712345678909876543217") '
"8765432109012345678287654321090123456782"

test% = cAddDigit("8765432109012345678287654321090123456782") ' 166
test% = cNumDigit("8765432109012345678287654321090123456782")' 4

test$ = cCplAlpha("ÀÁÂÃÄÅÆ") ' "?>=<;:9"

See also : String

CnvASCIItoEBCDIC, CnvEBCDICtoASCII

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

Purpose :

CnvASCIItoEBCDIC convert an ASCII string into EBCDIC equivalent.
CnvEBCDICtoASCII convert an EBCDIC string into ASCII equivalent.

Declare Syntax :

```
Declare Sub cCnvASCIItoEBCDIC Lib "time2win.dll" (Txt As String)
Declare Sub cCnvEBCDICtoASCII Lib "time2win.dll" (Txt As String)
```

Call Syntax :

```
Call cCnvASCIItoEBCDIC(Txt$)
Call cCnvEBCDICtoASCII(Txt$)
```

Where :

Txt$	the string to convert

Comments :

Examples :

```
Dim Tmp            As String

Tmp = "A/BC/DEF/GHIJ"

Call cCnvASCIItoEBCDIC(Tmp)
Debug.Print Tmp                                  ' ÁaÂÃaÄÅÆaÇÈÉÑ

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) {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

GetAscTime	retrieve the current date and time in a 26 chars string from a language number.
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.

Get.X.Day, Get.X.Month

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 0 and 6. 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, 0)          ' "Dim"
test$ = cGetLongDay(LNG_FRENCH, 0)           ' "Dimanche"
test$ = cGetShortDay(LNG_FRENCH, 6)          ' "Sam"
test$ = cGetLongDay(LNG_FRENCH, 6)           ' "Samedi"

test$ = cGetShortDay(LNG_DUTCH, 0)           ' "Zon"
test$ = cGetLongDay(LNG_DUTCH, 0)            ' "Zondag"
test$ = cGetShortDay(LNG_DUTCH, 6)           ' "Zat"
test$ = cGetLongDay(LNG_DUTCH, 6)            ' "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) {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) {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 :

See also : Compression

FileCompress, FileExpand

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 file1 As String, ByVal file2 As String) As Long
Declare Function cFileExpand Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String) As Long
```

Call Syntax :

```
Test& = cFileCompress(File1$, File2$)
Test& = cFileExpand(File2$, File1$)
```

Where :

File1$	is the original file.
File2$	is the compressed file.
Test&	<0, an error has occured.
	>=0, the length of the created file.

Comments :

The compression gives the better result on TEXT file.

Examples :

See also : Compression

Is.X

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

Purpose :

These routines checks if the specified string is :

IsAlnum	Alphanumeric ('A'-'Z', 'a'-'z', or '0'-'9')
IsAlpha	Letter ('A'-'Z' or 'a'-'z')
IsAscii	ASCII character (0x00 - 0x7F)
IsCsym	Letter, underscore, or digit
IsCsymf	Letter or underscore
IsDigit	Digit ('0'-'9')
IsISBN	International Standard Book Numbers (ISBNs)
IsLower	Lowercase letter ('a'-'z')
IsPalindrome	the string and the reverse string are the same
IsPunct	Punctuation character
IsSpace	White-space character (0x09 - 0x0D or 0x20)
IsUpper	Uppercase letter ('A'-'Z')
IsXdigit	Hexadecimal digit ('A'-'F','a'-'f', or '0'-'9')
IsBalance	test if the specified balance is a valid balance
IsDate	test if the specified date is a valid date
IsHour	test if the specified hour is a valid hour
IsLeapYear	test if the specified year is a leap year

Declare Syntax :

Declare Function cIsAlnum Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsAlpha Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsAscii Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsCsym Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsCsymf Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsDigit Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsISBN Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsLower Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsPalindrome Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsPunct Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsSpace Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsUpper Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsXDigit Lib "time2win.dll" (Txt As String) As Integer

Declare Function cIsBalance Lib "time2win.dll" (ByVal nHour As Long, ByVal nMinute As Integer, ByVal nSecond As Integer) As Integer
Declare Function cIsDate Lib "time2win.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Integer
Declare Function cIsHour Lib "time2win.dll" (ByVal nHour As Integer, ByVal nMinute As Integer, ByVal nSecond As Integer) As Integer
Declare Function cIsLeapYear Lib "time2win.dll" (ByVal nYear As Integer) As Integer

Call Syntax :

test = cIsAlnum(Txt)
test = cIsAlpha(Txt)
test = cIsAscii(Txt)
test = cIsCsym(Txt)
test = cIsCsymf(Txt)
test = cIsDigit(Txt)
test = cIsLower(Txt)
test = cIsPalindrome(Txt)

test = cIsPunct(Txt)
test = cIsSpace(Txt)
test = cIsUpper(Txt)
test = cIsXdigit(Txt)

test = cIsBalance(nHour, nMinute, nSecond)
test = cIsDate(nYear, nMonth, nDay)
test = cIsHour(nHour, nMinute, nSecond)
test = cIsLeapYear(nYear)

Where :

Txt	the string to proceed
nHour	the hour to test (can be negative and/or greater than 1439 for cIsBalance)
nMinute	the minute to test
nSecond	the second to test
nYear	the year to test
nMonth	the month to test
nDay	the dat to test
test	TRUE if test is OK FALSE if the test fails

Comments :

Examples :

Txt = "ABCDEFG"

test = cIsAlnum(Txt)	TRUE
test = cIsAlpha(Txt)	TRUE
test = cIsAscii(Txt)	TRUE
test = cIsCsym(Txt)	TRUE
test = cIsCsymf(Txt)	TRUE
test = cIsDigit(Txt)	FALSE
test = cIsLower(Txt)	FALSE
test = cIsPalindrome(Txt)	FALSE
test = cIsPunct(Txt)	FALSE
test = cIsSpace(Txt)	FALSE
test = cIsUpper(Txt)	TRUE
test = cIsXdigit(Txt)	FALSE

test = cIsBalance(-1200, 58, 34)	TRUE
test = cIsDate(1995, 2, 29)	FALSE
test = cIsHour(23, 60, 10)	FALSE
test = cIsLeapYear(1996)	TRUE

See also : Is

HMAPutType, HMArPutType, HMAsPutType

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

Purpose :

