Contents

@Blank

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

Examples :

See also :

AddD

Purpose :

AddD adds a constant value to all of the elements of a Double array.

Declare Syntax :

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

Call Syntax :

status = cAddD(array(), nValue)

Where :

array()	is the Double array.
nValue	is the value to add (if positive) or to substract (if negative) to all of the elements of the Double array.

Comments :

See Also : cAddD, cAddI, cAddL, cAddS, Array routines

AddI

Purpose :

AddI adds a constant value to all of the elements of an Integer array.

Declare Syntax :

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

Call Syntax :

status = cAddI(array(), value)

Where :

array()	is the Integer array.
nValue	is the value to add (if positive) or to substract (if negative) to all of the elements of the Integer array.

Comments :

See Also : cAddD, cAddI, cAddL, cAddS, Array routines

AddL

Purpose :

AddL adds a constant value to all of the elements of a Long array.

Declare Syntax :

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

Call Syntax :

status = cAddL(array(), value)

Where :

array()	is the Long array.
nValue	is the value to add (if positive) or to substract (if negative) to all of the elements of the Long array.

Comments :

See Also : cAddD, cAddI, cAddL, cAddS, Array routines

AddS

Purpose :

AddS adds a constant value to all of the elements of a Single array.

Declare Syntax :

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

Call Syntax :

status = cAddS(array(), value)

Where :

array()	is the Single array.
nValue	is the value to add (if positive) or to substract (if negative) to all of the elements of the Single array.

Comments :

See Also : cAddD, cAddI, cAddL, cAddS, Array routines

AddTime

Purpose :

AddTime retrieves 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)
        -> test = 1

test = cAddTime(2-4)
        -> test = 1438
```

See also : Date, Hour and Time routines

AllSubDirectories

Purpose :

AllSubDirectories retrieves 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 : cSubDirectory

ArabicToRoman

Purpose :

ArabicToRoman converts an integer or a long integer into Roman representation

Declare Syntax :

```
Declare Function cArabicToRoman Lib "time2win.dll" (Var As Variant) As String
```

Call Syntax :

```
test = cArabicToRoman(var)
```

Where :

var	is the integer or long integer value
test	returns the Roman representation of var

Comments :

The string returned by this function is always in lowercase

Examples :

```
test = cArabicToRoman(1994)
        test -> MCMXCIV

test = cArabicToRoman(1995)
        test -> MCMXCV

test = cArabicToRoman(1993)
        test -> MCMXCIII
```

ArrayPrm

Purpose :

ArrayPrm retrieves 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
status% = cArrayPrm(array(), arrayDef)
          array1.Bounds             is 1048577
          array1.LBound             is 1
          array1.UBound             is 16
          array1.ElemSize           is 2 (INTEGER)
          array1.IndexCount         is 1
          array1.TotalElem          is 16

Dim array(-7 To 25)         As Double
Dim arrayDef                as ArrayType
status% = cArrayPrm(array(), arrayDef)
          array1.Bounds             is 1703929
          array1.LBound             is -7
          array1.UBound             is 25
          array1.ElemSize           is 8 (DOUBLE)
          array1.IndexCount         is 1
          array1.TotalElem          is 33

Dim array(-10 To 10, 1 TO 7)          As Long
Dim arrayDef                as ArrayType
status% = cArrayPrm(array(), arrayDef)
          array1.Bounds             is 458753
          array1.LBound             is 1
          array1.UBound             is 7
          array1.ElemSize           is 4 (SINGLE)
          array1.IndexCount         is 2
          array1.TotalElem          is 7
```

See also : Constants and Types declaration

Between

Purpose :

Between checks to see if a value is between two other values.

Declare Syntax :

Declare Function cBetween 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 var1 and var2
	FALSE if var is not 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

var = 10
test = cBetween(var, var1, var2)
        -> test = TRUE
```

See Also : cTrueBetween

BlockCharFromLeft

Purpose :

BlockCharFromLeft reads n chars from the left of a string.

Declare Syntax :

Declare Function cBlockCharFromLeft Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As String

Call Syntax :

Test = cBlockCharFromLeft(Txt, Position)

Where :

Txt	the string to extract some left chars
Position	the number of chars to read
Test	the result

Comments :

This fonction is the same that Left$(Txt, Position) but doesn't generate an Error if a problem occurs.

Examples :

```
Txt = "ABCDEF"
Position = 3
Test = cBlockCharFromLeft(Txt, Position)
        Test = "ABC"
```

See also : cBlockCharFromLeft, cBlockCharFromRight, cOneCharFromLeft, cOneCharFromRight

BlockCharFromRight

Purpose :

BlockCharFromRight reads n chars from the right of a string.

Declare Syntax :

Declare Function cBlockCharFromRight Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As String

Call Syntax :

Test = cBlockCharFromRight(Txt, Position)

Where :

Txt	the string to extract some right chars
Position	the number of chars to read
Test	the result

Comments :

This fonction is the same that Right$(Txt, Position) but doesn't generate an Error if a problem occurs.

Examples :

```
Txt = "ABCDEF"
Position = 3
Test = cBlockCharFromRight(Txt, Position)
        Test = "DEF"
```

See also : cBlockCharFromLeft, cBlockCharFromRight, cOneCharFromLeft, cOneCharFromRight

ChDir

Purpose :

ChDir changes 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 : cChDrive

ChDrive

Purpose :

ChDir changes 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 : cChDir

CheckChars

Purpose :

CheckChars verifies 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
```

FilterX

Purpose :

FilterBlocks removes one or more sub-string separated by two delimitors in a gived string.
FilterChars removes some chars specifien in a gived string.
FilterFirstChars removes some chars beginning at first position of a gived string.
FilterNotChars removes 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"
charSet = "B/"
test = cFilterChars(Txt, charSet)
        test = "ACDEFGHIJ"

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

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

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

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

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

Txt = "A/BC/DEF/GHIJ"
charSet = "AF/"
test = cFilterNotChars(Txt, charSet)
        test = "A//F/"
```

SaveCtlLanguage, ReadCtlLanguage

Purpose :

SaveCtlLanguage creates or updates a file which contains the text for supporting a language.
ReadCtlLanguage reads a file which contains the text for supporting a language.

Declare Syntax :

Declare Function cSaveCtlLanguage Lib "time2win.dll" (Ctl As Control, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
Declare Function cReadCtlLanguage Lib "time2win.dll" (Ctl As Control, ByVal Property As Integer, ByVal FileLanguage As String) As Integer

Call Syntax :

test% = cSaveCtlLanguage(Ctl, Property, FileLanguage)
test% = cReadCtlLanguage(Ctl, Property, FileLanguage)

Where :

Ctl	is any control on the form to use the text language.
Property	is an association of constants (RS_CAPTION, RS_TEXT, RS_DATAFIELD, RS_DATASOURCE)
FileLangue	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 any 'OR' association of the four following constants (RS_CAPTION or RS_TEXT or RS_DATAFIELD or RS_DATASOURCE)

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 : Constants and Types declaration

CheckNumericity

See cIsDigit

FileCompressTab, FileExpandTab

Purpose :

FileCompressTab compress a number of spaces specified into a TAB char (horizontal tab).
FileExpandTab expands 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 :

CheckTime

Purpose :

CheckTime verifies 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)
        -> test = TRUE

Hr = 120 (02:00)
test = cCheckTime(Hr, Hr1, Hr2)
        -> test = FALSE
```

See also : cBetween, cTrueBetween, Date, Hour and Time routines

FileLastX

Purpose :

These routines read the date/time for a specified file.

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 :

The created, access, modified time/date are the same. The different routines are present for future version of Windows.

Compact

Purpose :

Compact compacts a string composed of numeric chars.

Declare Syntax :

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

Call Syntax :

test = cCompact(Txt)

Where :

Txt	is the string (only numeric chars) to compact
test	returns the string compacted

Comments :

If the size of the string is not a multiple of 2, the size used is the nearest below multiple of 2.

Examples :

```
Txt = "39383736353433323130"
test = cCompact(Txt)
        test = "9876543210"
```

See also : cUncompact

Compress

Purpose :

Compress removes 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 : cCompressTab, cExpandTab

CompressTab

Purpose :

CompressTab packs all n space chars into a tab char.

Declare Syntax :

Declare Function cCompressTab Lib "time2win.dll" (Txt As String, ByVal nTab As Integer) As String

Call Syntax :

test = cCompressTab(Txt, nTab)

Where :

Txt	the string to proceed
nTab	the number of space chars to replace by a tab char
test	the result

Comments :

Examples :

```
Txt = "A" + space$(2) + "B" + space$(3) + "C" + space$(4) + "D"
nTab = 2
test = cCompressTab(Txt, nTab)
        test = "A" + chr$(9) + "B" + chr$(9) + space$(1) + "C" + char$(9) + chr$(9) + "D"
```

See also : cCompress, cExpandTab

Count

Purpose :

Count counts 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
```

CountDirectories

Purpose :

CountDirectories counts 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 : cCountFiles

CountFiles

Purpose :

CountFiles counts 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 : cCountDirectories

CreateAndFill

Purpose :

CreateAndFill creates 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 : cFill

CreateBits

Purpose :

CreateBits creates 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 : Bit String Manipulation routines

CurrentTime

Purpose :

CurrentTime returns 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
```

MKx

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 : cCVB, cCVC, cCVD, cCVI, cCVL, cCVS, cBig.x.

DaysInMonth

Purpose :

DaysInMonth returns 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)
        test = 31

nYear = 1995
nMonth = 2
test = cDaysInMonth(nYear, nMonth)
        test = 28
```

Decrypt

Purpose :

Decrypt decodes a string encoded with Encrypt function.

Declare Syntax :

Declare Function cDecrypt Lib "time2win.dll" (Txt As String, password As String, ByVal level As Integer) As String

Call Syntax :

test = cDecrypt(Txt, password, level)

Where :

Txt	is the string to decrypt
password	is the key to use for decryption
level	level of the encryption
test	is the string decrypted

Comments :

The password/key is case sensitive.
The level is a number between 0 and 3 (Constants and Types declaration).
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_3
test = cEncrypt(Txt, password, level)
        txt = cDecrypt(test, password, level)
```

See also : cEncrypt

DeviationD

Purpose :

DeviationD will calculate the standard deviation from all elements in a Double array.

Declare Syntax :

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

Call Syntax :

deviation = cDeviationD(array())

Where :

array()	is the Double array.
deviation	is the standard deviation calculated. This value is always a Double value.

Comments :

See Also : cDeviationD, cDeviationI, cDeviationL, cDeviationS, Array routines

DeviationI

Purpose :

DeviationI will calculate the standard deviation from all elements in an Integer array.

Declare Syntax :

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

Call Syntax :

deviation = cDeviationI(array())

Where :

array()	is the Integer array.
deviation	is the standard deviation calculated. This value is always a Double value.

Comments :

See Also : cDeviationD, cDeviationI, cDeviationL, cDeviationS, Array routines

DeviationL

Purpose :

DeviationL will calculare the standard deviation from all elements in a Long array.

Declare Syntax :

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

Call Syntax :

deviation = cDeviationL(array())

Where :

array()	is the Long array.
deviation	is the standard deviation calculated. This value is always a Double value.

Comments :

See Also : cDeviationD, cDeviationI, cDeviationL, cDeviationS, Array routines

DeviationS

Purpose :

DeviationS will calculare the standard deviation from all elements in a Single array.

Declare Syntax :

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

Call Syntax :

deviation = cDeviationS(array())

Where :

array()	is the Single array.
deviation	is the standard deviation calculated. This value is always a Double value.

Comments :

See Also : cDeviationD, cDeviationI, cDeviationL, cDeviationS, Array routines

Encrypt

Purpose :

Encrypt encodes a string with a password/key.

Declare Syntax :

Declare Function cEncrypt Lib "time2win.dll" (Txt As String, password As String, ByVal level As Integer) As String

Call Syntax :

test = cEncrypt(Txt, password, level)

Where :

Txt	is the string to encrypt
password	is the key to use for encryption
level	level of the encryption
test	is the string decrypted

Comments :

The password/key is case sensitive.
The level is a number between 0 and 3 (Constants and Types declaration).
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_3
test = cEncrypt(Txt, password, level)
        txt = cDecrypt(test, password, level)
```

See also : cDecrypt

ExitWindowsAndExecute, RebootSystem, RestartWindows

Purpose :

ExitWindowsAndExecute terminates Windows, runs a specified MS-DOS application, and then restarts Windows.
RebootSystem reboots your system.
RestartWindows restarts 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()
```

ExpandTab

Purpose :

ExpandTab unpacks all tab chars into n space chars.

Declare Syntax :

Declare Function cExpandTab Lib "time2win.dll" (Txt As String, ByVal nTab As Integer) As String

Call Syntax :

test = cExpandTab(Txt, nTab)

Where :

Txt	the string to proceed
nTab	the number of space chars which replace a tab char
test	the result

Comments :

Examples :

Txt = test = "A" + chr$(9) + "B" + chr$(9) + space$(1) + "C" + char$(9) + chr$(9) + "D"
nTab = 2
test = cExpandTab(Txt, nTab)
        test =   "A" + space$(2) + "B" + space$(3) + "C" + space$(4) + "D"

See also : cCompress, cCompressTab

FileCRC32

Purpose :

FileCRC32 calculates 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)
	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 : cStringCRC32, Constants and Types declaration

FileDrive

Purpose :

FileDrive extracts the drive on which the file is present.

Declare Syntax :

Declare Function cFileDrive Lib "time2win.dll" (ByVal lpFilename As String) As String

Call Syntax :

test$ = cFileDrive(lpFilename)

Where :

lpFilename	the file to proceed
test$	EMPTY is the file not exist or an error occurs when accessing the file
	DRIVE LETTER for the file

Comments :

FileLineCount

Purpose :

FileLineCount counts 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& =

See also :

FilePathExists

Purpose :

FilePathExists verifies 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 :

CVx

Purpose :

CVB, CVC, CVD, CVI, CVL and CVS returns number in a certain precision given a string containing the IEEE representation of the number. Six separate functions are provided, with one each intended for BYTE, CURRENCY, DOUBLE, INTEGER, LONG and SINGLE.

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 : cMKB, cMKC, cMKD, cMKI, cMKL, cMKS

GetDiskFree, GetDiskSpace, GetDiskUsed, GetDiskClusterSize

Purpose :

GetDiskFree, GetDiskSpace, GetDiskUsed and GetDiskClusterSize retrieves respectively the free disk space, the size of the disk, the part of the disk used and the size of a cluster on a specified 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 : cFileSize, cFilesSize, cFilesSizeOnDisk, cFilesSlack

FilesInDirectory

Purpose :

FilesInDirectory retrieves 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 directoty 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 : CallSubDirectories, cSubDirectory

FileSize

Purpose :

FileSize returns 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 : cFilesSize, cFilesSizeOnDisk, cFilesSlack

FilesSize

Purpose :

FilesSize returns the logical size of all files specified by file mask.
FilesSizeOnDisk returns the physical size of all files specified by file mask.
FilesSlack returns 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 : cFileSize, cGetDiskClusterSize

IsFileX

Purpose :

The routines checks if a specified file has or not the specified attribute.
IsFilenameValid checks if the filename follows the DOS syntax for a file.
FileGetAttrib retrieves in a Call, all attributes of a gived file.