HMAPutType save a type'd variable from a huge array.
HMArPutType have the same functionnality but with a huge array with only one sheet and only one row.
HMAsPutType have the same functionnality 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)                    ' save the type'd variable in Row 500, Col 1,
Sheet 1
ErrCode = cTasks(TE, False)
Call cHMAPutType(HMA, HMA.nRows, HMA.nCols, 1, TE)            ' save the type'd variable in Row 500, Col
500, Sheet 1
```

See also : Huge memory array

Is : Overview

These routines checks if the specified string is :

IsAlnum	Alphanumeric ('A'-'Z', 'a'-'z', or '0'-'9')
IsAlpha	Letter ('A'-'Z' or 'a'-'z')
IsAscii	ASCII character (0x00 - 0x7F)
IsCsym	Letter, underscore, or digit
IsCsymf	Letter or underscore
IsDigit	Digit ('0'-'9')
IsISBN	International Standard Book Numbers (ISBNs)
IsLower	Lowercase letter ('a'-'z')
IsPalindrome	the string and the reverse string are the same
IsPunct	Punctuation character
IsSpace	White-space character (0x09 - 0x0D or 0x20)
IsUpper	Uppercase letter ('A'-'Z')
IsXdigit	Hexadecimal digit ('A'-'F','a'-'f', or '0'-'9')
IsBalance	test if the specified balance is a valid balance
IsDate	test if the specified date is a valid date
IsHour	test if the specified hour is a valid hour
IsLeapYear	test if the specified year is a leap year

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.
IsFileVolId 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) {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.
IsFileVolId check if the specified file is a VOLUME-ID file.
IsFileFlag check if the specified file have the specified attributes.

Declare Syntax :

```
Declare Function cIsFileArchive Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileEmpty Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileHidden Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFilenameValid Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileNormal Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileReadOnly Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileSubDir Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileSystem Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileVolId Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileFlag Lib "time2win.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer
```

Call Syntax :

```
test% = cIsFileArchive(nFilename)
test% = cIsFileEmpty(nFilename)
test% = cIsFileHidden(nFilename)
test% = cIsFilenameValid(nFilename)
test% = cIsFileNormal(nFilename)
test% = cIsFileReadOnly(nFilename)
test% =cIsFileSubDir(nFilename)
test% = cIsFileSystem(nFilename)
test% = cIsFileVolId(nFilename)
test% = cIsFileFlag(nFilename, nStatus)
```

Where :

nFilename	the filename to check
nStatus	the status to check (only for cIsFileFlag) combine file attributes 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 usen 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 theorical maximum and is depending of any application loaded in memory.

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

HMAClear	Clear a Huge Array (fill it with chr$(0) or chr$(32) (for string array)).
HMAClearCol	Clear a single Col on on one Sheet or on all sheets in a Huge Array (see above).
HMAClearRow	Clear a single Row on one Sheet or on all Sheets in a Huge Array (see above).
HMAClearSheet	Clear a single Sheet in a Huge Array (fill it with chr$(0) or chr$(32) (for string array)).
HMACreate	Create a Huge Array.
HMAFree	Free a Huge Array.
HMAGet	Read an element from a Huge Array.
HMAGetType	Read a type'd variable from a Huge Array.
HMAOnDisk	Get/Put a Huge Array from/to a file on disk.
HMAPut	Save an element to a Huge Array.
HMAPutType	Save a type'd variable to a Huge Array.
HMArGet	Read an element from a Huge Array with only one sheet and one row.
HMArGetType	Read a type'd variable from a Huge Array with only one sheet and one row.
HMArPut	Save an element from a Huge Array with only one sheet and one row.
HMArPutType	Save a type'd variable from a Huge Array with only one sheet and one row.
HMAsClearCol	Clear a single Col in a Huge Array with only one sheet.
HMAsClearRow	Clear a single Row in a Huge Array with only one sheet.
HMAsGet	Read an element from a Huge Array with only one sheet.
HMAsGetType	Read a type'd variable from a Huge Array with only one sheet.
HMAsPut	Save an element from a Huge Array with only one sheet.
HMAsPutType	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) {Win95/WinNT}, MSOffice 95

Purpose :

HMAPut save an element to a huge array.
HMArPut have the same functionnality but with a huge array with only one sheet and only one row.
HMAsPut have the same functionnality 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) {Win95/WinNT}, MSOffice 95

Purpose :

HMAGetType read a type'd variable from a huge array.
HMArGetType have the same functionnality but with a huge array with only one sheet and only one row.
HMAsGetType have the same functionnality 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))          ' read the type'd variable in Row 500, Col
500, Sheet 1
```

See also : Huge memory array

HMAGet, HMArGet, HMAsGet

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

Purpose :

HMAGet read an element from a huge array.
HMArGet have the same functionnality but with a huge array with only one sheet and only one row.
HMAsGet have the same functionnality 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) {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) {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 Constants and Types declaration. (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, HMAClearSheet, HMAClearCol, HMAsClearCol, HMAClearRow, HMAsClearRow

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

Purpose :

HMAClear clear a huge array (fill it with chr$(0) or chr$(32) (for string array)).
HMAClearSheet clear a single Sheet in a huge array (fill it with chr$(0) or chr$(32) (for string array)).
HMAClearCol 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)).
HMAsClearCol have the same functionnality but with a huge array with only one sheet.
HMAClearRow 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)).
HMAsClearRow have the same functionnality but with a huge array with only one sheet.

Declare Syntax :

Declare Function cHMAClear Lib "time2win.dll" (HMA As tagHMA) As Integer
Declare Function cHMAClearSheet Lib "time2win.dll" (HMA As tagHMA, ByVal Sheet As Long) As Integer
Declare Function cHMAClearCol Lib "time2win.dll" (HMA As tagHMA, ByVal Col As Long, ByVal Sheet As Long) As Integer
Declare Function cHMAsClearCol Lib "time2win.dll" (HMA As tagHMA, ByVal Col As Long) As Integer
Declare Function cHMAClearRow Lib "time2win.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Sheet As Long) As Integer
Declare Function cHMAsClearRow Lib "time2win.dll" (HMA As tagHMA, ByVal Row As Long) As Integer

Call Syntax :

ErrCode% = cHMAClear(HMA)
ErrCode% = cHMAClearSheet(HMA, Sheet&)
ErrCode% = cHMAClearCol(HMA, Col&, Sheet&)
ErrCode% = cHMAsClearCol(HMA, Col&)
ErrCode% = cHMAClearRow(HMA, Row&, Sheet&)
ErrCode% = cHMAsClearRow(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 cHMAClearSheet :

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 functionnality 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.

For cHMAClearCol, cHMAsClearCol :

If the Col is below 1, the Col 1 is used.
If the Col is greater than HMA.nCols, the Col HMA.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 HMA.nSheets, the Sheet HMA.nSheets is used.

For cHMAClearRow, cHMAsClearRow :

If the Row is below 1, the Row 1 is used.
If the Row is greater than HMA.nRows, the Row HMA.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 HMA.nSheets, the Sheet HMA.nSheets is used.

Examples :

```
Dim ErrCode                  As Integer
Dim HMA                      As tagHMA

HMA.nType = 50                                                   ' positive value for a string
HMA.nIsTyped = False                                             ' init the array with spaces
HMA.nRows = 500                                                  ' 500 rows
HMA.nCols = 500                                          ' 500 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 500, Sheet 1
Call cHMAPut(HMA, HMA.nRows, 1, 1, "D:1, OPQRSTUVWXYZ")          ' save the string in Row 500, Col 1, Sheet 1
Call cHMAPut(HMA, HMA.nRows, HMA.nCols, 1, "D:1, oprqstuvwxyz")  ' save the string in Row 500, Col 500, Sheet
1

'.......... some codes

ErrCode = cHMAClear(HMA)                                         ' clear all elements in the huge array

ErrCode = cHMAClearSheet(HMA, 2)                                 'clear the Sheet 2 in the huge array

ErrCode = cHMAClearCol(HMA, HMA.nCols, 2)                        ' clear the last Col in Sheet 2 in the huge
array
ErrCode = cHMAsClearCol(HMA, HMA.nCols)                          ' clear the last Col in Sheet 1 in the huge
array

ErrCode = cHMAClearRow(HMA, HMA.nRows, 2)                        ' clear the last Row in Sheet 2 in the huge
array
ErrCode = cHMAsClearRow(HMA, HMA.nRows)                          ' clear the last Row in Sheet 1 in the huge
array
```

See also : Huge memory array

HMAOnDisk

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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

Decrypt	decode a string encoded with Encrypt function.
Encrypt	encode a string with a password/key.
FileEncrypt	copy one file to an another file but with encryption.
FileDecrypt	copy one file to an another file but with decryption.

Encrypt, Decrypt

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 gived 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) {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 occured.

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 gived 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 gived file.
StringCRC32	calculate a 32 bits CRC for a gived string.

Crypt, FileCrypt

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

Purpose :

Crypt encrypt/decryt 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 occured
	> 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 " & IIf(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-decrypted"

    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 gived 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 gived 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 (&hFFFFFFFF).

Examples :

```
test = cStringCRC32("ABCDEFG")          ' &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/decryt 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.

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.

IsSerial, SerialGet, SerialInc, SerialPut, SerialRmv

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 cIsSerial 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% = cIsSerial(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 (tagSERIALDATA).
Increment&	is the increment (positive or negative).
Test%	TRUE if all is ok, <> TRUE if an error has occured.

Comments :

For 16-Bit :

The length of the serialization string is maximum 50 characters (SERIALDATA.Description1, SERIALDATA.Description2).

For 32-Bit :

The length of the serialization string is maximum 52 characters (SERIALDATA.Description1, SERIALDATA.Description2).

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 = "ABCDEFGHIJKLMNOPQRSTUVWYZ"
        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) {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) {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" + char$(9) +
chr$(9) + "D"

Txt = test = "A" + chr$(9) + "B" + chr$(9) + space$(1) + "C" + char$(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) {Win95/WinNT}, MSOffice 95

Purpose :

ChangeChars change all chars specifien by others chars in a string.
ChangeCharsUntil change all chars specifien 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) {Win95/WinNT}, MSOffice 95

Purpose :

CheckChars verify that all chars specifien 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 specifien in charSet are present in Txt.
	FALSE if all chars specifien in charSet are not present in Txt.

Comments :

Examples :

Txt = "ABCDEFG"
charSet = "CAD"
status = cCheckChars(Txt, charSet)' status = TRUE

Txt = "ABCDEFG"
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) {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) {Win95/WinNT}, MSOffice 95

Purpose :

Reverse reverse all chars in a gived 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) {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) {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) {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 WINtimetimeti"
```

See also : String

SwapD, SwapI, SwapL, SwapS, SwapStr

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

Purpose :

Lrc calculate the LRC of a gived 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) {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/uncompacted.

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) {Win95/WinNT}, MSOffice 95

Purpose :

MixChars will mix all chars in a gived 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) {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) {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) {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) {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.
Valeu&	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 cDecrI(Value)          ' 3
```

See also : Miscellaneous

StringSAR

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 file.

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

Miscelleanous : Overview

BaseConversion	convert a number string (long integer) from a radix to another radix.
Between	check to see if a value is between two other values.
Combination	compute C(n,m) which is the number of combinations of n items, taken m at a time.
DecrI	auto-decrement an integer value by 1.
DecrL	auto-decrement a long value by 1.
Fraction	return a value into the form of a fraction.
IncrI	auto-increment an integer value by 1.
IncrL	auto-increment a long value by 1.
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).
SpellMoney	spell money value with hundredth.
TrueBetween	check to see if a value is fully between two other values.

PatternMatch

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

Purpose :

PatternMatch search if a gived pattern can be found is a gived string.

Declare Syntax :

Declare Function cPatternMatch Lib "time2win.dll" (ByVal Txt As String, ByVal Pattern As String) As Integer

Call Syntax :

test% = cPatternMatch(Txt, Pattern)

Where :

Txt	the string to proceed
Pattern	the pattern to match
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.
The matching of all chars (not '?', '*') is case-sensitive.

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
```

See also : String

PatternExtMatch

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

Purpose :

PatternExtMatch search if a gived pattern can be found is a gived string.

Declare Syntax :

Declare Function cPatternExtMatch Lib "time2win.dll" (ByVal Txt As String, ByVal Pattern As String) As Integer

Call Syntax :

test% = cPatternExtMatch(Txt, Pattern)

Where :

Txt	the string to proceed
Pattern	the pattern to match
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).
The matching of all others chars is case-sensitive.

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
PATTERN_RANGE	malformed range in [..] construct
PATTERN_CLOSE	no end bracket in [..] construct
PATTERN_EMPTY	[..] contstruct is empty
PATTERN_INTERNAL_ERROR	internal error
PATTERN_MATCH	bad hexa in ~xy

Examples :

```
Dim Txt                         As String
```

```
Txt = "Under the blue sky, the sun lights"

test% = cPatternExtMatch(Txt, "*")                                          ' is TRUE
test% = cPatternExtMatch(Txt, "*??*???*?")                                  ' is TRUE
test% = cPatternExtMatch(Txt, "*Under*")                                    ' is TRUE
test% = cPatternExtMatch(Txt, "*sky*")                                      ' is TRUE
test% = cPatternExtMatch(Txt, "*lights")                                    ' is TRUE
test% = cPatternExtMatch(Txt, "Under*")                                     ' is TRUE
test% = cPatternExtMatch(Txt, "??der*sky*ligh??")                           ' is TRUE
test% = cPatternExtMatch(Txt, "Under?the * s?? *")                          ' is TRUE
test% = cPatternExtMatch(Txt, "[U-U][a-z][a-z][a-z][a-z]?the *")            ' is TRUE
test% = cPatternExtMatch(Txt, "[U-U][!A-Z][^A-Z][^A-Z][!A-Z]?the *[s-s]")   ' is TRUE
test% = cPatternExtMatch(Txt, "~55~6E*~73")                                 ' is TRUE
test% = cPatternExtMatch(Txt, "[Uu][Nn][dD][eE][opqrst]?the *[rstu]")       ' is TRUE
test% = cPatternExtMatch(Txt, "Under?the *[~72~73~74~75]")                  ' is TRUE

test% = cPatternExtMatch(Txt, "*under*")                                    ' is MATCH_ABORT
test% = cPatternExtMatch(Txt, "Under*sun")                                  ' is MATCH_ABORT
test% = cPatternExtMatch(Txt, "Under t??e*")                                ' is MATCH_LITERAL
test% = cPatternExtMatch(Txt, "[U-U][!a-z][^A-Z][^A-Z][!A-Z]?the *[!s-s]")  ' is MATCH_RANGE
test% = cPatternExtMatch(Txt, "~55~6G*~73")                                 ' is MATCH_HEXA
test% = cPatternExtMatch(Txt, "[Uu][Nn][dD][eE][opqrst]?the *[rStu]")       ' is MATCH_ABORT
test% = cPatternExtMatch(Txt, "Under?the *[~72~53~74~75]")                  ' is MATCH_ABORT
```