Declare Syntax :

```
Declare Function cIsFileArchive 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

Declare Function cFileGetAttrib Lib "time2win.dll" (ByVal nFilename As String, nFileAttribute As Any) As Integer
```

Call Syntax :

```
test% = cIsFileArchive(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)

test% = cFileGetAttrib(nFilename, nFileAttribute)
```

Where :

nFilename	the filename to check
nStatus	the status to check (only for cIsFileFlag) combine A_NORMAL, A_RDONLY, A_HIDDEN, A_SYSTEM, A_VOLID, A_SUBDIR, A_ARCH with logical OR.
nFileAttribute	the type variable 'FileAttributeType' (only for cFileGetAttrib)
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 : IsX Family Test routines, Constants and Types declaration

FillD

Purpose :

FillD fills, with an automatic incremented value, all of the elements of a Double array.

Declare Syntax :

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

Call Syntax :

status = cFillD(array(), nValue)

Where :

array()	is the Double array.
nValue	is the Double value automatiCally incremented by one.
status	is always TRUE.

Comments :

See Also : cFillD, cFillI, cFillL, cFillS, Array routines

FillI

Purpose :

FillI fills, with an automatic incremented value, all of the elements of an Integer array.

Declare Syntax :

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

Call Syntax :

status = cFillI(array(), nValue)

Where :

array()	is the Integer array.
nValue	is the Integer value automatiCally incremented by one.
status	is always TRUE.

Comments :

See Also : cFillD, cFillI, cFillL, cFillS, Array routines

FillL

Purpose :

FillL fills, with an automatic incremented value, all of the elements of a Long array.

Declare Syntax :

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

Call Syntax :

status = cFillL(array(), nValue)

Where :

array()	is the Long array.
nValue	is the Long value automatiCally incremented by one.
status	is always TRUE.

Comments :

See Also : cFillD, cFillI, cFillL, cFillS, Array routines

FillS

Purpose :

FillS fills, with an automatic incremented value, all of the elements of a Single array.

Declare Syntax :

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

Call Syntax :

status = cFillS(array(), nValue)

Where :

array()	is the Single array.
nValue	is the Single value automatiCally incremented by one.
status	is always TRUE.

Comments :

See Also : cFillD, cFillI, cFillL, cFillS, Array routines

Conversion table for Hundreds

The table below show the international table conversion between minutes and hundreds.
Don't forget that some hundreds are rounded.

Minutes	Hundreds	true value		Minutes	Hundreds	true value
0	**00**	0	|	30	**50**	50
1	**02**	1,666667	|	31	**52**	51,666667
2	**03**	3,333333	|	32	**53**	53,333333
3	**05**	5	|	33	**55**	55
4	**07**	6,666667	|	34	**57**	56,666667
5	**08**	8,333333	|	35	**58**	58,333333
6	**10**	10	|	36	**60**	60
7	**12**	11,66667	|	37	**62**	61,66667
8	**13**	13,33333	|	38	**63**	63,33333
9	**15**	15	|	39	**65**	65
10	**17**	16,66667	|	40	**67**	66,66667
11	**18**	18,33333	|	41	**68**	68,33333
12	**20**	20	|	42	**70**	70
13	**22**	21,66667	|	43	**72**	71,66667
14	**23**	23,33333	|	44	**73**	73,33333
15	**25**	25	|	45	**75**	75
16	**27**	26,66667	|	46	**77**	76,66667
17	**28**	28,33333	|	47	**78**	78,33333
18	**30**	30	|	48	**80**	80
19	**32**	31,66667	|	49	**82**	81,66667
20	**33**	33,33333	|	50	**83**	83,33333
21	**35**	35	|	51	**85**	85
22	**37**	36,66667	|	52	**87**	86,66667
23	**38**	38,33333	|	53	**88**	88,33333
24	**40**	40	|	54	**90**	90
25	**42**	41,66667	|	55	**92**	91,66667
26	**43**	43,33333	|	56	**93**	93,33333
27	**45**	45	|	57	**95**	95
28	**47**	46,66667	|	58	**97**	96,66667
29	**48**	48,33333	|	59	**98**	98,33333

Note : you can see if you've a good look in this table that some difference between two minutes are "better" than others if converted in hundreds. This is due to the rounding value.

if I works from 12 to 16 minutes (4 minutes), I've worked (27 - 20) = 7 hundreds
if I works from 16 to 20 minutes (4 minutes), I've worked (33 - 27) = 6 hundreds

In the two cases, I've worked 4 minutes but in the first case, I receive 7 hundreds and in the second case, I receive only 6 hundreds.

TypeX

Purpose :

TypesCompare compares two Types variable.
CompareTypeString compares a Type to a String.
CompareStringType compares a String to a Type.

TypeClear clears a Type variable.
TypeMid extracts information from a Type variable.

TypesCopy copies a Type variable into a variable.
TypeTransfert transfers a Type variable into a String.

StringToType copies a String to a Type variable.
TypeToString copies a Type 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 variable
Src, Dst,	the String variable
Offset	the offset in the Type variable
Length	the length in the Type 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 variable mixed with INTEGER, LONG, SINGLE, DOUBLE, CURRENCY and FIXED STRING can be used.

When   you compare 2 types variables or 1 type variable and 1 string, the size of each variable must be same.
When you copy 1 Type variable into a string or a string into Type variable, the size of each variable must be same.

Examples :

See also :

LngInpBox

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 : cLngBoxMsg, cLngMsgBox

FindBitReset

Purpose :

FindBitReset finds the first bit Reset 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

Call Syntax :

test = cFindBitReset(Txt, Position)

Where :

Txt	the string to proceed
Position	the starting position
test	TRUE if no bit founded
	<> TRUE if a bit founded

Comments :

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.

See also : Bit String Manipulation routines

FindBitSet

Purpose :

FindBitSet finds the first bit Set starting at the position gived for a a gived string.

Declare Syntax :

Declare Function cFindBitSet Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As Integer

Call Syntax :

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 :

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 : Bit String Manipulation routines

FindFileInEnv

Purpose :

FindFileInEnv searches if a specified file is present is the specified environment variable.

Declare Syntax :

Declare Function cFindFileInEnv Lib "time2win.dll" (ByVal lpFilename As String, ByVal lpEnv As String) As Integer

Call Syntax :

test% = cFindFileInEnv(lpFilename, lpEnv)

Where :

lpFilename	name of file to search for
lpEnv	environment to search
test%	TRUE if founded
	FALSE if not founded

Comments :

This function searches for the target file in the specified domain. The lpEnv variable can be any environment variable that specifies a list of directory paths, such as PATH, LIB, INCLUDE, or other user-defined variables. This function function is case-sensitive, so the lpEnv variable should match the case of the environment variable.
The routine first searches for the file in the current working directory. If it doesn't find the file, it next looks through the directories specified by the environment variable.

Examples :

```
test% = cFileFileInEnv("win.com", "windir")                    -> TRUE
```

See also : cFindFileInPath

FindFileInPath

Purpose :

FindFileInPath searches if a specified file is present is the path.

Declare Syntax :

Declare Function cFindFileInPath Lib "time2win.dll" (ByVal lpFilename As String) As Integer

Call Syntax :

test% = cFindFileInPath(lpFilename)

Where :

lpFilename	name of file to search for
test%	TRUE if founded
	FALSE if not founded

Comments :

This function searches for the target file in the PATH environment variable that specifies a list of directory paths.
The routine first searches for the file in the current working directory. If it doesn't find the file, it next looks through the all directories specified in the PATH environment variable.
This function is a subset of cFindFileInEnv : cFileFileInEnv(lpFilename, "PATH")

Examples :

test% = cFileFileInPath("xcopy.exe"")   -> TRUE

See also : cFindFileInEnv

FromBinary, FromBinary2, ToBinary, ToBinary2

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 : cFromHexa, cToHexa

FromHexa, ToHexa

Purpose :

ToHexa converts a ascii string to hexa string.
FromHexa converts a hexa string to an ascii 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 : cFromBinary, cToBinary

Get, GetBlock, GetIn

Purpose :

Get reads a sub-string delimited by **'|'** in a gived string.
GetBlock reads a block of n chars starting at a gived block in a gived string.
GetIn reads a sub-string delimited by a separator 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

Call Syntax :

test$ = cGet(Txt, Position)
test$ = cGetBlock(Txt, Position, Length)
test$ = cGetIn(Txt, Separator, 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
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)

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$ = cGetBlock("A/BC/DEF/G",1,2)      -> "A/"
test$ = cGetBlock("A/BC/DEF/G",4,2)      -> "EF"
```

See also : cSetDefaultSeparator, cInsertBlocks, cInsertBlockBy, cInsertByMask, cInsertChars

GetBit

Purpose :

GetBit returns 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 : Bit String Manipulation routines

IsFormEnabled

Purpose :

IsFormEnabled checks if the specified form is enabled or not.

Declare Syntax :

Declare Function cIsFormEnabled Lib "time2win.dll" (ByVal hWnd As Integer) As Integer

Call Syntax :

test% = cIsFormEnabled(hWnd)

Where :

hWnd	is the .hWnd of the specified form.
test%	TRUE if the form is enabled.
	FALSE is the form is disabled.

Comments :

If you disable a form with the cDisableForm or cDisableFI and if you display a MODAL form, you must take care that Windows reenables the disabled form.

Examples :

test% = cIsFormEnabled(Me.hWnd)

See also : cDisableForm, cEnableForm, cDisableFI, cEnableFI

GetChangeTaskName

Purpose :

GetChangeTaskName gets and changes 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 Integer, 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 : cChangeTaskName, cGetTaskName

FullPath

Purpose :

FullPath converts 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 : cSplitPath, cMakePath

LngBoxMsg, LngMsgBox

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 : cLngInpBox

SetCtlX

Purpose :

The functions below applies to a custom control.

SetCtlCaption sets the .Caption property of the control.
SetCtlDataField sets the .DataField property of the control.
SetCtlFocus gives the Focus to a control.
SetCtlPropString sets the specified property (founded with cGetCtlPropString function) of the control.
SetCtlTag sets the .Tag property of the control.
SetCtlText sets the .Text property of the control.

Declare Syntax :

```
Declare Sub cSetCtlCaption Lib "time2win.dll" (Ctl As Control, ByVal Text As String)
Declare Sub cSetCtlDataField Lib "time2win.dll" (Ctl As Control, ByVal Text As String)
Declare Sub cSetCtlFocus Lib "time2win.dll" (Ctl As Control)
Declare Sub cSetCtlPropString Lib "time2win.dll" (Ctl As Control, ByVal PropIndex As Integer, ByVal Text As String)
Declare Sub cSetCtlTag Lib "time2win.dll" (Ctl As Control, ByVal Text As String)
Declare Sub cSetCtlText Lib "time2win.dll" (Ctl As Control, ByVal Text As String)
```

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 : cSetX, cGetX, cGetCtlX

Morse

Purpose :

Morse converts 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")	is '--- ... ---'
test$ = cMorse("TIME TO WIN")	is '. -- .. - ~. ... ~-.. -- .- '

See also :

GetCurrentDrive

Purpose :

GetCurrentDrive returns 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 : cGetDefaultCurrentDir

GetAscTime

Purpose :

GetAscTime retrieves 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 language number defined in Constants and Types declaration. 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 : Get.x.Day, Get.x.Month

GetDefaultCurrentDir

Purpose :

GetDefaultCurrentDir retrieves 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 : cGetDriveCurrentDir, cGetCurrentDrive

GetDefaultPrinter

Purpose :

GetDefaultPrinter returns 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 : cGetPrinterPorts

GetDevices

Purpose :

GetDevices returns 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 : cGetDefaultPrinter

GetDriveCurrentDir

Purpose :

GetDriveCurrentDir retrieves 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 : cGetDefaultCurrentDir, cGetCurrentDrive

GetDriveType

Purpose :

GetDriveType determines 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 :

The returned value can be :

DRIVE_UNKNOW (drive type can't be founded, drive not present or unknow)
DRIVE_REMOVABLE (disk can be removed from the drive)
DRIVE_FIXED (disk cannot be removed from the drive)
DRIVE_REMOTE (drive is a remote, or network, drive)
DRIVE_CDROM (drive is a cd-rom)

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 : Constants and Types declaration

GetFileVersion

Purpose :

GetFileVersion returns 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 : cGetFileVersionInfo, Constants and Types declaration

GetFileVersionInfo

Purpose :

GetFileVersionInfo returns 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 : cGetFileVersion, Constants and Types declaration

GetFullNameInEnv

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

GetFullNameInPath

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

SetX

Purpose :

The functions below applies to the .hWnd of a custom control.

SetCaption sets the .Caption property of the control.
SetDataField sets the .DataField property of the control.
SetFocus gives the Focus to a control.
SetTag sets the .Tag property of the control.
SetText sets the .Text property of the control.

Declare Syntax :

```
Declare Sub cSetCaption Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetDataField Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetFocus Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Sub cSetTag Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetText Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
```

Call Syntax :

The purpose and the declare syntax are very explicite.

Where :

hWnd	the hWnd of the custom control.

Comments :

•The advantage to use these routines is that these routines doesn't generates an error if the property not exists.
•If the custom control doesn't have a .hWnd (Label control b.e.), you must use the cSetCtlX function.

Examples :

See also : cSetCtlX, cGetX, cGetCtlX

GetIni

Purpose :

see Comments

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 : cPutIni

GetNetConnection

Purpose :

The GetNetConnection function returns 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 :

FileReset

Purpose :

FileResetAllAttrib, FileResetArchive, FileResetHidden, FileResetReadOnly, FileResetSystem, FileResetFlag resets respectively all attributes, archive attribute, hidden attribute, read-only attribute, system attribute, specified attribute for the gived file.

Declare Syntax :

```
Declare Function cFileResetAllAttrib Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetArchive Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetHidden Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetReadOnly Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetSystem Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetFlag Lib "time2win.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer
```

Call Syntax :

```
status = cFileResetAllAttrib(nFilename)
status = cFileResetArchive(nFilename)
status = cFileResetHidden(nFilename)
status = cFileResetReadOnly(nFilename)
status = cFileResetSystem(nFilename)
status = cFileResetFlag(nFilename, nStatus)
```

Where :

nFilename	is the filename to change the attributes
nStatus	is a combination of A_NORMAL, A_RDONLY, A_HIDDEN, A_SYSTEM, A_ARCH
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 : FileSet

GetPid

Purpose :

cGetPid returns the process ID, an integer that uniquely identifies the Calling process.

Declare Syntax :

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

Call Syntax :

test% = cGetPid()

Where :

test%	the return process ID

Comments :

In the MS-DOS environment, the process ID is usually considered to be the address of the program segment prefix, or PSP. However, in environments with multiple MS-DOS sessions, such as Windows, this value is often not unique. Therefore, the value returned by cGetPid in the MS-DOS libraries is a value based on a combination of the program segment prefix and the system time at the moment when cGetPid is Called for the first time.

GetPrinterPorts

Purpose :

GetPrinterPorts returns 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 : cGetDefaultPrinter

GetSectionItems

Purpose :

GetSectionItems retrieves 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

GetSystemDirectory

Purpose :

GetSystemDirectory retrieves 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:\WINDOWS\SYSTEM"
```

See also : cGetWindowsDirectory

GetTaskName

Purpose :

GetTaskName reads 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 Integer) 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
            TaskName is "Microsoft Visual Basic"
```

See also : cChangeTaskName, cGetChangeTaskName

SetCapture, ResetCapture

Purpose :

SetCapture and ResetCapture captures or liberates the mouse and keyboard inputs to a hWnd of a control. Only this control can receive the inputs.

Declare Syntax :

```
Declare Sub cSetCapture Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Sub cResetCapture Lib "time2win.dll" ()
```

Call Syntax :

```
Call cSetCapture(hWnd)
Call cResetCapture
```

Where :

hWnd	the hWnd of a control

Comments :

Use this with caution.
If your program crashes, the inputs are limited to the window specified by the control.
Only a control at a gived time can be use these functions.

GetWindowsDirectory

Purpose :

GetWindowsDirectory retrieves 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:\WINDOWS"
```

See also : cGetSystemDirectory

Distribution Note

When you create and distribute applications that use 'TIME TO WIN', you should install the file TIME2WIN.DLL in the customer's Microsoft Windows \SYSTEM 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 '**TIME2WIN.LIC**' *file with any application that you distribute.*

GetWinSection

Purpose :

GetWinSection retrieves all items founden in a section of the Win.INI.