See also : String

```
' definition for error type for PATTERNMATCHEXT
Public Const MATCH_HEXA = 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_HEXA = 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) {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) {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) {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) {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.
Valeu&	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) {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) {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) {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) {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) {Win95/WinNT}, MSOffice 95

Purpose :

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.

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 FALSE if the variables to compare are not the same
test$	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) {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) {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 occured

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) {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 : Overview

DBFileCopy	copy a file to an another file and display a dialog box with title, captions, progress bar and cancel button.
MemoryStatus	retrieve the actual state of the memory.
PBFileCopy	copy a file to an another file and display a progress bar a client standard control.
TaskBarAddIcon	add an icon for an application in the tray of the task bar.
TaskBarDeleteIcon	delete the tray icon from an application in the task bar.
TaskBarModifyIcon	modify an icon for an application in the tray of the task bar.

GetRegistry, KillRegistry, PutRegistry

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

Purpose :

GetRegistry return a key setting value from an application's Windows registry entry.
KillRegistry delete a section or key setting from the Windows registry entry.
PutRegistry save or create an application entry in the Windows registry entry.

Declare Syntax :

Declare Function cGetRegistry Lib "time2win.dll" (ByVal lpSection As String, ByVal lpKey As String, ByVal lpDefault As String) As String
Declare Function cPutRegistry Lib "time2win.dll" (ByVal lpSection As String, ByVal lpKey As String, ByVal lpValue As String) As Integer
Declare Function cKillRegistry Lib "time2win.dll" (ByVal lpSection As String, ByVal lpKey As String) As Integer

Call Syntax :

retCode% = cPutRegistry(lpSection$, lpKey$, lpValue$)
retData$ = cGetRegistry(lpSection$, lpKey$, lpDefault$)
retCode% = cKillRegistry(lpSections$, lpKey$)

Where :

lpSection$	string expression containing the name of the section where the key setting is being saved.
lpKey$	string expression containing the name of the key setting being saved.
lpValue$	string expression containing the value that key is being set to.
lpDefault$	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 cPutRegistry("under the fox", "", "no key")                                    ' -1 (RK_NO_ERROR)
Debug.Print cPutRegistry("under the fox", "key1", "test key1")                             ' -1 (RK_NO_ERROR)
Debug.Print cPutRegistry("under the fox", "key2", "test key2")                             ' -1 (RK_NO_ERROR)
Debug.Print cPutRegistry("under the fox\time2win", "ID", "25")                             ' -1 (RK_NO_ERROR)
Debug.Print cPutRegistry("under the fox\time2win", "Name", "MR")                           ' -1 (RK_NO_ERROR)
Debug.Print cPutRegistry("under the fox\time2win", "", "license")                          ' -1 (RK_NO_ERROR)

Debug.Print cPutRegistry("software\The MCR Company\TIME TO WIN for VB 4.0", "", "Code name")    ' -1 (RK_NO_ERROR)
Debug.Print cPutRegistry("software\The MCR Company\TIME TO WIN for VB 4.0", "Name", "James")    ' -1 (RK_NO_ERROR)
Debug.Print cPutRegistry("software\The MCR Company\TIME TO WIN for VB 4.0", "Id", "Donb")       ' -1 (RK_NO_ERROR)
Debug.Print cPutRegistry("software\The MCR Company\TIME TO WIN for VB 4.0", "N°", "007")        ' -1 (RK_NO_ERROR)
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
```

```
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) {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

Rnd	return a double random number between 0.0 and 1.0.
RndD	return a double random number.
RndI	return an integer random number.
RndInit	initialize the random generator.
RndL	return a long random number.
RndS	return a single random number.

Matrix

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 substract 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

AllSubDirectories	retrieve all sub-directories from a specified directory (root or sub-directory).
ChDir	change the directory.
ChDrive	change the drive.
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.
CountDirectories	count the total directory in a specified directory.
CountFiles	count the total files founded in a specified directory.
FileChangeChars	replace all chars in a char set by a new char set.
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.
FileCopy	copy one file to an another file.
FileCopy2	copy one file to an another file.
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.
FileDrive	extract the drive on which the file is present.
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.
FileGetAttrib	retrieves in one call, all attributes of a file.
FileLineCount	count the total number of lines in an ASCII file.
FileMerge	merge two files in one.
FileMove	move/rename a file in the same or in an another directory.
FilePathExists	verify if the specified file is present.
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.
FileSearch	search a string in a gived TEXT file.
FileSearchCount	count.the occurence of a string in a gived TEXT file.
FileSearchAndReplace	search and replace a string by an another in the specified TEXT file.
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 gived file.
FilesInDirectory	retrieve each file in the specified directory.
FilesInDirOnDisk	write all files from a specified directory into a file on disk.
FilesInDirToArray	read all files from a specified directory into an array.
FilesInfoInDir	retrieve each file in the specified directory and returns name, size, scalar date, scalar time, attribute.
FileSize	return the size of the specified file.
FileSort	sort an ASCII file or a BINARY file in ascending or descending order with case sensitive or not.
FilesSize	return the logical size of all files specified by file mask.
FilesSizeOnDisk	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.
FileStatictics	count the lines, words and chars in a specified file.
FileToLower	convert a file to a file with lower case.
FileToUpper	convert a file to a file with upper case.
FullPath	convert a partial path stored in path to a fully qualified path.

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).
GetDriveType	determine whether a disk drive is removable, fixed, or remote.
KillDir	delete the specified empty directory.
KillDirs	delete the specified direcory and its associated directories.
KillDirFilesAll	delete all files specified by a mask in the specified directory and its associated sub-dir.
KillFile	delete the specified filename.
KillFileAll	delete the specified filename with any attribute.
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.
MakeDir	create the specified directory.
MakeMultipleDir	create a multiple directory in one call.
MakePath	create a single path, composed of a drive letter, directory path, filename, and filename extension.
RcsCountFileDir	count the total directories or files in a specified directory (with recursivity or not).
RcsFilesSize	return the logical size of files by file mask in a specified directory (with recursivity or not).
RcsFilesSizeOnDisk	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, the logical size and the physical size (with recursivity or not).
RenameFile	rename a file or moves a file from one path to an other path.
SearchFile	perform a file match starting with a specified path.
SplitPath	break a full path into its four components.
SubDirectory	retrieve all sub-directories from the specified mask.
TruncatePath	truncate a long path with filename.
UniqueFileName	create a unique filename by modifying a gived template argument.

AllSubDirectories

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 occured, > 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) {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) {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) {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 occured.

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) {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 occured.

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) {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 occured.

Comments :

Examples :

See also : File

FileFilter, FileFilterNot

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 occured.

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", "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz")
test& = cFileFilterNot("c:\autoexec.bat", "c:\autoexec.tab", "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz")

See also : File

FileSize

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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 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

FileToLower, FileToUpper

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 occured.

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

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

Purpose :

FileMerge merge two files in one.

Declare Syntax :

Declare Function cFileMerge Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal fileTo As String) As Long

Call Syntax :

test& = cFileMerge(file1, file2, fileTo)

Where :

file1$	is the first file.
file2$	is the second file.
fileTo$	is the destination file.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occured.

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 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 :

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) {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 occured.

Comments :

cFileSearchAndReplace can handle lines with a maximum of 2304 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) {Win95/WinNT}, MSOffice 95

Purpose :

FileSearch search a string in a gived TEXT file.
FileSearchCount count.the occurence of a string in a gived 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 occured.

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 occurence 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) {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 sorting 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 occured.

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) {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 occured.

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) {Win95/WinNT}, MSOffice 95

Purpose :

KillDir delete the specified empty directory.
KillDirs delete the specified direcory 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$)
test% = cKillDirs(lpDir$)

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 occured.

For KillDirs :

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.

This function doesn't generates an VB Error if the speficied dir not exists.

See also : File

KillDirFilesAll

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 :

lpDi$r	is the starting directory
lpMask$	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 occured

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 speficied 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) {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 occured

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 speficied file not exists.

See also : File

KillFiles, KillFilesAll

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 occured

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 speficied files not exists.

See also : File

MakeDir, MakeMultipleDir

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 occured

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 occured)
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) {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 Long
Declare Function cGetDiskSpace Lib "time2win.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDiskUsed Lib "time2win.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDiskClusterSize Lib "time2win.dll" (ByVal lpDrive As String) As Long
```