Declare Syntax :

```
Declare Function cGetWinSection Lib "time2win.dll" (ByVal Section As String) As String
```

Call Syntax :

```
test$ = cGetWinSection(Section)
```

Where :

Section	is the section to proceed
test$	is the contents of the specified section

Comments :

Each item in the section is separated by a chr$(13).

Examples :

```
Dim n              As Integer

Debug.Print cGetWinSection("desktop")
```

On my system :

```
Pattern=(None)
GridGranularity=0
IconSpacing=77
TileWallPaper=1
IconTitleFaceName=MS Sans Serif
IconTitleSize=-11
IconTitleStyle=0
IconVerticalSpacing=72
wallpaper=(None)
```

See also : cGetSectionItems

GiveBitPalindrome

Purpose :

GiveBitPalindrome returns 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 : Bit String Manipulation routines

HourTo

Purpose :

HourTo converts 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, Hour and Time routines

MixChars

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")
test2$ = cMixChars("Nothing can beat the fox")
```

On my system :

```
        test1$ = "ON EI WMTIT"
        test2$ = "Nt honn ia ttechx baefog"
```

See also :

IntoBalance, IntoBalanceFill

Purpose :

IntoBalance converts a VARIANT value (INTEGER or LONG) in a time string.
IntoBalance converts 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, Hour and Time routines

IntoDate, IntoDateFill, IntoDateNull

Purpose :

IntoDate converts a date value into a date string specified the short date format order in the Control Panel.
IntoDateFill converts 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 converts 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, Hour and Time routines

IntoFixHour, IntoHour, IntoVarHour

Purpose :

IntoFixHour is super-set for converting a VARIANT (INTEGER or LONG) into a fixed time string.
IntoHour concerts a VARIANT (INTEGER or LONG) into a hour string.
IntoVarHour concerts 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, Hour and Time routines, Conversion table for Hundreds

LngSysMenu

Purpose :

LngSysMenu changes 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 Integer)

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 : cSysMenuChange

IsBitPalindrome

Purpose :

IsBitPalindrome checks 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 : Bit String Manipulation routines

FileToLower, FileToUpper

Purpose :

FileToLower converts a file to a file with lower case.
FileToLower converts 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 :

IsX

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 : IsX Family Test routines

FileMerge

Purpose :

FileMerge merges 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 : cFileCopy

BigAdd, BigDiv, BigMul, BigSub, BigFmt

Purpose :

BigAdd, BigDiv, BigMul, BigSub performs Addition, Substraction, Multiplication, Division of big double value.
BigFmt displays a big double value into a string to display or print it.

Declare Syntax :

```
Declare Function cBigAdd Lib "time2win.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigDiv Lib "time2win.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigMul Lib "time2win.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigSub Lib "time2win.dll" (Num1 As String, Num2 As String) As String

Declare Function cBigFmt Lib "time2win.dll" (Num As String, ByVal Fmt As Integer) As String
```

Call Syntax :

```
test$ = cBigAdd(num1$, num2$)
test$ = cBigDiv(num1$, num2$)
test$ = cBigMul(num1$, num2$)
test$ = cBigSub(num1$, num2$)

test$ = cBigFmt(num$, fmt%)
```

Where :

num1$	is the first big double value (string representation) (left operand).
num2$	is the second big double value (string representation) (right operand).
num$	is a big double value to format it (string representation).
fmt%	is the significant number of formatting.
test$	is the returned value.

Comments :

A big double value (string representation) is always a string with 10 chars.
The cBigFmt can process from 1 TO 19 significant numbers (not included the exponent). If the significant number is below or equal to 0 then 19 is used.

Examples :

```
    Dim m1          As Double
    Dim m2          As Double

    m1 = 123456789012345#
    m2 = 987654321098765#

For the double test          : m1 + m2
                               m1 / m2
                               m1 * m2
                               m1 - m2

For the big double test      : cBigAdd(cMKN(str$(m1)),cMKN(str$(m2)))
                               cBigDiv(cMKN(str$(m1)),cMKN(str$(m2)))
                               cBigMul(cMKN(str$(m1)),cMKN(str$(m2)))
                               cBigSub(cMKN(str$(m1)),cMKN(str$(m2)))

Double            : Add '123456789012345' and '987654321098765' is '1,11111111011111E+15'
Big Double        : Add '123456789012345' and '987654321098765' is '1111111110111110'

Double            : Sub '123456789012345' and '987654321098765' is '-864197532086420'
```

```
Big Double          : Sub '123456789012345' and '987654321098765' is '-864197532086420'

Double              : Mul '123456789012345' and '987654321098765' is '1,21932631137021E+29'
Big Double          : Mul '123456789012345' and '987654321098765' is '1.219326311370210714e+029'

Double              : Div '123456789012345' and '987654321098765' is ',124999998860937'
Big Double          : Div '123456789012345' and '987654321098765' is '0.1249999988609368673'
```

See also : cMKN

Big Numbers

cBigAdd
cBigDiv
cBigMul
cBigSub

cMKN

cBigNum

GetClassName

Purpose :

GetClassName retrieves the full class name of a control.

Declare Syntax :

Declare Function cGetClassName Lib "time2win.dll" (ByVal hWnd As Integer) 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 : cGetClass, cGetCtlClass

BigNum

Purpose :

BigNum make some operations on two big numbers. BigNum can handle big numbers (without decimal part) greater than the limit of a long integer.

Declare Syntax :

Declare Function cBigNum Lib "time2win.dll" (ByVal n1 As String, ByVal op As Integer, ByVal n2 As String) As String

Call Syntax :

test$ = cBigNum(n1$, op%, n2$)

Where :

n1$	is the first big number (left operand).
op%	is the operation to perform. (see Constants and Types declaration)
n2$	is the second big number (right operand).

Comments :

A big number is a string which have a representation of a number but on a string form. The big number can't have decimal part.
A big number can have a sign : '+' or '' for positive value, '-' for negative value. The sign must be the first char.
A big number can't have any other chars that the following chars : "+-0123456789", others chars are filtered and dus not processed.
The leading's 0 are automatically removed for the calculation.

Examples :

```
Dim X           As String
Dim Y           As String
Dim Z           As String

X = "12345678901234567890l"
Y = "98765432109876543210O"

Z = cBigNum(X, BIG_ADD, Y)

        '(X) + (Y)'     is '1111111110111111111001'
        '(X) + (-Y)'    is '-864197532086419753199'
        '(-X) + (Y)'    is '864197532086419753199'
        '(-X) + (-Y)'   is '-1111111110111111111001'

Z = cBigNum(X, BIG_SUB, Y)

        '(X) - (Y)'     is '-864197532086419753199'
        '(X) - (-Y)'    is '1111111110111111111001'
        '(-X) - (Y)'    is '-1111111110111111111001'
        '(-X) - (-Y)'   is '864197532086419753199'

Z = cBigNum(X, BIG_MUL, Y)

        '(X) * (Y)'     is '121932631137021795224734034432225118122100'
        '(X) * (-Y)'    is '-121932631137021795224734034432225118122100'
        '(-X) * (Y)'    is '-121932631137021795224734034432225118122100'
        '(-X) * (-Y)'   is '121932631137021795224734034432225118122100'
```

See also : cBig.x.

Returned Errors

-32720
: The number of chars in a block for writing differs from the number of chars for reading.

-32730
: An error has occured when reading the file (bad CRC, bad cluster, ...).

-32740
: An error has occured when writing a file (bad CRC, bad cluster, not a valid drive, not enough space on drive).

-32759 to -32750
: An error has occured when opening a file.

-32767 to -32761
: An error has occured when allocating memory buffer

KillDir

Purpose :

KillDir deletes the specified empty directory.
KillDirs deletes 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 cKillDir :

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 cKillDirs :

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 : cKillFile, cKillFiles, cKillDirFilesAll

KillFile, KillFileAll

Purpose :

KillFile deletes the specified filename.
KillFileAll deletes 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 : cKillFiles, cKillFilesAll, cKillDir, cKillDirs, cKillDirFilesAll

KillFilesAll

Purpose :

KillFiles deletes all files specified by a file mask.
KillFilesAll deletes 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 : cKillFile, cKillFileAll, cKillDir, cKillDirs

Lrc

Purpose :

Lrc calculates 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 : cStringCRC32, cFileCRC32

MakeDir

Purpose :

MakeDir creates the specified directory.

Declare Syntax :

Declare Function cMakeDir Lib "time2win.dll" (ByVal lpFilename As String) As Integer

Call Syntax :

test% = cMakeDir(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.

Examples :

test% = cMakeDir("C:\")	-> 13 (<> TRUE => an error has occured)
test% = cMakeDir("C:\~~TEST~~")	-> TRUE (no error, the directory has been created)

See also : cChDir, cKillDir

Max

Purpose :

Max returns the highest 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

Call Syntax :

test = cMax(Var1, Var2)

Where :

Var1	the first value
Var2	the second value
test	the highest value of the two

Comments :

Examples :

```
test = cMax(1234, 4321)          -> 4321
```

See also : cMin

MaxD

Purpose :

MaxD will return the largest value in a Double array.

Declare Syntax :

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

Call Syntax :

largest = cMaxD(array())

Where :

array()	is the Double array.
largest	is the largest value from all of the elements of the Double array.

Comments :

See Also : cMaxI, cMaxL, cMaxS, Array routines

MaxI

Purpose :

MaxI will return the largest value in an Integer array.

Declare Syntax :

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

Call Syntax :

largest = cMaxI(array())

Where :

array()	is the Integer array.
largest	is the largest value from all of the elements of the Integer array.

Comments :

See Also : cMaxD, cMaxL, cMaxS, Array routines

MaxL

Purpose :

MaxL will return the largest value in a Long array.

Declare Syntax :

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

Call Syntax :

largest = cMaxL(array())

Where :

array()	is the Long array.
largest	is the largest value from all of the elements of the Long array.

Comments :

See Also : cMaxD, cMaxI, cMaxS, Array routines

MaxS

Purpose :

MaxS will return the largest value in a Single array.

Declare Syntax :

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

Call Syntax :

largest = cMaxS(array())

Where :

array()	is the Single array.
largest	is the largest value from all of the elements of the Single array.

Comments :

See Also : cMaxD, cMaxI, cMaxL, Array routines

MeanD

Purpose :

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

Declare Syntax :

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

Call Syntax :

mean = cMeanD(array())

Where :

array()	is the Double array.
mean	is the mean calculated. This value is always a Double value.

Comments :

See Also : cMeanD, cMeanI, cMeanL, cMeanS, Array routines

MeanI

Purpose :

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

Declare Syntax :

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

Call Syntax :

mean = cMeanI(array())

Where :

array()	is the Integer array.
mean	is the mean calculated. This value is always a Double value.

Comments :

See Also : cMeanD, cMeanI, cMeanL, cMeanS, Array routines

MeanL

Purpose :

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

Declare Syntax :

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

Call Syntax :

mean = cMeanL(array())

Where :

array()	is the Long array.
mean	is the mean calculated. This value is always a Double value.

Comments :

See Also : cMeanD, cMeanI, cMeanL, cMeanS, Array routines

MeanS

Purpose :

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

Declare Syntax :

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

Call Syntax :

mean = cMeanS(array())

Where :

array()	is the Single array.
mean	is the mean calculated. This value is always a Double value.

Comments :

See Also : cMeanD, cMeanI, cMeanL, cMeanS, Array routines

Min

Purpose :

Max returns the smallest value of the two VARIANT value (INTEGER or LONG)

Declare Syntax :

Declare Function cMin Lib "time2win.dll" (Var1 As Variant, Var2 As Variant) As Variant

Call Syntax :

test = cMin(Var1, Var2)

Where :

Var1	the first value
Var2	the second value
test	the smallest value of the two

Comments :

Examples :

```
test = cMin(1234, 4321)          -> 1234
```

See also : cMax

MinD

Purpose :

MinD will return the smallest value in a Double array.

Declare Syntax :

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

Call Syntax :

smallest = cMinD(array())

Where :

array() is the Double array.
smallest is the smallest value from all of the elements of the Double array.

Comments :

See Also : cMinI, cMinL, cMinS, Array routines

MinI

Purpose :

MinI will return the smallest value in an Integer array.

Declare Syntax :

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

Call Syntax :

smallest = cMinI(array())

Where :

array() is the Integer array.
smallest is the smallest value from all of the elements of the Integer array.

Comments :

See Also : cMinD, cMinL, cMinS, Array routines

MinL

Purpose :

MinL will return the smallest value in a Long array.

Declare Syntax :

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

Call Syntax :

smallest = cMinL(array())

Where :

array()	is the Long array.
smallest	is the smallest value from all of the elements of the Long array.

Comments :

See Also : cMinD, cMinI, cMinS, Array routines

MinS

Purpose :

MinS will return the smallest value in a Single array.

Declare Syntax :

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

Call Syntax :

smallest = cMinS(array())

Where :

array() is the Single array.
smallest is the smallest value from all of the elements of the Single array.

Comments :

See Also : cMinD, cMinI, cMinL, Array routines

ModuleFind

Purpose :

ModuleFind retrieves 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 typed variable which receives the parameters (tagMODULEENTRY)
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:\WINDOWS\SYSTEM\KRNL386.EXE
MODULEENTRY.wNext = 279
```

See also : cModules, cTaskFind, cTasks, Constants and Types declaration

Modules

Purpose :

Modules retrieves 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 typed variable which receives the parameters (tagMODULEENTRY)
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:\WINDOWS\SYSTEM\KRNL386.EXE	279
276	SYSTEM	279	32	K:\WINDOWS\SYSTEM\SYSTEM.DRV	343
276	KEYBOARD	343	31	K:\WINDOWS\SYSTEM\KEYBOARD.DRV	367
276	MOUSE	367	31	K:\WINDOWS\SYSTEM\MOUSE.DRV RV	463
276	DISPLAY	463	32	K:\WINDOWS\SYSTEM\SVGA256.DRV	487
276	SOUND	487	31	K:\WINDOWS\SYSTEM\MMSOUND.DRV	583
276	COMM	583	31	K:\WINDOWS\SYSTEM\COMM.DRV RV	1271

See also : cModuleFind, cTaskFind, cTasks, Constants and Types declaration

NextHwnd

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

OneCharFromLeft

Purpose :

OneCharFromLeft reads 1 char at a position starting from the left of a string.

Declare Syntax :

Declare Function cOneCharFromLeft Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As String

Call Syntax :

test = cOneCharFromLeft(txt, position)

Where :

Txt	the string to extract one char
Position	the position of the char
Test	the result

Comments :

This function is the same that MID$(Txt, Position, 1)

Examples :

```
Txt = "ABCDEF"
Position = 3
Test = cOneCharFromLeft(Txt, Position)
        Test = "C"
```

See also : cBlockCharFromLeft, cBlockCharFromRight, cOneCharFromLeft, cOneCharFromRight

OneCharFromRight

Purpose :

OneCharFromRight reads 1 char at a position starting from the right of a string.

Declare Syntax :

Declare Function cOneCharFromRight Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As String

Call Syntax :

Test = cOneCharFromRight(Txt, Position)

Where :

Txt	the string to extract one char
Position	the position of the char
Test	the result

Comments :

This function is the same that MID$(Txt, Len(Txt) - Position + 1, 1)

Examples :

```
Txt = "ABCDEF"
Position = 3
Test = cOneCharFromRight(Txt, Position)
        Test = "D"
```

See also : cBlockCharFromLeft, cBlockCharFromRight, cOneCharFromLeft, cOneCharFromRight

PatternMatch

Purpose :

PatternMatch searches 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 : cPatternExtMatch

RebootSystem

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

RemoveBlockChar

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

RemoveOneChar

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

RenameFile

Purpose :

RenameFile renames 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.

ResizeString

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

ResizeStringAndFill

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

RestartWindows

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

Reverse

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

ReverseSortD

Purpose :

ReverseSortD will sort, in descending order, all elements in a Double array.

Declare Syntax :

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

Call Syntax :

status = cReverseSortD(array())

Where :

array()	is the Double array.
status	is always TRUE.

Comments :

See Also : cReverseSortD, cReverseSortI, cReverseSortL, cReverseSortS, cReverseSortStr, Array routines

ReverseSortI

Purpose :

ReverseSortD will sort, in descending order, all elements in an Integer array.

Declare Syntax :

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

Call Syntax :

status = cReverseSortI(array())

Where :

array()	is the Integer array.
status	is always TRUE.

Comments :

See Also : cReverseSortD, cReverseSortI, cReverseSortL, cReverseSortS, cReverseSortStr, Array routines

ReverseSortL

Purpose :

ReverseSortL will sort in descending order all elements in a Long array.

Declare Syntax :

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

Call Syntax :

status = cReverseSortL(array())

Where :

array()	is the Long array.
status	is always TRUE.

Comments :

See Also : cReverseSortD, cReverseSortI, cReverseSortL, cReverseSortS, cReverseSortStr, Array routines

ReverseSortS

Purpose :

ReverseSortS will sort in descending order all elements in a Single array.

Declare Syntax :

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

Call Syntax :

status = cReverseSortS(array())

Where :

array()	is the Single array.
status	is always TRUE.

Comments :

See Also : cReverseSortD, cReverseSortI, cReverseSortL, cReverseSortS, cReverseSortStr, Array routines

ReverseSortStr

Purpose :

ReverseSortD will sort, in descending order, a string divided in basis elements of a fixed length.

Declare Syntax :

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

Call Syntax :

status = cReverseSortStr(txt, nItem, ItemLength)

Where :

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 : cReverseSortD, cReverseSortI, cReverseSortL, cReverseSortS, cReverseSortStr, Array routines

RomanToArabic

Purpose :

RomanToArabic converts a Roman string into an integer or a long integer.

Declare Syntax :

Declare Function cRomanToArabic Lib "time2win.dll" (Txt As String) As Variant

Call Syntax :

test = cRomanToArabic(txt)

Where :

txt	is a Roman string.
test	returns the Arabic representation of txt.

Comments :

The value returned by this function is an integer or a long integer.

Examples :

```
test = cArabicToRoman(1994)
        test -> MCMXCIV

test = cArabicToRoman(1995)
        test -> MCMXCV

test = cArabicToRoman(1993)
        test -> MCMXCIII
```

See Also : cArabicToRoman

SetD

Purpose :

SetD fills, with the same value, all of the elements of a Double array.

Declare Syntax :

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

Call Syntax :

status = cSetD(array(), nValue)

Where :

array()	is the Double array.
nValue	is the Double value to initialize the array.
status	is always TRUE.

Comments :

See Also : cSetD, cSetI, cSetL, cSetS, Array routines

SetHandleCount

Purpose :

SetHandleCount specifies the number of file handles the application requires.

Declare Syntax :

Declare Function cSetHandleCount Lib "time2win.dll" (ByVal nHandle As Integer) As Integer

Call Syntax :

test% = cSetHandleCount(nHandle)

Where :

nHandle	to number of handles that you want.
test%	> 0 if all is OK
	= 0 if a problem has occured.

Comments :

The return value is the number of file handles available to the application, if the function is successful. This number may be less than the number of handles specified.

By default, the maximum number of file handles available to a task is 20.

If the specified number of handle is below or equal to 0, or greater than 255, the returned value is 0

Examples :

```
test% = cSetHandleCount(0)          -> 0
test% = cSetHandleCount(70)         -> 70
```

SetI

Purpose :

SetI fills, with the same value, all of the elements of an Integer array.

Declare Syntax :

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

Call Syntax :

status = cSetI(array(), nValue)

Where :

array()	is the Integer array.
nValue	is the Integer value to initialize the array.
status	is always TRUE.

Comments :

See Also : cSetD, cSetI, cSetL, cSetS, Array routines

SetL

Purpose :

SetL fills, with the same value, all of the elements of a Long array.

Declare Syntax :

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

Call Syntax :

status = cSetL(array(), nValue)

Where :

array()	is the Long array.
nValue	is the Long value to initialize the array.
status	is always TRUE.

Comments :

See Also : cSetD, cSetI, cSetL, cSetS, Array routines

SetS

Purpose :

SetS fills, with the same value, all of the elements of a Single array.

Declare Syntax :

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

Call Syntax :

status = cSetS(array(), nValue)

Where :

array()	is the Single array.
nValue	is the Single value to initialize the array.
status	is always TRUE.

Comments :

See Also : cSetD, cSetI, cSetL, cSetS, Array routines

Sleep

Purpose :

Sleep suspends 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 :

```
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

   Dim status     As Integer

   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"

SortD

Purpose :

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

Declare Syntax :

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

Call Syntax :

status = cSortD(array())

Where :

array()	is the Double array.
status	is always TRUE.

Comments :

See Also : cSortD, cSortI, cSortL, cSortS, cSortStr, Array routines

SortI

Purpose :

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

Declare Syntax :

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

Call Syntax :

status = cSortI(array())

Where :

array()	is the Integer array.
status	is always TRUE.

Comments :

See Also : cSortD, cSortI, cSortL, cSortS, cSortStr, Array routines

SortL

Purpose :

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

Declare Syntax :

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

Call Syntax :

status = cSortL(array())

Where :

array()	is the Long array.
status	is always TRUE.

Comments :

See Also : cSortD, cSortI, cSortL, cSortS, cSortStr, Array routines

SortS

Purpose :

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

Declare Syntax :

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

Call Syntax :

status = cSortS(array())

Where :

array()	is the Single array.
status	is always TRUE.

Comments :

See Also : cSortD, cSortI, cSortL, cSortS, cSortStr, Array routines

SortStr

Purpose :

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

Declare Syntax :

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

Call Syntax :

status = cSortStr(txt, nItem, ItemLength)

Where :

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 : cSortD, cSortI, cSortL, cSortS, cSortStr, Array routines

StringCRC32

Purpose :

StringCRC32 calculates 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 : cFileCRC32, Constants and Types declaration

SubDirectory

Purpose :

SubDirectory retrieves 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 : CallSubDirectories, cFilesInDirectory

SumD

Purpose :

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

Declare Syntax :

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

Call Syntax :

sum = cSumD(array())

Where :

array()	is the Double array.
sum	is the sum calculated. This value is always a Double value.

Comments :

See Also : cSumD, cSumI, cSumL, cSumS, Array routines

SumI

Purpose :

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

Declare Syntax :

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

Call Syntax :

sum = cSumI(array())

Where :

array()	is the Integer array.
sum	is the sum calculated. This value is always a Double value.

Comments :

See Also : cSumD, cSumI, cSumL, cSumS, Array routines

SumL

Purpose :

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

Declare Syntax :

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

Call Syntax :

sum = cSumL(array())

Where :

array()	is the Long array.
sum	is the sum calculated. This value is always a Double value.

Comments :

See Also : cSumD, cSumI, cSumL, cSumS, Array routines

SumS

Purpose :

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

Declare Syntax :

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

Call Syntax :

sum = cSumS(array())

Where :

array()	is the Single array.
sum	is the sum calculated. This value is always a Double value.

Comments :

See Also : cSumD, cSumI, cSumL, cSumS, Array routines

TaskFind

Purpose :

TaskFind retrieves 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:\WINDOWS\SYSTEM\KRNL386.EXE
MODULEENTRY.wNext = 279
```

See also : cModules, cModuleFind, cTasks, Constants and Types declaration

Tasks

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 : cModules, cModuleFind, cTaskFind, Constants and Types declaration

TimeBetween

Purpose :

TimeBetween calculates 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, Hour and Time routines

InsertBlocks, InsertBlocksBy, InsertByMask, InsertChars

Purpose :

InsertBlocks inserts different block of char in a gived string separated by '~'.
InsertBlocks inserts different block of char in a gived string separated by a gived separator.
InsertByMask replaces 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 : cGet, cGetIn, cGetBlock

AddDigit, CplDigit, NumDigit, CplAlpha

Purpose :

AddDigit sums all numerics chars in a gived string.
CplDigit returns the complementary string from a gived string composed with numerics chars.
NumDigit sums and sums all numerics chars in a gived string to have a maximum value of 9.
CplDigit returns 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("1234567890987654321712345678909876543217")  -> 194
test% = cNumDigit("1234567890987654321712345678909876543217")-> 5

test$ = cCplDigit("1234567890987654321712345678909876543217")    ->
"8765432109012345678287654321090123456782"

test% = cAddDigit("8765432109012345678287654321090123456782")  -> 166
test% = cNumDigit("8765432109012345678287654321090123456782")-> 4

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

GetCtlX

Purpose :

The functions below applies to a custom control.

GetCtlCaption returns the .Caption property.
GetCtlClass returns the class name defined in the properties windows in the design-mode of VB.
GetCtlContainer returns the name of the container did contains the control. The container can be the form or an another control.
GetCtlDataField returns the .DataField property.
GetCtlForm returns the name of the form did contains the control.
GetCtlIndex returns the .Index property. If the control has no index, -1 is returned.
GetCtlName returns the .Name of the control.
GetCtlNameIndex returns the name and the of the control. The format is Name(x), if no index => Name is used.
GetCtlPropCaption returns the position of the .Caption property in the definition table of the control.
GetCtlPropDataField returns the position of the .DataField property in the definition table of the control.
GetCtlPropText returns the position of the .Text property in the definition table of the control.
GetCtlTag returns the .Tag property of the control. The returned string is limited to the first chr$(0) founded.
GetCtlTagSized returns the full .Tag property of the control.
GetCtlText returns the .Text property of the control.
GetHwnd returns the .hWnd of the control. If the control has no .hWnd, the returned value is 0.

Declare Syntax :

```
Declare Function cGetCtlCaption Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlClass Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlContainer Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlDataField Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlForm Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlIndex Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetCtlName Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlNameIndex Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlPropCaption Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetCtlPropDataField Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetCtlPropText Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetCtlTag Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlTagSized Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlText Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetHwnd Lib "time2win.dll" (Ctl As Control) 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 : cGetX, cSetX, cSetCtlX

TrueBetween

Purpose :

TrueBetween checks to see if a value is fully between two other values.

Declare Syntax :

Declare Function cTrueBetween Lib "time2win.dll" (Var As Variant, Var1 As Variant, Var2 As Variant) As Integer

Call Syntax :

test = cTrueBetween(var, var1, var2)

Where :

var	value to test
var1	first value
var2	second value
test	TRUE if var is fully between var1 and var2 FALSE if var is not 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 = cTrueBetween(var, var1, var2)
        -> test = TRUE

var = 10
test = cTrueBetween(var, var1, var2)
        -> test = FALSE
```

See Also : cBetween

GetX

Purpose :

The functions below applies to the .hWnd of a custom control.

GetCaption returns the .Caption property.
GetClass returns the class name defined in the properties windows in the design-mode of VB.
GetContainer returns the name of the container did contains the control. The container can be the form or an another control.
GetDataField returns the .DataField property.
GetForm returns the name of the form did contains the control.
GetIndex returns the .Index property. If the control has no index, -1 is returned.
GetNameIndex returns the name and the of the control. The format is Name(x), if no index => Name is used.
GetText returns the .Text property of the control.

Declare Syntax :

```
Declare Function cGetCaption Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetClass Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetContainer Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetDataField Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetForm Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetIndex Lib "time2win.dll" (ByVal hWnd As Integer) As Integer
Declare Function cGetNameIndex Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetText Lib "time2win.dll" (ByVal hWnd As Integer) As String
```

Call Syntax :

The purpose and the declare syntax are very explicite.

Where :

hWnd	the hWnd of the custom control.

Comments :

•The advantage to use these routines is that these routines doesn't generates an error if the property not exists.
•If the custom control doesn't have a .hWnd (Label control b.e.), you must use the cGetCtlX function.