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).

Examples :

```
test& = cGetDiskFree("C")          ' 268197888
test& = cGetDiskSpace("C")         ' 527654912
test& = cGetDiskUsed("C")' 259457024
test& = cGetDiskClusterSize("C")   ' 8192
```

See also : File

RcsCountFileDir

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 Integer

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 '.' 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

RcsFilesSize, RcsFilesSizeOnDisk, RcsFilesSlack

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

Purpose :

RcsFilesSize return the logical size of files by file mask in a specified directory (with recursivity or not).
RcsFilesSizeOnDisk 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 cRcsFilesSize Lib "time2win.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Long
Declare Function cRcsFilesSizeOnDisk Lib "time2win.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Long
Declare Function cRcsFilesSlack Lib "time2win.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer, Size1 As Long, Size2 As Long) As Integer

Call Syntax :

test& = cRcsFilesSize(FirstDir$, MaskDir$, Recurse%)
test& = cRcsFilesSizeOnDisk(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 founden 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 :

```
test& = cRcsFilesSize("C:\", "*.*", True)              ' on my system, 437,896,805 bytes
test& = cRcsFilesSize("C:\", "*.*", False)             ' on my system,    87,141,863 bytes

test& = cRcsFilesSizeOnDisk("C:\", "*.*", True)        ' on my system, 487,784,448 bytes
test& = cRcsFilesSizeOnDisk("C:\", "*.*", False)       ' on my system,    87,343,104 bytes

test& = cRcsFilesSlack("C:\", "*.*", True, 0, 0)' on my system, 10 %
test& = cRcsFilesSlack("C:\", "*.*", False, 0, 0)      ' on my system,    0%
```

See also : File

FilesSize, FilesSizeOnDisk, FilesSlack

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

Purpose :

FilesSize return the logical size of all files specified by file mask.
FilesSizeOnDisk 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 cFilesSize Lib "time2win.dll" (ByVal lpFilename As String) As Long
Declare Function cFilesSizeOnDisk Lib "time2win.dll" (ByVal nFileName As String) As Long
Declare Function cFilesSlack Lib "time2win.dll" (ByVal nFileName As String, Size1 As Long, Size2 As Long) As Integer

Call Syntax :

test& = cFilesSize(nFilename)
test& = cFilesSizeOnDisk(nFilename)
test% = cFilesSlack(nFilename, Size1, Size2)

Where :

nFilename	is the mask file to proceed.
test&	is the size of all files founden 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 :

```
test& = cFilesSize("*.*")               ' on my system, 5607689 bytes
test& = cFilesSizeOnDisk("*.*")         ' on my system, 5890048 bytes
test% = cFilesSlack("*.*", 0, 0)        ' on my system, 4 %
```

See also : File

CountFiles

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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) {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) {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 'tagSPLITPATH' 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) {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) {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) {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 occured

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 speficied old filename not exists.

Examples :

See also : File

UniqueFileName

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 specifien 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) {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) {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) {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 gived 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 attributes
nFileAttribute	the type variable 'FileAttributeType' (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) {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 attributes
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) {Win95/WinNT}, MSOffice 95

Purpose :

FileDrive extract 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) {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) {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 :

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)
Array()	is the variable array string with one dimension.
test%	>=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) {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 = cFileLastTimeModifed(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) {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) {Win95/WinNT}, MSOffice 95

Purpose :

FileStatictics 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& = cFileStatictics(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 occured.

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) {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

SearchFile

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

Purpose :

SearchFile perform a file match starting with a specified path.

Declare Syntax :

Declare Function cSearchFile Lib "time2win.dll" (ByVal lpStartPath As String, ByVal lpFileMask As String, ByVal lpFileResult As String) As Long

Call Syntax :

lngResult& = cSearchFile(lpStartPath$, lpFileMask$, lpFileResult$)

Where :

lpStartPath$	is the starting path to begin the search.
lpFileMask$	is the file mask to match.
lpFileResult$	is the file with the result of the search (cSearchFile).
lngResult&	is the number of files founded.

Comments :

Examples :

Debug.Print cSearchFile("c:\", "time2win.dll", "c:\tmp\test.sch")

See also : File

CmpFile.X

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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

```
' structure for split path
Type tagSPLITPATH
    nDrive           As String
    nDir             As String
    nName            As String
    nExt             As String
End Type
```

FileGetAttrib

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

Purpose :

FileGetAttrib set in one call, attributes of a gived 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 'FileAttributeType'
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) {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 occured.

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 uucp
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) {Win95/WinNT}, MSOffice 95

Purpose :

FileUUCP uuencode/uudecode a file (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) As Long

Call Syntax :

lngResult& = cFileUUCP(lpFileName1$, lpFileName2$, EncodeDecode%)

Where :

lpFileName1$	is the file to be uuencoded/uudecoded
lpFileName2$	is the file uuencoded/uudecoded
EncodeDecode	is the mode for encoding/decoding
lngResult&	< 0 : an error has occured >= 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) & vbCrLf
strDisplay = strDisplay & "File UUdecode '" & File2 & "' to '" & File3 & "' is " & cFileUUCP(File2, File3,
MODE_UUDECODE) & vbCrLf
strDisplay = strDisplay & "Compare File contents (not sensitive) '" & File1 & "' with '" & File3 & "' is " &
IIf(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) & vbCrLf
strDisplay = strDisplay & "File UUdecode '" & File2 & "' to '" & File3 & "' is " & cFileUUCP(File2, File3,
MODE_UUDECODE) & 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 : UUCP

File I/O from C : Overview

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.
Fflush	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.
Ferror	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).