Examples :

See also : cGetCtlX ,cSetX, cSetCtlX

MakePath

Purpose :

MakePath creates 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 : cSplitPath, cFullPath

Uncompact

Purpose :

Uncompact uncompacts a string composed of numeric chars.

Declare Syntax :

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

Call Syntax :

test = cUncompact(Txt)

Where :

Txt	is the string (only numeric chars) to uncompact
test	returns the string uncompacted

Comments :

The size of the returned string is always a multiple of 2.

Examples :

```
Txt = "0123456789"
test = cUncompact(Txt)
        test = "30313233343536373839"
```

See also : cCompact

UniqueFileName

Purpose :

UniqueFileName creates 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"
```

ChangeChars

Purpose :

ChangeChars changes all chars specifien by others chars in a string.

Declare Syntax :

Declare Sub cChangeChars Lib "time2win.dll" (Txt As String, charSet As String, newCharSet As String)

Call Syntax :

Call cChangeChars(Txt, charSet, newCharSet)

Where :

Txt	the string to process
charSet	the chars in the string to be changed
newCharSet	the new chars

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.

Examples :

```
Txt = "ABCDEF"
charSet = "ACE"
newCharSet = "ace"
Call cChangeChars(Txt, charSet, newCharSet)
        Txt = "aBcDeF"
```

See also : cChangeCharsUntil

ChangeCharsUntil

Purpose :

ChangeCharsUntil changes all chars specifien by others chars in a string until a char is encountered.

Declare Syntax :

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, 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.
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 :

```
Txt = "ABCDEF"
charSet = "ACE"
newCharSet = "ace"
nUntil = "D"
Call cChangeCharsUntil(Txt, charSet, newCharSet, nUntil)
        Txt = "aBcDEF"
```

See also : cChangeChars

ChangeTaskName

Purpose :

ChangeTaskName changes 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 Integer, 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 : cGetTaskName, cGetChangeTaskName

EnableFI, DisableFI

Purpose :

EnableFI and DisableFI enables or disables 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 ignores input such as mouse clicks and key presses. When input is enabled, the form processes all input.

Declare Syntax :

```
Declare Sub cEnableFI Lib "time2win.dll" (Ctl As Control)
Declare Sub cDisableFI Lib "time2win.dll" (Ctl As Control)
```

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 enables or disables mouse and keyboard input to the given form by sending a WM_ENABLE message. When input is disabled, the form ignores input such as mouse clicks and key presses. When input is enabled, the form processes all input.

Declare Syntax :

```
Declare Sub cEnableForm Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Sub cDisableForm Lib "time2win.dll" (ByVal hWnd As Integer)
```

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

Purpose :

EnableRedraw and DisableRedraw sends a WM_SETREDRAW message from a hWnd of a control to allow changes in that window to be redrawn or to prevent changes in that window from being redrawn.

EnableCtlRedraw and DisableCtlRedraw sends a WM_SETREDRAW message to a control to allow changes in that window to be redrawn or to prevent changes in that window from being redrawn.

Declare Syntax :

```
Declare Sub cEnableRedraw Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Sub cDisableRedraw Lib "time2win.dll" (ByVal hWnd As Integer)

Declare Sub cEnableCtlRedraw Lib "time2win.dll" (Ctl As Control)
Declare Sub cDisableCtlRedraw Lib "time2win.dll" (Ctl As Control)
```

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.

Fill

Purpose :

Fill fills 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 : cCreateAndFill

KillFocus

Purpose :

KillFocus kills and recreates the focus of a gived hWnd

Declare Syntax :

Declare Sub cKillFocus Lib "time2win.dll" (ByVal hWnd As Integer)

Call Syntax :

Call cKillFocus(hWnd)

Where :

hWnd	the hWnd of the control

Comments :

PutIni

Purpose :

see Comments

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 : cGetIni

ResetFocus

Purpose :

ResetFocus kills the focus of a gived hWnd and set the focus to an another hWnd.

Declare Syntax :

Declare Sub cResetFocus Lib "time2win.dll" (ByVal hWnd1 As Integer, ByVal hWnd2 As Integer)

Call Syntax :

Call cResetFocus(hWnd1, hWnd2)

Where :

hWnd1	the hWnd of the control that you want kill the focus.
hWnd2	the hWnd of the control that you want set the focus.

Comments :

ReverseAllBits

Purpose :

ReverseAllBits reverses all bits in a gived string

Declare Syntax :

Declare Sub cReverseAllBits Lib "time2win.dll" (Txt As String)

Call Syntax :

Call cReverseAllBits(Txt)

Where :

Txt	the string to proceed

Comments :

See also : Bit String Manipulation routines

ReverseAllBitsByChar

Purpose :

ReverseAllBitsByChar reverses all bits by each char in a gived string

Declare Syntax :

Declare Sub cReverseAllBitsByChar Lib "time2win.dll" (Txt As String)

Call Syntax :

Call cReverseAllBitsByChar(Txt)

Where :

Txt	the string to proceed

Comments :

See also : Bit String Manipulation routines

SetAllBits

Purpose :

SetAllBits sets all bits of a gived string to Set state or Reset state.

Declare Syntax :

Declare Sub cSetAllBits Lib "time2win.dll" (Txt As String, ByVal Value As Integer)

Call Syntax :

Call cSetAllBits(Txt, Value)

Where :

Txt	the string to proceed
Value	TRUE to Set all bits FALSE to Reset all bits

Comments :

See also : Bit String Manipulation routines

SetBit

Purpose :

SetBit sets a gived bit in a gived string to Set state or Reset state.

Declare Syntax :

Declare Sub cSetBit Lib "time2win.dll" (Txt As String, ByVal Position As Integer, ByVal Value As Integer)

Call Syntax :

Call cSetBit(Txt, Position, Value)

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.

See also : Bit String Manipulation routines

SetBitToFalse

Purpose :

SetBitToFalse sets a gived bit in a gived string to Reset state.

Declare Syntax :

Declare Sub cSetBitToFalse Lib "time2win.dll" (Txt As String, ByVal Position As Integer)

Call Syntax :

Call cSetBitToFalse(Txt, Position)

Where :

Txt	the string to proceed
Position	the bit position to Reset

Comments :

The first bit in the string is the bit 0. This routine is a short-cut routine from cSetBit(Txt, Position, FALSE)

See also : Bit String Manipulation routines

SetBitToTrue

Purpose :

SetBitToTrue sets a gived bit in a gived string to Set state.

Declare Syntax :

```
Declare Sub cSetBitToTrue Lib "time2win.dll" (Txt As String, ByVal Position As Integer)
```

Call Syntax :

```
Call cSetBitToTrue(Txt, Position)
```

Where :

Txt	the string to proceed
Position	the bit position to Set

Comments :

The first bit in the string is the bit 0. This routine is a short-cut routine from cSetBit(Txt, Position, TRUE)

See also : Bit String Manipulation routines

FileFilter, FileFilterNot

Purpose :

FileFilter copies one file to an another file but filters some chars.
FileFilterNot copies 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 : cFileCopy

SplitPath

Purpose :

SplitPath breaks 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 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	Contains the drive letter followed by a colon (:) if a drive is specified in path.
nDir	Contains the path of subdirectories, if any, including the trailing slash.
nName	Contains the base filename without any extensions.
nExt	Contains 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 : cFullPath, cMakePath, Constants and Types declaration

Revision History

See also : New Features

Version	Comments
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, cAlllSubDirectories 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' data link library for VB 3.0.

New Features

See also : Revision History

Version	Comments
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
	Save an element to a big sized array on disk. cDAPut
	Read a type'd variable from a big sized array on disk.
	cDAGetType
	Save a type'd variable to a big sized array on disk.
	cDAPutType
	Clear a big sized array (fill it with chr$(0)).
	cDAClear
1.22	Modification of a system menu in one call (6 different languages)
	cLngSysMenu
1.21	Multi-Language Message Box (fully replacement of the standard sub MsgBox)
	cLngBoxMsg
	Multi-Language Message Box (fully replacement of the standard function MsgBox)
	cLngMsgBox
	Multi-Language InputBox (fully replacement of the standard function InputBox$)
	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' data link library.

FileCopy

Purpose :

FileCopy copies 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 : cFileFilter, cFileFilterNot, cFileMerge

SetDefaultSeparator

Purpose :

SetDefaultSeparator sets the default char for use the cGet function.

Declare Syntax :

```
Declare Sub cSetDefaultSeparator Lib "time2win.dll" (Separator As String)
```

Call Syntax :

```
Call cSetDefaultSeparator(Separator)
```

Where :

Separator	the new separator

Comments :

The default char is '|'.
This char is changed for all applications did use the TIME2WIN.DLL.
If you must initialize the default, change it only at the starting of your program.

GetSeparatorX

Purpose :

All values returned are readed from the Win.INI file.

GetCountry returns the country name.
GetCountryCode returns the country code.
GetCurrency returns the currency.
GetDateFormat returns the format for the date.
GetDateSeparator returns the separator for the date.
GetHourFormat returns the format for the hour.
GetLanguage returns the letters for the language.
GetListSeparator returns the separator for list.
GetTimeSeparator returns the separator for the date.
GetWinINI returns the information for a gived item (see Constants and Types declaration)

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 GET_TIME_SEPARATOR GET_DATE_SEPARATOR GET_TIME_FORMAT GET_DATE_FORMAT GET_CURRENCY GET_LANGUAGE GET_COUNTRY GET_COUNTRY_CODE GET_LIST_SEPARATOR GET_DEFAULT_PRINTER

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 : cGetIni

Installation

Demonstration version :

The files TIME2WIN.DLL and TIME2WIN.HLP should be copied in your   WINDOWS\SYSTEM directory.

Registered version :

The files TIME2WIN.DLL, TIME2WIN.HLP should be copied in your WINDOWS\SYSTEM directory.
The file TIME2WIN.LIC should be copied in your WINDOWS directory.

Distribution note:

When you create and distribute applications that use 'TIME TO WIN' data link library, you should install the file 'TIME2WIN.DLL' in the customer's Microsoft Windows \SYSTEM 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.*

SetWait, StartWait, CheckWait

Purpose :

SetWait sets the time to wait in a specified timer.
StartWait starts the specified timer.
CheckWait checks 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 : cReadTimer, cStartTimer, cStopTimer, Timer functions

StartBasisTimer, ReadBasisTimer, StopBasisTimer

Purpose :

StartBasisTimer starts the default timer.
ReadBasisTimer reads the value of the default timer.
StopBasisTimer stops 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 : cReadTimer, cStartTimer, cStopTimer, Timer functions

StartTimer, ReadTimer, StopTimer

Purpose :

StartBasisTimer starts the specified timer.
ReadBasisTimer reads the value of the specified timer.
StopBasisTimer stops 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 : cReadBasisTimer, cStartBasisTimer, cStopBasisTimer, Timer functions

SysMenuChange

Purpose :

SysMenuChange changes the name of an item in the system menu of an application.

Declare Syntax :

Declare Sub cSysMenuChange Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Position As Integer, ByVal NewMessage As String)

Call Syntax :

Call cSysMenuChange(hWnd, Position, NewMessage)

Where :

hWnd%	is the .hWnd of the form.
Position%	is the position of the item in the system menu.
NewMessage$	is the new message to set for the specified item.

Comments :

The position starts at offset 0.
Don't forget that some items in the menu are only separators.
This function only changes the message not the fonctionnality.
This function take care of the menu 'grayed'.

Examples :

Change the system menu of a form in French

Call cSysMenuChange(Me.hWnd, 0, "&Restaure") Restore becomes Restaure
Call cSysMenuChange(Me.hWnd, 1, "&Positionne") Move becomes Positionne
Call cSysMenuChange(Me.hWnd, 2, "&Taille") Size becomes Taille
Call cSysMenuChange(Me.hWnd, 3, "&Icône") Minimize becomes Icône
Call cSysMenuChange(Me.hWnd, 4, "&Plein écran") Maximize becomes Plein écran
Call cSysMenuChange(Me.hWnd, 6, "&Fermer" + Chr$(9) + "Alt+F4") Close Alt+F4 becomes Fermer Alt+F4
Call cSysMenuChange(Me.hWnd, 8, "&Tâche..." + Chr$(9) + "Ctrl+Esc")Switch To... Ctrl+Esc becomes Tâche... Ctrl+Esc

See also : cLngSysMenu

FileEncrypt, FileDecrypt

Purpose :

FileEncrypt copies one file to an another file but with encryption.
FileDecrypt copies 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 3 (Constants and Types declaration).
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 filter 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_3)
test& = cFileDecrypt("c:\autoexec.tb1", "c:\autoexec.tb2", "Time To Win", ENCRYPT_LEVEL_3)

See also :

ToggleAllBits

Purpose :

ToggleAllBits toggles 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.

Declare Syntax :

Declare Sub cToggleAllBits Lib "time2win.dll" (Txt As String)

Call Syntax :

Call cToggleAllBits(Txt)

Where :

Txt	the string to proceed

Comments :

See also : Bit String Manipulation routines

ToggleBit

Purpose :

ToggleBit toggles 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 cToggleBit Lib "time2win.dll" (Txt As String, ByVal Position As Integer)

Call Syntax :

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 : Bit String Manipulation routines

Multi-Language support

cLngBoxMsg
cLngInpBox
cLngMsgBox
cReadCtlLanguage
cSaveCtlLanguage

UnloadDLL

Purpose :

UnloadDLL unloads a DLL from the memory.

Declare Syntax :

```
Declare Sub cUnloadDLL Lib "time2win.dll" (ByVal hMod As Integer)
```

Call Syntax :

```
Call cUnloadDLL(hMod)
```

Where :

hModule	is the module handle of the DLL.

Comments :

Use this with care.

Examples :

```
    Dim MODULEENTRY    As tagMODULEENTRY
    Dim Tmp            As String

    Tmp = "LZEXPAND"

    If (cModuleFind(MODULEENTRY, "LZEXPAND") = True) Then

        Call cUnloadDLL(MODULEENTRY.hModule)

        If (cModuleFind(MODULEENTRY, Tmp) = False) Then
            MsgBox Tmp + " has been UnLoaded."
        Else
            MsgBox Tmp + " can't be UnLoaded."
        End If

    Else

        MsgBox Tmp + " not found in memory."

    End If
```

```
On my system :   after running one time :           LZEXPAND has been Unloaded."
                 after running a second time :            LZEXPAND not found in memory."
```

CmpFileAttribute, CmpFileContents, CmpFileSize, CmpFileTime

Purpose :

CmpFileAttribute compares the attribute of two files.
CmpFileContents compares the contents of two files.
CmpFileSize compares the size of two files.
CmpFileTime compares 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 :

All Functions and Subs

```
Declare Function cAddD Lib "time2win.dll" (array() As Double, ByVal nValue As Double) As Integer
Declare Function cAddDigit Lib "time2win.dll" (Txt as string) 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
Declare Function cAddTime Lib "time2win.dll" (ByVal Hr As Integer) As Integer
Declare Function cAllSubDirectories Lib "time2win.dll" (ByVal lpBaseDirectory As String, nDir As Integer) As String
Declare Function cArabicToRoman Lib "time2win.dll" (Var As Variant) As String
Declare Function cArrayPrm Lib "time2win.dll" (array() As Any, nArray As Any) As Integer
Declare Function cBaseConversion Lib "time2win.dll" (ByVal Num As String, ByVal RadixIn As Integer, ByVal
RadixOut As Integer) As String
Declare Function cBetween Lib "time2win.dll" (Var As Variant, Var1 As Variant, Var2 As Variant) As Integer
Declare Function cBigAdd Lib "time2win.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigDiv Lib "time2win.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigMul Lib "time2win.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigNum Lib "time2win.dll" (ByVal n1 As String, ByVal op As Integer, ByVal n2 As String) As String
Declare Function cBigSub Lib "time2win.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigFmt Lib "time2win.dll" (Num As String, ByVal Fmt As Integer) As String
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 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)
Declare Sub cChangeTaskName Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Function cChDir Lib "time2win.dll" (ByVal lpDir As String) As Integer
Declare Function cChDrive Lib "time2win.dll" (ByVal lpDrive As String) As Integer
Declare Function cCheckChars Lib "time2win.dll" (Txt As String, charSet As String) As Integer
Declare Function cCheckNumericity Lib "time2win.dll" (Txt As String) As Integer
Declare Function cCheckTime Lib "time2win.dll" (ByVal Hr As Integer, ByVal Hr1 As Integer, ByVal Hr2 As Integer) As
Integer
Declare Function cCheckWait Lib "time2win.dll" (ByVal nTimer As Integer) As Integer
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
Declare Function cCompact Lib "time2win.dll" (Txt As String) As String
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 Function cCompress Lib "time2win.dll" (Txt As String) As String
Declare Function cCompressTab Lib "time2win.dll" (Txt As String, ByVal nTab As Integer) As String
Declare Function cCount Lib "time2win.dll" (Txt As String, Separator As String) As Integer
Declare Function cCountDirectories Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cCountFiles Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cCplAlpha Lib "time2win.dll" (Txt As String) As String
Declare Function cCplDigit Lib "time2win.dll" (Txt As String) As String
Declare Function cCreateAndFill Lib "time2win.dll" (ByVal Length As Integer, Txt As String) As String
Declare Function cCreateBits Lib "time2win.dll" (ByVal nBits As Integer) As String
Declare Function cCurrentTime Lib "time2win.dll" () As Integer
Declare Function cCVB Lib "time2win.dll" (Value As String) As Integer
Declare Function cCVC Lib "time2win.dll" (Value As String) As Currency
Declare Function cCVD Lib "time2win.dll" (Value As String) As Double
Declare Function cCVI Lib "time2win.dll" (Value As String) As Integer
Declare Function cCVL Lib "time2win.dll" (Value As String) As Long
Declare Function cCVS Lib "time2win.dll" (Value As String) As Single
Declare Function cDAClear Lib "time2win.dll" (DISKARRAY As tagDISKARRAY) As Integer
Declare Sub cDAClose Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal DeleteFile As Integer)
Declare Function cDACreate Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal CreateOrUse As Integer) As Integer
Declare Function cDAGet Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long) As Variant
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 cDAPut Lib "time2win.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, Var As Variant)
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 Function cDaysInMonth Lib "time2win.dll" (ByVal nYear As Integer, ByVal nMonth As Integer) As Integer
Declare Function cDecrypt Lib "time2win.dll" (Txt As String, password As String, ByVal level As Integer) As String
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
Declare Sub cDisableCtlRedraw Lib "time2win.dll" (Ctl As Control)
Declare Sub cDisableFI Lib "time2win.dll" (Ctl As Control)
Declare Sub cDisableForm Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Sub cDisableRedraw Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Sub cEnableCtlRedraw Lib "time2win.dll" (Ctl As Control)
Declare Sub cEnableFI Lib "time2win.dll" (Ctl As Control)
Declare Sub cEnableForm Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Sub cEnableRedraw Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Function cEncrypt Lib "time2win.dll" (Txt As String, password As String, ByVal level As Integer) As String
Declare Function cEXEnameActiveWindow Lib "time2win.dll" () As String
Declare Function cEXEnameTask Lib "time2win.dll" (ByVal nFileName As String) As String
Declare Function cEXEnameWindow Lib "time2win.dll" (ByVal hModule As Integer) As String
Declare Function cExitWindowsAndExecute Lib "time2win.dll" (ByVal lpszExe As String, ByVal lpszParams As String) As Integer
Declare Function cExpandTab Lib "time2win.dll" (Txt As String, ByVal nTab As Integer) As String
Declare Function cFileCompressTab Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal nTab As Integer) As Long
Declare Function cFileCopy Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String) As Long
Declare Function cFileCRC32 Lib "time2win.dll" (ByVal lpFilename As String, ByVal mode As Integer) As Long
Declare Function cFileDateCreated Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileDecrypt Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal password As String, ByVal level As Integer) As Long
Declare Function cFileDrive Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileEncrypt Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal password As String, ByVal level As Integer) As Long
Declare Function cFileExpandTab Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal nTab As Integer) As Long
Declare Function cFileFilter Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal Filter As String) As Long
Declare Function cFileFilterNot Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal Filter As String) As Long
Declare Function cFileGetAttrib Lib "time2win.dll" (ByVal nFilename As String, nFileAttribute As Any) As Integer
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 cFileLastTimeAccess Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastTimeModified Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cFileLineCount Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cFileMerge Lib "time2win.dll" (ByVal file1 As String, ByVal file2 As String, ByVal fileTo As String) As Long
Declare Function cFilePathExists Lib "time2win.dll" (ByVal lpFilename As String) As Integer
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 cFileResetFlag Lib "time2win.dll" (ByVal nFilename As String, ByVal nStatus As Integer) 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 cFileSearch Lib "time2win.dll" (ByVal nFileName As String, ByVal Search As String, ByVal sensitivity As Integer) As Long
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 Integer
Declare Function cFileSearchCount Lib "time2win.dll" (ByVal nFileName As String, ByVal Search As String, ByVal sensitivity As Integer) As Long
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 cFileSetAttrib Lib "time2win.dll" (ByVal nFilename As String, nFileAttribute As Any) As Integer
Declare Function cFileSetFlag Lib "time2win.dll" (ByVal nFilename As String, ByVal nStatus As Integer) 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 cFilesInDirectory Lib "time2win.dll" (ByVal nFilename As String, ByVal firstnext As Integer) As String
Declare Function cFileSize Lib "time2win.dll" (ByVal lpFilename As String) As Long
Declare Function cFilesSize Lib "time2win.dll" (ByVal nFilename As String) As Long
Declare Function cFilesSizeOnDisk Lib "time2win.dll" (ByVal nDrive As String, ByVal nFileName As String) As Long
Declare Function cFilesSlack Lib "time2win.dll" (ByVal nDrive As String, ByVal nFileName As String, Size1 As Long, Size2 As Long) As Integer
Declare Function cFileStatistics Lib "time2win.dll" (ByVal nFilename As String, nLines As Long, nWords As Long, nChars As Long) As Long
Declare Function cFileTimeCreated Lib "time2win.dll" (ByVal lpFilename As String) As String
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
Declare Sub cFill Lib "time2win.dll" (Txt As String, Fill As String)
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
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
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
Declare Function cFindFileInEnv Lib "time2win.dll" (ByVal lpFilename As String, ByVal lpEnv As String) As Integer
Declare Function cFindFileInPath Lib "time2win.dll" (ByVal lpFilename As String) As Integer
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 cFromHexa Lib "time2win.dll" (Text As String) As String
Declare Function cFullPath Lib "time2win.dll" (ByVal nFilename As String) As String
Declare Function cGet Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cGetAscTime Lib "time2win.dll" (ByVal nLanguage As Integer) As String
Declare Function cGetBit Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As Integer
Declare Function cGetBlock Lib "time2win.dll" (Txt As String, ByVal Position As Integer, ByVal Length As Integer) As String
Declare Function cGetCaption Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetChangeTaskName Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String) As String
Declare Function cGetClass Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetClassName Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetContainer Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetCountry Lib "time2win.dll" () As String
Declare Function cGetCountryCode Lib "time2win.dll" () As String
Declare Function cGetCtlCaption Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlClass Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlContainer Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlDataField Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlForm Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlIndex Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetCtlName Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlNameIndex Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlPropCaption Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetCtlPropDataField Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetCtlPropText Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetCtlTag Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlTagSized Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCtlText Lib "time2win.dll" (Ctl As Control) As String
Declare Function cGetCurrency Lib "time2win.dll" () As String
Declare Function cGetCurrentDrive Lib "time2win.dll" () As String
Declare Function cGetDataField Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetDateFormat Lib "time2win.dll" () As String
Declare Function cGetDateSeparator Lib "time2win.dll" () As String
Declare Function cGetDefaultCurrentDir Lib "time2win.dll" () As String
Declare Function cGetDefaultPrinter Lib "time2win.dll" () As String
Declare Function cGetDevices Lib "time2win.dll" () As String
Declare Function cGetDiskClusterSize Lib "time2win.dll" (ByVal lpDrive As String) As Long
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 cGetDriveCurrentDir Lib "time2win.dll" (ByVal lpDrive As String) As String
Declare Function cGetDriveType Lib "time2win.dll" (ByVal lpDrive As String) As Integer
Declare Function cGetFileVersion Lib "time2win.dll" (ByVal filename As String, ByVal nFonction As Integer) As String
Declare Function cGetFileVersionInfo Lib "time2win.dll" (ByVal filename As String, FILEVERSIONINFO As Any) As
Integer
Declare Function cGetForm Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetFullNameInEnv Lib "time2win.dll" (ByVal lpFilename As String, ByVal lpEnv As String) As String
Declare Function cGetFullNameInPath Lib "time2win.dll" (ByVal lpFilename As String) As String
Declare Function cGetHourFormat Lib "time2win.dll" () As String
Declare Function cGetHwnd Lib "time2win.dll" (Ctl As Control) As Integer
Declare Function cGetIn Lib "time2win.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetIndex Lib "time2win.dll" (ByVal hWnd As Integer) As Integer
Declare Function cGetIni Lib "time2win.dll" (ByVal AppName As String, ByVal szItem As String, ByVal szDefault As
String, ByVal InitFile As String) As String
Declare Function cGetLanguage Lib "time2win.dll" () As String
Declare Function cGetListSeparator Lib "time2win.dll" () As String
Declare Function cGetLongDay Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetLongMonth Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As
String
Declare Function cGetName Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetNameIndex Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetNetConnection Lib "time2win.dll" (ByVal lpDrive As String, ErrCode As Integer) As String
Declare Function cGetPid Lib "time2win.dll" () As Integer
Declare Function cGetPrinterPorts Lib "time2win.dll" () As String
Declare Function cGetSectionItems Lib "time2win.dll" (ByVal Section As String, ByVal InitFile As String, nItems 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 cGetShortMonth Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As
String
Declare Function cGetSystemDirectory Lib "time2win.dll" () As String
Declare Function cGetTaskName Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetText Lib "time2win.dll" (ByVal hWnd As Integer) As String
Declare Function cGetTimeSeparator Lib "time2win.dll" () As String
Declare Function cGetTinyDay 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 cGetWindowsDirectory Lib "time2win.dll" () As String
Declare Function cGetWinINI Lib "time2win.dll" (ByVal Info As Integer) As String
Declare Function cGetWinSection Lib "time2win.dll" (ByVal Section As String) As String
Declare Function cGiveBitPalindrome Lib "time2win.dll" () As String
Declare Function cHourTo Lib "time2win.dll" (Txt As String) As Variant
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
Declare Function cIntoBalance Lib "time2win.dll" (Var As Variant) As String
Declare Function cIntoBalanceFill Lib "time2win.dll" (Var As Variant) As String
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
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
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 cIsBalance Lib "time2win.dll" (ByVal nHour As Long, ByVal nMinute As Integer, ByVal nSecond As
Integer) As Integer
Declare Function cIsBitPalindrome 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 cIsDate Lib "time2win.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As
Integer) As Integer
Declare Function cIsDigit Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsFileArchive 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
Declare Function cIsFileHidden Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileNormal Lib "time2win.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFilenameValid 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 cIsFormEnabled Lib "time2win.dll" (ByVal hWnd 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 cIsISBN Lib "time2win.dll" (Txt As String) As Integer
Declare Function cIsLeapYear Lib "time2win.dll" (ByVal nYear As Integer) 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 cKillDir Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillDirFilesAll Lib "time2win.dll" (ByVal lpDir As String, ByVal lpMask As String) As Integer
Declare Function cKillDirs Lib "time2win.dll" (ByVal lpDir As String, ByVal HeaderDirectory As Integer) As Integer
Declare Function cKillFile Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFileAll Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFiles Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFilesAll Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Sub cKillFocus Lib "time2win.dll" (ByVal hWnd As Integer)
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 cLngInpBox Lib "time2win.dll" (ByVal nLanguage As Integer, ByVal Message As String, ByVal Title
As String, ByVal Default As String) 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
Declare Function cLrc Lib "time2win.dll" (Txt As String) As String
Declare Function cMakeDir Lib "time2win.dll" (ByVal lpFilename As String) As Integer
Declare Function cMakePath Lib "time2win.dll" (ByVal nDrive As String, ByVal nDir As String, ByVal nFilename As
String, ByVal Ext As String) As String
Declare Function cMax Lib "time2win.dll" (Var1 As Variant, Var2 As Variant) As Variant
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
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
Declare Function cMin Lib "time2win.dll" (Var1 As Variant, Var2 As Variant) As Variant
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
Declare Function cMixChars Lib "time2win.dll" (Txt As String) As String
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 cMKN Lib "time2win.dll" (ByVal Value As Double) As String
Declare Function cMKS Lib "time2win.dll" (ByVal Value As Single) As String
Declare Function cModuleFind Lib "time2win.dll" (MODULEENTRY As Any, ByVal ModuleName As String) As Integer
Declare Function cModules Lib "time2win.dll" (MODULEENTRY As Any, ByVal firstnext As Integer) As Integer
Declare Function cMorse Lib "time2win.dll" (ByVal morse As String) As String
Declare Function cNextHwnd Lib "time2win.dll" (ByVal hWnd As Integer) As Integer
Declare Function cNumDigit Lib "time2win.dll" (Txt as string) As integer
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
Declare Function cPatternExtMatch Lib "time2win.dll" (ByVal Txt As String, ByVal Pattern As String) As Integer
Declare Function cPatternMatch Lib "time2win.dll" (ByVal Txt As String, ByVal Pattern As String) As Integer
Declare Sub cPutIni Lib "time2win.dll" (ByVal AppName As String, ByVal szItem As String, ByVal szDefault As String, ByVal InitFile As String)
Declare Function cReadBasisTimer Lib "time2win.dll" () As Long
Declare Function cReadCtlLanguage Lib "time2win.dll" (Ctl As Control, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
Declare Function cReadTimer Lib "time2win.dll" (ByVal nTimer As Integer) As Long
Declare Function cRebootSystem Lib "time2win.dll" () As Integer
Declare Function cRemoveBlockChar Lib "time2win.dll" (Txt As String, ByVal Position As Integer, ByVal Length As Integer) As String
Declare Function cRemoveOneChar Lib "time2win.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cRenameFile Lib "time2win.dll" (ByVal lpFilename1 As String, ByVal lpFilename2 As String) As Integer
Declare Sub cResetCapture Lib "time2win.dll" ()
Declare Sub cResetFocus Lib "time2win.dll" (ByVal hWnd1 As Integer, ByVal hWnd2 As Integer)
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
Declare Function cRestartWindows Lib "time2win.dll" () As Integer
Declare Function cReverse Lib "time2win.dll" (Txt As String) As String
Declare Sub cReverseAllBits Lib "time2win.dll" (Txt As String)
Declare Sub cReverseAllBitsByChar Lib "time2win.dll" (Txt As String)
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
Declare Function cRomanToArabic Lib "time2win.dll" (Txt As String) As Variant
Declare Function cSaveCtlLanguage Lib "time2win.dll" (Ctl As Control, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
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)
Declare Sub cSetCaption Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetCapture Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Sub cSetCtlCaption Lib "time2win.dll" (Ctl As Control, ByVal Text As String)
Declare Sub cSetCtlDataField Lib "time2win.dll" (Ctl As Control, ByVal Text As String)
Declare Sub cSetCtlFocus Lib "time2win.dll" (Ctl As Control)
Declare Sub cSetCtlPropString Lib "time2win.dll" (Ctl As Control, ByVal PropIndex As Integer, ByVal Text As String)
Declare Sub cSetCtlTag Lib "time2win.dll" (Ctl As Control, ByVal Text As String)
Declare Sub cSetCtlText Lib "time2win.dll" (Ctl As Control, ByVal Text As String)
Declare Function cSetD Lib "time2win.dll" (array() As Double, ByVal nValue As Double) As Integer
Declare Sub cSetDataField Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetDefaultSeparator Lib "time2win.dll" (Separator As String)
Declare Sub cSetFocus Lib "time2win.dll" (ByVal hWnd As Integer)
Declare Function cSetHandleCount Lib "time2win.dll" (ByVal nHandle As Integer) 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
Declare Sub cSetTag Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetText Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetWait Lib "time2win.dll" (ByVal nTimer As Integer, ByVal nValue As Long)
Declare Function cSleep Lib "time2win.dll" (ByVal Delay As Long) As Integer
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
Declare Sub cSplitPath Lib "time2win.dll" (ByVal nFilename As String, SPLITPATH As Any)
Declare Sub cStartBasisTimer Lib "time2win.dll" ()
Declare Sub cStartTimer Lib "time2win.dll" (ByVal nTimer As Integer)
Declare Sub cStartWait Lib "time2win.dll" (ByVal nTimer As Integer)
Declare Sub cStopBasisTimer Lib "time2win.dll" ()
Declare Function cStopTimer Lib "time2win.dll" (ByVal nTimer As Integer) As Long
Declare Function cStringCRC32 Lib "time2win.dll" (Txt As String) As Long
Declare Sub cStringToType Lib "time2win.dll" Alias "cTypesCopy" (ByVal Src As String, TypeDst As Any, ByVal
lenTypeSrc As Integer)
Declare Function cSubDirectory Lib "time2win.dll" (ByVal nFilename As String, ByVal firstnext As Integer) As String
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
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)
Declare Sub cSysMenuChange Lib "time2win.dll" (ByVal hWnd As Integer, ByVal Position As Integer, ByVal
NewMessage As String)
Declare Function cTaskFind Lib "time2win.dll" (TASKENTRY As Any, ByVal hTask As Integer) As Integer
Declare Function cTasks Lib "time2win.dll" (TASKENTRY As Any, ByVal firstnext As Integer) As Integer
Declare Function cTimeBetween Lib "time2win.dll" (ByVal Hr1 As Integer, ByVal Hr2 As Integer) As Integer
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
Declare Sub cToggleAllBits Lib "time2win.dll" (Txt As String)
Declare Sub cToggleBit Lib "time2win.dll" (Txt As String, ByVal Position As Integer)
Declare Function cToHexa Lib "time2win.dll" (Text As String) As String
Declare Function cTrueBetween Lib "time2win.dll" (Var As Variant, Var1 As Variant, Var2 As Variant) 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 Function cTypesCompare Lib "time2win.dll" (Type1 As Any, Type2 As Any, ByVal lenType1 As Integer) As
Integer
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 cTypeToString Lib "time2win.dll" Alias "cTypesCopy" (TypeSrc As Any, ByVal Dst As String, ByVal lenTypeSrc As Integer)
Declare Function cUncompact Lib "time2win.dll" (Txt As String) As String
Declare Function cUniqueFileName Lib "time2win.dll" (Txt As String) As String
Declare Sub cUnloadDLL Lib "time2win.dll" (ByVal hMod As Integer)