FileIO

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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.
Fflush 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.
Ferror 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).

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 cFflush 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

Call Syntax :

see above

Where :

File$	the name to use for streaming.
Mode$	the open mode for the file (see comments).
IOstream&	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%	the seeking method (see definition for file I/O in Constants and Types declaration)

Comments :

Code returned by these routines :

Fopen	>= 0	: I/O stream in a long integer.
Fclose	= 0	: all is OK,
	< 0	: error.
Fgetc	>= 0	: the char readed,
	< 0	: error.
Fputc	>= 0	: the char writed,
	< 0	: error.
Fputs	>= 0	: all is OK,
	< 0	: error.
Fgets	= 0	: all is OK,
	< 0	: error.
Fwrite	>= 0	: all is OK,
	< 0	: error.
Fread	>= 0	: all is OK,
	< 0	: error.
Fcloseall	= 0	: all is OK,
	< 0	: error.
Fflush	= 0	: all is OK,
	< 0	: error.
Fflushall	= 0	: all is OK
	< 0	: error.
Feof	= 0	: not EOF,
	= -1	: EOF.
Ferror	= 0	: no error,
	<>0	: error number.
Fseek	= 0	: all is OK,
	< 0	: error.
Ftell	>= 0	: the pointer position,
	< 0	: error.
FProcessAsciiFile	> 0	: the number of lines in the ASCII file (CR/LF terminated),
	= 0	: error : can't allocate memory buffer (each line can't be longer than 16384 characters),
	< 0	: error.
FGotoRecord	= -1	: all is ok,
	= 0	: record is outside of the limits of the array,

< 0	: error.

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 *cFflush*, *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) {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) {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) {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) {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) {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) {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) {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) {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) {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) {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) {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) {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) {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) {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

DisableCtlRedraw	disable the redraw of a object (by Control).
DisableFI	disable mouse and keyboard input to the given form (by Object).
DisableForm	disable mouse and keyboard input to the given form (by hWnd).
DisableRedraw	disable the redraw of a object (by hWnd).
EnableCtlRedraw	enable the redraw of a object (by Control).
EnableFI	enable mouse and keyboard input to the given form (by Object).
EnableForm	enable mouse and keyboard input to the given form (by hWnd).
EnableRedraw	enable the redraw of a object (by hWnd).
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.
GetCtlDataField	return the .DataField property.
GetCtlForm	return the name of the form did contains the control.
GetCtlIndex	return the .Index property.
GetCtlName	return the .Name of the control.
GetCtlNameIndex	return the name and the of the control.
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.
GetCtlTagSized	return the full .Tag property of the control.
GetCtlText	return the .Text property of the control.
GetHwnd	return the .hWnd of the control.
GetObjCaption	
GetObjClassName	
GetObjContainer	
GetObjDataField	
GetObjDataSource	
GetObjIndex	
GetObjName	
GetObjNameIndex	
GetObjParent	
GetObjTag	
GetObjText	
ObjDisableRedraw	disable the redraw of a object (by Object).
ObjEnableRedraw	enable the redraw of a object (by Object).
ObjectGetBoolean	
ObjectGetByte	
ObjectGetIndex	
ObjectGetInteger	
ObjectGetLong	
ObjectGetPropertyByName	read data in properties (by name) from OCX custom controls.
ObjectGetPropertyByPos	read data in properties (by position) from OCX custom controls.
ObjectGetString	
ObjectGetStringW	
ObjectGetVariant	
ObjectMethod	
ObjectMethodByName	give the access of method (by name) of OCX custom controls.
ObjectMethodByPos	give the access of method (by position) of OCX custom controls.
ObjectPutBoolean	
ObjectPutByte	
ObjectPutInteger	
ObjectPutLong	
ObjectPutPropertyByName	write data in properties (by name) from OCX custom controls.
ObjectPutPropertyByPos	write data in properties (by position) in OCX custom controls.
ObjectPutString	
ObjectPutVariant	
PutObjCaption	
PutObjDataField	

PutObjDataSource	
PutObjTag	
PutObjText	
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.

HugeStrOnDisk

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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) {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) {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) {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 usen 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 theorical maximum and is depending of any application loaded in memory.

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

HugeStrAdd	add a VB string into a Huge String.
HugeStrAddress	return a pointer for the first char of a Huge String.
HugeStrAppend	append a VB string into a Huge String.
HugeStrBlocks	return the number of block of 64,000 chars from a Huge String.
HugeStrClear	clear a Huge String.
HugeStrCreate	create a Huge String.
HugeStrFree	free a Huge String (destroy it).
HugeStrGetNP	get the Next Pointer of a Huge String.
HugeStrGetWP	get the Write Pointer of a Huge String.
HugeStrLength	return the length of data in a Huge String.
HugeStrMid	extract a VB sub-string from a Huge String.
HugeStrNext	read the next part of a Huge String.
HugeStrOnDisk	get/put a Huge String from/to a file on disk.
HugeStrRead	read a block of 64,000 chars or minder from a Huge String.
HugeStrSetNP	set the Next Pointer of a Huge String.
HugeStrSetWP	set the Write Pointer of a Huge String.
HugeStrSize	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) {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) {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) {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) {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) {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) {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 containes 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

EnumPrinterJobs enumerate all pending jobs on a printer.

Date and time : Overview

AddTime	retrieve only the part for hours on one day.
CheckTime	verify if an hour (in minutes) is between two others hours (in minutes).
CurrentTime	return the minutes elapsed since midnight.
DateToScalar	compute a scalar from all date parts.
DayOfWeek	calculate the day of the week.
DayOfYear	calculate the day of the year.
DaysInMonth	return the total days in a month.
HourTo	convert a time string to a VARIANT value in minutes (INTEGER or LONG).
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.
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.
IntoDateNull	convert a date value into a date string specified the short date format order in the Control Panel.
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).
ScalarToDate	decompose a scalar date into these components.
ScalarToTime	decompose a scalar time into these components.
TimeBetween	calculate the time (in minutes) between two hours (in minutes).
TimeToScalar	compute a scalar from all time parts.
WeekOfYear	calculate the week of the year.

GetVersion

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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()                ' 2.10
```

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) {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 occured
test$	the returned name of the remote network resource.

Comments :

Examples :

See also : Network

Network : Overview

GetNetConnection return the name of the network resource associated with the specified redirected local device.

LngInpBox

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 :

nLanguage	is the language number.
Message	is the message to display.
Title	is the title of the message box.
Default	is the default string to display in the input part.
Test$	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) {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).
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 functionnality 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) {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 occured

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% = 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	read a generic file (one file per language) which contains the text for supporting a language.
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	create or update a generic file (one file par language) which contains the text for supporting a language.
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) {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) {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 occured

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) {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 occured.

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

ArrayToComboBox	read an string array and append it to a Combo Box.
ArrayToListBox	read an string array and append it to a List Box.
ComboFiles	fill a Combo Box with files with the specified attribute and mask.
ComboSearchFile	perform a file match starting with a specified path and fill a standard combo box.
FileToComboBox	read a file and append it to a Combo Box.
FileToListBox	read a file and append it to a List Box.
ListFiles	fill a List Box with files with the specified attribute and mask.
ListSearchFile	perform a file match starting with a specified path and fill a standard list box.
ListSetTabs	set tabulation in a List Box.

ArrayToComboBox, ArrayToListBox

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 occured.

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) {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 type'd variable to access the drive.
Test%	TRUE, all is ok
	FALSE, no media ID or an error has ocurred.

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) {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 occured.

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 Test As 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) {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 cIntoDate Lib "time2win.dll" (ByVal nDate As Long) As String
Declare Function cIntoDateFill Lib "time2win.dll" (ByVal nDate As Long) As String
Declare Function cIntoDateNull Lib "time2win.dll" (ByVal nDate As Long) As String
```

Call Syntax :

```
test$ = cIntoDate(nDate)
test$ = cIntoDateFill(nDate)
test$ = cIntoDateNull(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$ = cIntoDate(Int(Now))          ' "09/12/1994"
test$ = cIntoDateFill(Int(Now))      ' "09/12/1994"
test$ = cIntoDateNull(Int(Now))      ' "09/12/1994"

test$ = cIntoDate(-1)                ' "29/12/1899"
test$ = cIntoDateFill(-1)            ' "29/12/1899"
test$ = cIntoDateNull(-1)            ' "29/12/1899"

test$ = cIntoDate(0)                 ' "30/12/1899"
test$ = cIntoDateFill(0)             ' "          "
test$ = cIntoDateNull(0)             ' ""

test$ = cIntoDate(1)                 ' "31/12/1899"
test$ = cIntoDateFill(1)             ' "31/12/1899"
test$ = cIntoDateNul(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 Test As 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) {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 spefication

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) {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) {Win95/WinNT}, MSOffice 95

Purpose :

DateToScalar compute a scalar from all date parts.
ScalarToDate decompose a scalar 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 :

Scalar& = cDateToScalar(nYear%, nMonth%, nDay%)
Call cScalarToDate(Scalar&, nYear%, nMonth%, nDay%)

Where :

nYear%	is the year.
nMonth%	is the month.
nDay%	is the day.
Scalar&	is the returned computed scalar.

Comments :

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

Test = cDateToScalar(1995, 3, 25)              ' 728377

Call cScalarToDate(728377, 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) {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) {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) {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 cIntoFixHour Lib "time2win.dll" (Var As Variant, ByVal Length As Integer, ByVal fillZero As Integer, ByVal Hundreds As Integer) As String
Declare Function cIntoHour Lib "time2win.dll" (Var As Variant) As String
Declare Function cIntoVarHour Lib "time2win.dll" (Var As Variant) As String

Call Syntax :

test$ = cIntoFixHour(Var, Length, fillZero, Hundreds)
test$ = cIntoHour(Var)
test$ = cIntoVarHour(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 cIntoFixHour 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) {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 cIntoBalance Lib "time2win.dll" (Var As Variant) As String
Declare Function cIntoBalanceFill Lib "time2win.dll" (Var As Variant) As String

Call Syntax :

test$ = cIntoBalance(Var)
test$ = cIntoBalanceFill(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 formart 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) {Win95/WinNT}, MSOffice 95

Purpose :

CurrentTime return the minutes elapsed since midnight.

Declare Syntax :

Declare Function cCurrentTime Lib "time2win.dll" () As Integer

Call Syntax :

test% = cCurrentTime()

Where :

test%	the minutes

Comments :

Examples :

```
test% = cCurrentTime()                    ' 1234
```

See also : Date and time

Bitmap : Overview

DIBSaveScreen	save the screen (entire desktop) in a file (DIB format).
DIBSaveWindow	save a window in a file (DIB format).
TileBitmapOnWindow	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) {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) {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) {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) {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) {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) {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 occured	

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) {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 occured 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) {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).

Registry key : Overview

GetRegistry	return a key setting value from an application's Windows registry entry.
KillRegistry	delete a section or key setting from the Windows registry entry.
PutRegistry	save or create an application entry in the Windows registry entry.

RegistrationKey, RegistrationKey2, RegistrationKey3

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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.
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) {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 occured

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) {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 = cHashMD5("TIME TO WIN")          ' $Ei"é£,%~"3□ìXA‘
```

See also : Protection

Windows : Overview

ArrangeDesktopIcons	arrange all desktop icons.
CenterWindow	center a window in the screen.
EXEnameActiveWindow	retrieve the full filename (path and file) of the active window.
EXEnameTask	retrieve the full path and filename of the executable file from which the specified module was loaded.
EXEnameWindow	retrieve the full filename (path and file) of the specified window.
ExitWindowsAndExecute	terminate Windows, runs a specified MS-DOS application, and then restarts Windows.
GetClassName	retrieve the full class name of a window.
GetCountry	return the country name.
GetCountryCode	return the country code.
GetCurrency	return the currency.
GetCurrentDrive	return the current default drive.
GetDateFormat	return the format for the date.
GetDateSeparator	return the separator for the date.
GetDefaultPrinter	return the default printer in the [windows] section of Win.INI
GetDevices	return all devices founden in the [devices] section in the Win.INI
GetDriveCurrentDir	retrieve the current dir on the specified drive.
GetHourFormat	return the format for the hour.
GetIni	retrieve an item in a section of an INI file.
GetLanguage	return the letters for the language.
GetListSeparator	return the separator for list.
GetPrinterPorts	return all printers set in the [printerports] section in the Win.INI
GetSectionItems	retrieve all items founden in a section of a specified INI file.
GetSystemDirectory	retrieve the full path of the System directory for Windows.
GetTimeSeparator	return the separator for the date.
GetWindowsDirectory	retrieve the full path for the Windows directory.
GetWinINI	return the information for a gived item.
PutIni	save an item in a section of an INI file.
RebootSystem	reboot your system.
RestartWindows	restart your Windows.
ShowWindow	show a window after an exploded/imploded focus rectangle has been displayed.
WalkThruWindow	walk in the window's list of all windows at a gived moment.