Get.x.Day, Get.x.Month

Purpose :

GetTinyDay returns the specified day into one letter.
GetSmallDay returns the specified day into two letters.
GetShortDay returns the specified day into three letters.
GetLongDay returns the specified day into full day name.
GetTinyMonth returns the specified month into one letter.
GetShortMonth returns the specified month into three letters.
GetLongMonth returns 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 language number defined in Constants and Types declaration. 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 : cGetAscTime

Array routines

Adding a value to all elements in a single array

cAddD cAddI cAddL cAddS

Read the configuration of a single array

cArrayPrm

Calculating the standard deviation from all elements in a single array

cDeviationD cDeviationI cDeviationL cDeviationS

Filling on all elements on a single array with a value incremented by one for any element

cFillD cFillI cFillL cFillS

Finding the maximum value in a single array

cMaxD cMaxI cMaxL cMaxS

Calculating the mean from all elements in a single array

cMeanD cMeanI cMeanL cMeanS

Finding the minimum value in a single array

cMinD cMinI cMinL cMinS

Sort a single array in descending order

cReverseSortD cReverseSortI cReverseSortL cReverseSortS cReverseSortStr

Setting all elements in a single array with the same value

cSetD cSetI cSetL cSetS

Sort a single array in ascending order

cSortD cSortI cSortL cSortS cSortStr

Add all elements from a single array

cSumD cSumI cSumL cSumS

Bit String Manipulation routines

All strings used in these functions can be have embedded chr$(0) (if needed). These functions use the full description of a VB string.

cCreateBits
cFindBitReset
cFindBitSet
cGetBit
cGiveBitPalindrome
cIsBitPalindrome
cReverseAllBits
cReverseAllBitsByChar
cSetAllBits
cSetBit
cSetBitToFalse
cSetBitToTrue
cToggleAllBits
cToggleBit

DOS routines

cGetDiskSpace
cGetDiskUsed
cGetDriveCurrentDir
cGetDriveType
cGetFullNameInEnv
cGetFullNameInPath
cGetNetConnection
cIsFileArchive
cIsFileFlag
cIsFileHidden
cIsFileNormal
cIsFileReadOnly
cIsFileSubDir
cIsFileSystem
cIsFileVolId
cKillDir
cKillDirFilesAll
cKillDirs
cKillFile
cKillFileAll
cKillFiles
cKillFilesAll
cMakeDir
cMakePath
cRenameFile
cSplitPath
cSubDirectory
cUniqueFileName

IsX Family Test routines

cIsAlnum
cIsAlpha
cIsAscii
cIsBalance
cIsBitPalindrome
cIsCsym
cIsCsymf
cIsDate
cIsDigit
cIsFileArchive
cIsFileFlag
cIsFileHidden
cIsFilenameValid
cIsFileNormal
cIsFileReadOnly
cIsFileSubDir
cIsFileSystem
cIsFileVolId
cIsFormEnabled
cIsHour
cIsISBN
cIsLeapYear
cIsLower
cIsPalindrome
cIsPunct
cIsSpace
cIsUpper
cIsXdigit

String Manipulation routines

All strings used in these functions can be have embedded chr$(0) (if needed). These functions use the full description of a VB string.

Timer functions

cCheckWait
cReadBasisTimer
cReadTimer
cSetWait
cSleep
cStartBasisTimer
cStartTimer
cStartWait
cStopBasisTimer
cStopTimer

Type functions

cCompareStringType
cCompareTypeString
cStringToType
cTypeClear
cTypeMid
cTypesCompare
cTypesCopy
cTypeToString
cTypeTransfert

VB Control Specific routines

cDisableCtlRedraw
cDisableFI
cDisableForm
cDisableRedraw
cEnableCtlRedraw
cEnableFI
cEnableForm
cEnableRedraw
cGetCaption
cGetClass
cGetContainer
cGetCtlCaption
cGetCtlClass
cGetCtlContainer
cGetCtlDataField
cGetCtlForm
cGetCtlIndex
cGetCtlName
cGetCtlNameIndex
cGetCtlPropCaption
cGetCtlPropDataField
cGetCtlPropText
cGetCtlTag
cGetCtlTagSized
cGetCtlText
cGetDataField
cGetForm
cGetHwnd
cGetIndex
cGetName
cGetNameIndex
cGetText
cKillFocus
cResetCapture
cResetFocus
cSetCaption
cSetCapture
cSetCtlCaption
cSetCtlDataField
cSetCtlFocus
cSetCtlPropString
cSetCtlTag
cSetCtlText
cSetDataField
cSetFocus
cSetTag
cSetText

Windows Specific routines

cChangeTaskName
cEXEnameActiveWindow
cEXEnameTask
cEXEnameWindow
cExitWindowsAndExecute
cGetChangeTaskName
cGetClassName
cGetCountry
cGetCountryCode
cGetCurrency
cGetDateFormat
cGetDateSeparator
cGetDefaultCurrentDir
cGetDefaultPrinter
cGetDevices
cGetFileVersion
cGetFileVersionInfo
cGetHourFormat
cGetIni
cGetLanguage
cGetListSeparator
cGetPrinterPorts
cGetSectionItems
cGetSystemDirectory
cGetTaskName
cGetTimeSeparator
cGetWindowsDirectory
cGetWinINI
cGetWinSection
cModuleFind
cModules
cPutIni
cRebootSystem
cRestartWindows
cTaskFind
cTasks
cUnloadDLL