PutIni

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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. If this parameter is NULL, the entire section, including all entries within the section, is deleted.
szDefault	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) {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 gived 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) {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) {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) {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) {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 is 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=dddd 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) {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) {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) {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) {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, TaskBarDeleteIcon, TaskBarModifyIcon

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

Purpose :

TaskBarAddIcon add an icon for an application in the tray of the task bar.
TaskBarDeleteIcon delete the tray icon from an application in the task bar.
TaskBarModifyIcon 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 cTaskBarDeleteIcon Lib "time2win.dll" (ByVal hWnd As Long) As Integer
Declare Function cTaskBarModifyIcon Lib "time2win.dll" (ByVal hWnd As Long, ByVal hIcon As Long, ByVal lpszTip As String) As Integer

Call Syntax :

intResult% = cTaskBarAddIcon(hWnd&, hIcon&, lpszTip$)
intResult% = cTaskBarDeleteIcon(hWnd&)
intResult% = cTaskBarModifyIcon(hWnd&, hIcon&, lpszTip$)

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 cTaskBarDeleteIcon 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 cTaskBarModifyIcon(Me.hWnd, Me.Icon., "Form1 minimized")
```

in the Form_QueryUnload event :

```
debug.print cTaskBarDeleteIcon(Me.hWnd)
```

See also : Windows 95

GetClassName

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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) {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) {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) {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 lpszExe As String, ByVal lpszParams As String) As Integer
Declare Function cRebootSystem Lib "time2win.dll" () As Integer
Declare Function cRestartWindows Lib "time2win.dll" () As Integer

Call Syntax :

test% = cExitWindowsAndExecute(lpszExe, lpszParams)
test% = cRebootSystem()
test% = cRestartWindows()

Where :

lpszExe	is the program to launch after exiting Windows.
lpszParams	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) {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) {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) {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) {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

ComboSearchFile, ListSearchFile

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

Purpose :

ComboSearchFile perform a file match starting with a specified path and fill a standard combo box.
ListSearchFile perform a file match starting with a specified path and fill a standard list box.

Declare Syntax :

Declare Function cListSearchFile 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

Call Syntax :

lngResult& = cListSearchFile(hWnd&, lpStartPath$, lpFileMask$)
lngResult& = cComboSearchFile(hWnd&, lpStartPath$, lpFileMask$)

Where :

lpStartPath$	is the starting path to begin the search.
lpFileMask$	is the file mask to match.
hWnd&	is the .hWnd property of a standard list or combo box.

Comments :

Examples :

debug.print cListSearchFile(List1.hWnd, "c:\", "time2win.dll")
debug.print cComboSearchFile(Combo1.hWnd, "c:\", "time2win.dll")

See also : List box - combo box

ComboFiles, ListFiles

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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

ChangeTaskName	change the name of the task.
GetChangeTaskName	get and change the name of the task.
GetFileVersion	return a partial information over a specified file.
GetFileVersionInfo	return a full information over a specified file in one call.
GetTaskName	read the name of the task.
ModuleFind	retrieve some parameters for a specified loaded module.
Modules	retrieve each loaded module one by one.

GetFileVersion

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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) {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 occured

Comments :

Examples :

```
Dim status                      As Integer
Dim FILEVERSIONINFO   As tagFILEVERSIONINFO

status = cGetFileVersionInfo("k:\windows\system\krnl386.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
    LegalCopyrightAs 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) {Win95/WinNT}, MSOffice 95

Purpose :

WalkThruWindow walk in the window's list of all windows at a gived 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) {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 'tagMODULEENTRY' which receives the parameters.
test%	TRUE if all is Ok FALSE if an error has occured

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

Modules

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

Purpose :

Modules retrieve each loaded module one by one.

Declare Syntax :

Declare Function cModules Lib "time2win.dll" (MODULEENTRY As Any, ByVal firstnext As Integer) As Integer

Call Syntax :

test% = cModules(MODULEENTRY, firstnext)

Where :

MODULEENTRY	is the type'd variable 'tagMODULEENTRY' which receives the parameters.
firstnext	TRUE for the first module FALSE for each next module
test%	TRUE if all is Ok FALSE if an error has occured or if no more modules.

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 i                       As Integer
Dim status                  As Integer
Dim MODULEENTRY             As tagMODULEENTRY

i = 0

Close #1
Open "c:\tmp.tmp" For Output Shared As #1

Print #1, "dwSize"; Chr$(9);
Print #1, "szModule"; Chr$(9);
Print #1, "hModule"; Chr$(9);
Print #1, "wcUsage"; Chr$(9);
Print #1, "szExePath"; Chr$(9);
Print #1, "wNext"; Chr$(13)

status = cModules(MODULEENTRY, True)

Do While (status = True)

   Print #1, MODULEENTRY.dwSize; Chr$(9);
   Print #1, MODULEENTRY.szModule; Chr$(9);
   Print #1, MODULEENTRY.hModule; Chr$(9);
   Print #1, MODULEENTRY.wcUsage; Chr$(9);
   Print #1, MODULEENTRY.szExePath; Chr$(9);
   Print #1, MODULEENTRY.wNext
```

```
    status = cModules(MODULEENTRY, False)

    i = i + 1
    If (i >= 7) Then Exit Do

Loop

Close #1

'On my system, the first 7 modules are :

    ' dwSize          szModule            hModule             wcUsage             szExePath
            wNext
    '   276                 KERNEL              295                 41                  K:\WIN95\SYSTEM\KRNL386.EXE
                    279
    '   276                 SYSTEM              279                 32                  K:\WIN95\SYSTEM\SYSTEM.DRV
                    343
    '   276                 KEYBOARD            343                 31                  K:\WIN95\SYSTEM\
    KEYBOARD.DRV        367
    '   276                 MOUSE               367                 31                  K:\WIN95\SYSTEM\MOUSE.DRV
    RV      463
    '   276                 DISPLAY             463                 32                  K:\WIN95\SYSTEM\SVGA256.DRV
                    487
    '   276                 SOUND               487                 31                  K:\WIN95\SYSTEM\
    MMSOUND.DRV         583
    '   276                 COMM                583                 31                  K:\WIN95\SYSTEM\COMM.DRV
    RV                  1271
```

See also : Task - File version

Tasks

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 occured 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             0                  8263    ICONBAR              8279              4439
 40     4439      1783             4398      4463             4399       5850     1022               5992
        5992              0                  4471    WINEXIT              4447              16279
 40     16279      4231            15878     16295            15879      -4188    -23384    -        10032
        -4054             0                  16255   MSVC                 16271             2087
 40     2087      1783             8030      2095             8031       29198    9004               29334
        29334             0                  8047    FASTLOAD             8063              1783
 40     1783      335              5846      1799             5847       8202     2358               5950
        8304              0                  2079    PROGMAN              791               7007
 40     7007      4231             9926      6767             9927       -23760   13124              23498
        -23562            1                  6879    FOREHELP             6903              4431
 40     4431      1783             4278      4455             4279       7654     2844               6998
        7814              1                  4359    FREEMEM              4375              12127
 40     12127     1783             9022      12143            9023       -29164   16534              -31948
        28672             0                  9039    VB                           9231               0
```

See also : Task - File version

TaskFind

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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 :

hTask	is the task number
TASKENTRY	is the typed variable which receives the parameters 'tagTASKENTRY'
test%	TRUE if all is Ok
	FALSE if an error has occured

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 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
    GlblcntUsage        As Long
    ProccntUsage        As Long
    modBaseAddr         As Byte
    modBaseSize         As Long
    hModule             As Long
    szModule            As String * 256
    szExePath           As String * 260
End Type
```

FilesInfoInDir

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

Purpose :

FilesInfoInDir retrieve each file in the specified directory and returns name, size, scalar date, scalar 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 'tagFILEINFO'.
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 (scalar date)
    fTime           As Long         'time of the file (scalar 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) {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) {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

GetCurrentDrive

QuickInfo : VB 3.0, VB 4.0 (16-Bit), VB 4.0 (32-Bit) {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