Constants and Types declaration

```
Global Const GET_TIME_SEPARATOR = 1
Global Const GET_DATE_SEPARATOR = 2
Global Const GET_TIME_FORMAT = 3
Global Const GET_DATE_FORMAT = 4
Global Const GET_CURRENCY = 5
Global Const GET_LANGUAGE = 6
Global Const GET_COUNTRY = 7
Global Const GET_COUNTRY_CODE = 8
Global Const GET_LIST_SEPARATOR = 9
Global Const GET_DEFAULT_PRINTER = 10

Global Const DRIVE_UNKNOW = 0
Global Const DRIVE_REMOVABLE = 2
Global Const DRIVE_FIXED = 3
Global Const DRIVE_REMOTE = 4
Global Const DRIVE_CDROM = 20

Global Const A_NORMAL = &H0                 'Normal file - No read/write restrictions
Global Const A_RDONLY = &H1                 'Read only file
Global Const A_HIDDEN = &H2         'Hidden file
Global Const A_SYSTEM = &H4                 'System file
Global Const A_VOLID = &H8          'Volume ID file
Global Const A_SUBDIR = &H10        'Subdirectory
Global Const A_ARCH = &H20          'Archive file

Global Const ENCRYPT_LEVEL_0 = 0
Global Const ENCRYPT_LEVEL_1 = 1
Global Const ENCRYPT_LEVEL_2 = 2
Global Const ENCRYPT_LEVEL_3 = 3

Global Const OPEN_MODE_BINARY = 0
Global Const OPEN_MODE_TEXT = 1

Global Const BIG_ADD = 0
Global Const BIG_SUB = 1
Global Const BIG_MUL = 2

Global Const VER_VERSION_PRODUCT = -1
Global Const VER_VERSION_FILE = 0
Global Const VER_COMPANY_NAME = 1
Global Const VER_FILE_DESCRIPTION = 2
Global Const VER_FILE_VERSION = 3
Global Const VER_INTERNAL_NAME = 4
Global Const VER_LEGAL_COPYRIGHT = 5
Global Const VER_LEGAL_TRADEMARKS = 6
Global Const VER_PRODUCT_NAME = 7
Global Const VER_PRODUCT_VERSION = 8

Global Const LNG_FRENCH = 1
Global Const LNG_DUTCH = 2
Global Const LNG_GERMAN = 3
Global Const LNG_ENGLISH = 4
Global Const LNG_ITALIAN = 5
Global Const LNG_SPANISH = 6

Global Const MB_MESSAGE_LEFT = 0
Global Const MB_MESSAGE_CENTER = 8192
Global Const MB_MESSAGE_RIGHT = 16384
Global Const MB_TIMEOUT_2 = 32768
Global Const MB_TIMEOUT_4 = 2 * MB_TIMEOUT_2
Global Const MB_TIMEOUT_8 = 2 * MB_TIMEOUT_4
Global Const MB_TIMEOUT_16 = 2 * MB_TIMEOUT_8

Global Const MB_TIMEOUT_6 = MB_TIMEOUT_2 Or MB_TIMEOUT_4
Global Const MB_TIMEOUT_10 = MB_TIMEOUT_2 Or MB_TIMEOUT_8
Global Const MB_TIMEOUT_12 = MB_TIMEOUT_4 Or MB_TIMEOUT_8
Global Const MB_TIMEOUT_14 = MB_TIMEOUT_2 Or MB_TIMEOUT_4 Or MB_TIMEOUT_8
Global Const MB_TIMEOUT_18 = MB_TIMEOUT_2 Or MB_TIMEOUT_16
Global Const MB_TIMEOUT_20 = MB_TIMEOUT_4 Or MB_TIMEOUT_16
Global Const MB_TIMEOUT_22 = MB_TIMEOUT_2 Or MB_TIMEOUT_4 Or MB_TIMEOUT_16
Global Const MB_TIMEOUT_24 = MB_TIMEOUT_8 Or MB_TIMEOUT_16
Global Const MB_TIMEOUT_26 = MB_TIMEOUT_2 Or MB_TIMEOUT_8 Or MB_TIMEOUT_16
Global Const MB_TIMEOUT_28 = MB_TIMEOUT_4 Or MB_TIMEOUT_8 Or MB_TIMEOUT_16
Global Const MB_TIMEOUT_30 = MB_TIMEOUT_2 Or MB_TIMEOUT_4 Or MB_TIMEOUT_8 Or MB_TIMEOUT_16

Global Const MB_DISPLAY_TIMEOUT = 524288

Global Const RS_CAPTION = 1
Global Const RS_TEXT = 2
Global Const RS_DATAFIELD = 4
Global Const RS_DATASOURCE = 8

Global Const MATCH_HEXA = 17
Global Const MATCH_INTERNAL_ERROR = 16
Global Const MATCH_PATTERN = 15
Global Const MATCH_LITERAL = 14
Global Const MATCH_RANGE = 13
Global Const MATCH_ABORT = 12
Global Const MATCH_END = 11
Global Const MATCH_VALID = -1

Global Const PATTERN_VALID = 0
Global Const PATTERN_INVALID = 1
Global Const PATTERN_ESC = 2
Global Const PATTERN_RANGE = 3
Global Const PATTERN_CLOSE = 4
Global Const PATTERN_EMPTY = 5
Global Const PATTERN_INTERNAL_ERROR = 6
Global Const PATTERN_HEXA = 7

Global Const IFV_ERROR = 0
Global Const IFV_NAME_TOO_LONG = 1
Global Const IFV_EXT_TOO_LONG = 2
Global Const IFV_TOO_MANY_BACKSLASH = 3
Global Const IFV_BAD_DRIVE_LETTER = 4
Global Const IFV_BAD_COLON_POS = 5
Global Const IFV_EXT_WITHOUT_NAME = 6

Global Const DA_BYTE = 1
Global Const DA_INTEGER = -2
Global Const DA_LONG = -3
Global Const DA_SINGLE = -4
Global Const DA_DOUBLE = -5
Global Const DA_CURRENCY = -6

Global Const DA_NO_ERROR = True
Global Const DA_EMPTY_FILENAME = 1
Global Const DA_BAD_FILENAME = 2
Global Const DA_CAN_KILL_FILE = 3
Global Const DA_CAN_NOT_OPEN_FILE = 4
Global Const DA_FILE_NOT_FOUND = 5
Global Const DA_BAD_TYPE = 6
Global Const DA_BAD_ROWS = 7
Global Const DA_BAD_COLS = 8
Global Const DA_BAD_SHEETS = 9
Global Const DA_CAN_NOT_WRITE_HEADER = 10
Global Const DA_CAN_NOT_WRITE_PART = 11
Global Const DA_CAN_NOT_WRITE_REMAIN = 12
Global Const DA_CAN_NOT_READ_HEADER = 13
Global Const DA_HEADER_SIZE = 14
Global Const DA_BAD_SIGNATURE = 15
Global Const DA_FILE_SIZE_MISMATCH = 16

Type tagSPLITPATH
        nDrive                          As String
        nDir                            As String
        nName                           As String
        nExt                            As String
End Type

Type tagFILEVERSIONINFO
        VersionProduct          As String
        VersionFile                     As String
        CompanyName             As String
        FileDescription         As String
        FileVersion                     As String
        InternalName                    As String
        LegalCopyright          As String
        LegalTrademarks         As String
        Comments                        As String
        ProductName             As String
        ProductVersion          As String
End Type

Type FileAttributeType
        ErrNo                           As Integer
        Archive                         As Integer
        Hidden                          As Integer
        Normal                          As Integer
        ReadOnly                        As Integer
        SubDir                          As Integer
        System                          As Integer
        VolId                           As Integer
End Type

Type ArrayType
        Bounds                          As Long
        LBound                          As Integer
        UBound                          As Integer
        ElemSize                        As Integer
        IndexCount                      As Integer
        TotalElem                       As Integer
End Type

Type tagMODULEENTRY
        dwSize                          As Long
        szModule                        As String * 10
        hModule                         As Integer
        wcUsage                         As Integer
        szExePath                       As String * 256
        wNext                           As Integer
End Type
```

```
Type tagTASKENTRY
       dwSize                      As Long
       hTask                       As Integer
       hTaskParent                 As Integer
       hInst                As Integer
       hModule                     As Integer
       wSS                         As Integer
       wSP                         As Integer
       wStackTop                   As Integer
       wStackMinimum        As Integer
       wStackBottom         As Integer
       wcEvents                    As Integer
       hQueue                      As Integer
       szModule                    As String * 10
       wPSPOffset                  As Integer
       hNext                       As Integer
End Type

Type tagDISKARRAY
       daSize                      As Integer            'size of the type'd
       Signature                   As String * 7         'signature
       nFilename                   As String * 64   'name of the file
       nType                       As Integer            'variable type
       nRows                       As Long               'number of rows
       nCols                       As Long               'number of cols
       nSheets                     As Long               'number of sheets
       rHandle                     As Integer            'returned handle for use with other functions
       rElementSize                As Integer            'returned size of a element
       rFileSize            As Long               'returned size of the file
       rParts                      As 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 take for the last correct transaction
       dummy                       As String * 9         'reserved for future use
End Type
```

EXEnameActiveWindow

Purpose :

EXEnameActiveWindow retrieves 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:\WINDOWS\VB\VB.EXE"

See also : cEXEnameTask, cEXEnameWindow

EXEnameWindow

Purpose :

EXEnameActiveWindow retrieves 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:\WINDOWS\VB\VB.EXE"

See also : cEXEnameTask, cEXEnameActiveWindow

EXEnameTask

Purpose :

The EXEnameTask function retrieves 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:\WINDOWS\PROGMAN.EXE"

See also : cEXEnameWindow, cEXEnameActiveWindow

Date, Hour and Time routines

IEEE Conversion routines

cCVB
cCVC
cCVD
cCVI
cCVL
cCVS

cMKB
cMKC
cMKD
cMKI
cMKL
cMKN
cMKS

Miscellaneous routines

- cAddDigit
- cBaseConversion
- cBetween
- cCplAlpha
- cCplDigit
- cCurrentTime
- cFileCRC32
- cGetPid
- cLrc
- cMax
- cMin
- cMorse
- cNumDigit
- cSetHandleCount
- cStringCRC32
- cSwapD
- cSwapI
- cSwapL
- cSwapS
- cSwapStr
- cSysMenuChange
- cTrueBetween

Technical Support

Only registered users can receive support and update.

To receive support, you must specify your registration ID.

The following information may be of help to you in streamlining your efforts to resolve any technical problems you may have with 'TIME TO WIN' data link library for Visual Basic® 3.0 for Windows®.

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.

CompuServe Mail:

Name :   Michaël RENARD
CIS :   100042,3646

I'm on CompuServe one time a day (after 19 o'clock European Time).

Days and Months in different language

cGetAscTime
cGetTinyDay
cGetSmallDay
cGetShortDay
cGetLongDay
cGetTinyMonth
cGetShortMonth
cGetLongMonth

License Agreement

Acknowledgement

Thanks to Andreas Thoele for some translations in German.
Thanks to Silvio Sorrentino for some translations in Italian.

Special thanks to J. Kercheval, Michael M. Dodd.

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

Overview

'TIME TO WIN' is a DLL (**D**ata **L**ink **L**ibrary) only for use with Visual Basic® 3.0 for Windows®.

I'm a Visual Basic® Developper's specialized in Time Attendance, Access Control and Job Control. In this specialization, you must manipulate data on date, hour, bit and string; you must support multi-language and you must make the better and faster program. For all this reasons, I've writed this DLL (fully in C/C++) because I've not founded some functions or subroutines in the Visual Basic® or in other third party.

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

'TIME TO WIN' contains more over 390 functions or subroutines. You can find functions or routines over the following sections :

- Array routines
- Big Numbers
- Bit String Manipulation routines
- Date, Hour and Time routines
- Days and Months in different language
- Disk Array routines
- DOS, Disk and Files routines
- IEEE Conversion routines
- IsX Family Test routines
- Miscellaneous routines
- Multi-Language support
- String Manipulation routines
- Timer functions
- Type functions
- VB Control Specific routines
- Windows Specific routines

Registering 'TIME TO WIN'

The easiest way to Register 'TIME TO WIN' is through CompuServe's SWREG forum.

1) GO SWREG
2) Choose Register Shareware.
3) 'TIME TO WIN' SWREG ID is : #**4045**.

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

The price for 'TIME TO WIN' is fixed at $61.00

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++ 1.51.
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.

If the version 4.0 of VB will be in 32 Bits, I will make 'TIME TO WIN' also in 32 Bits.

<u>Others products :</u>

In the future, I will place on CompuServe (MSBASIC forum), two new products :

1) Adding/Removing error handling to your application (by reading all files included in a .MAK file).

2) Adding multi-language support to your application.by creating external language files (by reading all .FRM included in a .MAK file).

These products will be use 'TIME TO WIN' data link library.

SwapD

Purpose :

SwapD swaps two Double values.

Declare Syntax :

```
Declare Sub cSwapD Lib "time2win.dll" (swap1 As Double, swap2 As Double)
```

Call Syntax :

```
Call cSwapD(swap1, swap2)
```

Where :

swap1	first Double value
swap2	second Double value

Comments :

Examples :

```
swap1 = 2345.12
swap2 = 5432.21
Call cSwapD(swap1, swap2
        -> swap1 = 5432.21
        -> swap2 = 2345.12
```

See Also : cSwapD, cSwapI, cSwapL, cSwapS, cSwapStr

SwapL

Purpose :

SwapL swaps two Long values.

Declare Syntax :

Declare Sub cSwapL Lib "time2win.dll" (swap1 As Long, swap2 As Long)

Call Syntax :

Call cSwapL(swap1, swap2)

Where :

swap1	first Long value
swap2	second Long value

Comments :

Examples :

```
swap1 = 234512
swap2 = 543221
Call cSwapL(swap1, swap2
        -> swap1 = 543221
        -> swap2 = 234512
```

See Also : cSwapD, cSwapI, cSwapL, cSwapS, cSwapStr

SwapI

Purpose :

SwapI swaps two Integer values.

Declare Syntax :

Declare Sub cSwapI Lib "time2win.dll" (swap1 As Integer, swap2 As Integer)

Call Syntax :

Call cSwapI(swap1, swap2)

Where :

swap1	first Integer value
swap2	second Integer value

Comments :

Examples :

```
swap1 = 2345
swap2 = 5432
Call cSwapI(swap1, swap2
        -> swap1 = 5432
        -> swap2 = 2345
```

See Also : cSwapD, cSwapI, cSwapL, cSwapS, cSwapStr

SwapS

Purpose :

SwapS swaps two Single values.

Declare Syntax :

Declare Sub cSwapS Lib "time2win.dll" (swap1 As Single, swap2 As Single)

Call Syntax :

Call cSwapS(swap1, swap2)

Where :

swap1	first Single value
swap2	second Single value

Comments :

Examples :

```
swap1 = 2345.1
swap2 = 5432.2
Call cSwapS(swap1, swap2
          -> swap1 = 5432.2
          -> swap2 = 2345.1
```

See Also : cSwapD, cSwapI, cSwapL, cSwapS, cSwapStr

SwapStr

Purpose :

SwapStr swaps two Strings.

Declare Syntax :

Declare Sub cSwapStr Lib "time2win.dll" (swap1 As String, swap2 As String)

Call Syntax :

Call cSwapStr(swap1, swap2)

Where :

swap1	first String
swap2	second String

Comments :

Examples :

```
swap1 = "Hello"
swap2 = "World"
Call cSwapStr(swap1, swap2
        -> swap1 = "World"
        -> swap2 = "Hello"
```

See Also : cSwapD, cSwapI, cSwapL, cSwapS, cSwapStr

FileSearchAndReplace

Purpose :

FileSearchAndReplace searchs and replaces 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 : cFileSearch, cFileSearchCount

FileSet

Purpose :

FileSetAllAttrib, FileSetArchive, FileSetHidden, FileSetReadOnly, FileSetSystem, FileSetFlag sets respectively all attributes, archive attribute, hidden attribute, read-only attribute, system attribute, specified attribute for the gived file.
FileSetAttrib sets in a Call, all 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 A_NORMAL, A_RDONLY, A_HIDDEN, A_SYSTEM, A_ARCH
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 : FileReset, Constants and Types declaration

FileSearch, FileSearchCount

Purpose :

FileSearch searchs a string in a gived TEXT file.
FileSearchCount counts.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& =
test2& =

See also : cFileSearchAndReplace

PatternExtMatch

Purpose :

PatternExtMatch searches 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 : cPatternMatch, Constants and Types declaration

KillDirFilesAll

Purpose :

KillDirFilesAll deletes 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 : cKillFile, cKillFiles, cKillDir, cKillDirs

BaseConversion

Purpose :

BaseConversion converts 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 :

FileStatistics

Purpose :

FileStatictics counts 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 :

Disk Array routines

The functions/subs usen 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 :

cDAClear	clear a big sized array (fill it with chr$(0)).
cDAClose	close a big sized array and keep it or close a big sized array and destroy it.
cDACreate	create a new big sized array on disk or use an existing big sized array on disk.
cDAGet	read an element from a big sized array on disk.
cDAGetType	read a type'd variable from a big sized array on disk.
cDAPut	save an element to a big sized array on disk.
cDAPutType	save a type'd variable to a big sized array on disk.

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

When you create a new array on disk, a header (128 chars) 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.

DACreate

Purpose :

DACreate creates 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, see Constants and Types declaration. (DA_x)

Comments :

In theory :

The maxium number of Rows is 2147483647
The maxium number of Cols is 2147483647
The maxium 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 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.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.

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

Examples :

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

```
DA.nFilename = "c:\t2w_tmp\dastring.tmp"                    '
DA.nType = 50                                               'positive value for a string
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 routines, cDAClose

DAClose

Purpose :

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 cDACreate

See also : Disk Array routines, cDACreate

DAGet

Purpose :

DAGet reads an element from a big sized array on disk.

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

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, 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 cDACreate

See also : Disk Array routines, cDAPut

DAPut

Purpose :

DAPut saves an element to a big sized array on disk.

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)

Call Syntax :

Call cDAPut(DISKARRAY, Row&, Col&, Sheet&, 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, 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 cDACreate

See also : Disk Array routines, cDAGet

DAPutType

Purpose :

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

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)

Call Syntax :

Call cDAPut(DISKARRAY, Row&, Col&, Sheet&, 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, 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 also : Disk Array routines, cDAGetType

DAGetType

Purpose :

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

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)

Call Syntax :

Call cDAGet(DISKARRAY, Row&, Col&, Sheet&, 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, 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 also : Disk Array routines, cDAPutType

DAClear

Purpose :

DAClear clears a big sized array (fill it whith chr$(0)).

Declare Syntax :

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

Call Syntax :

ErrCode% = cDAClear(DISKARRAY)

Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
ErrCode%	is the returned error code, see Constants and Types declaration. (DA_x)

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.

Examples :

```
Dim ErrCode                As Integer
Dim DA                     As tagDISKARRAY

DA.nFilename = "c:\t2w_tmp\dastring.tmp"                    '
DA.nType = 50                                               'positive value for a string
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 the big sized array on disk
```

See also : Disk Array routines, cDACreate