T I M E   T O   W I N (16-Bit) (version 7.07)

for VISUAL BASIC 4.0 (16-Bit Edition)

Contents

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

@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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (Txt As String, Delimitor As String) As String
Declare Function cFilterChars Lib "t2win-16.dll" (Txt As String, charSet As String) As String
Declare Function cFilterFirstChars Lib "t2win-16.dll" (Txt As String, charSet As String) As String
Declare Function cFilterNotChars Lib "t2win-16.dll" (Txt As String, charSet As String) As String
```

Call Syntax :

```
test = cFilterBlocks(Txt, Delimitor)
test = cFilterChars(Txt, charSet)
test = cFilterFirstChars(Txt, charSet)
test = cFilterNotChars(Txt, charSet)
```

Where :

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

Comments :

Examples :

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

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

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

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

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

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

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

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

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 "t2win-16.dll" (Obj As Object, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
Declare Function cReadCtlLanguage Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer, ByVal FileLanguage As String) As Integer

Call Syntax :

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

Where :

Obj	is any object on the form to use the text language.
Property	is an association of constants (RS_CAPTION, RS_TEXT, RS_DATAFIELD, RS_DATASOURCE, RS_TAG)
FileLanguage	is the file name to perform the language management.
test%	TRUE if all is ok
	FALSE is an error has occured

Comments :

These functions are very, VERY simple to use and your application can support multi-language very fast.

If a problem occurs when accessing the controls or if the filename is an EMPTY string, the returned value is FALSE.
These fonctions doesn't test the validity of the file name.

Ctl can be any control on the form (also Label1).

Property can be RS_CAPTION to use only controls did have a .Caption property.
can be RS_TEXT to use only controls did have a .Text property.
can be RS_DATAFIELD to use only controls did have a .DataField property.
can be RS_DATASOURCE to use only controls did have a .DataSource property.
can be RS_TAG to use only controls did have a .Tag property.
can be any 'OR' association of the four following constants :
RS_CAPTION Or RS_TEXT Or RS_DATAFIELD Or RS_DATASOURCE Or RS_TAG

If ypu want to use all properties, you can pass the value 255.

If you use of RS_DATAFIELD and/or RS_DATASOURCE, you don't need to set the .DataField and/or .DataSource in the Properties Window is design mode. This is can be useful and is not memory hungry, and the EXE size of your application is minder.

FileLanguage is the name of the file to use to store or retrieve the Property. After the first saving, you translate the file (with NOTEPAD, b.e.) into an another language and save it to an other name. You can use the extension als follows .T?? with ?? is FR (for FRench), UK (for United Kingdom, GE (for GErmany), IT (for ITaly), SP (for SPain), ...

Examples :

test% = cSaveCtlLanguage(Command1, RS_CAPTION Or RS_TEXT, "D:\TIME2WIN\DEMO\TIME2WIN.TUK")
translate it to French and save it in the file "D:\TIME2WIN\DEMO\TIME2WIN.TFR"
test% = cReadCtlLanguage(Command1, RS_CAPTION Or RS_TEXT, "D:\TIME2WIN\DEMO\TIME2WIN.TFR")

See also : 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 "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal nTab As Integer) As Long
Declare Function cFileExpandTab Lib "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastDateAccess Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastDateModified Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileTimeCreated Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastTimeAccess Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastTimeModified Lib "t2win-16.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 "t2win-16.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

HB2X

Purpose :

B2I converts a binary string into an integer variable.
B2L converts a binary string into a long variable.
H2I converts a hexa string into an integer variable.
H2L converts a hexa string into a long variable.

Declare Syntax :

```
Declare Function cB2I Lib "t2win-16.dll" (ByVal Txt As String) As Integer
Declare Function cB2L Lib "t2win-16.dll" (ByVal Txt As String) As Long
Declare Function cH2I Lib "t2win-16.dll" (ByVal Txt As String) As Integer
Declare Function cH2L Lib "t2win-16.dll" (ByVal Txt As String) As Long
```

Call Syntax :

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

Where :

txtBinary$	is a binary string (only combinaison of 0, 1)
txtHexa$	is a hexa string (only combinaison of A-Z, a-z, 0-9)

Comments :

If the function detects that that a char is not valid, the conversion is stopped and the returned value is truncated.

Examples :

```
Debug.Print cB2I("1")                                   ' -> 1
Debug.Print cB2I("11")                                  ' -> 3
Debug.Print cB2I("11111111")                            ' -> 255
Debug.Print cB2I("1111111111111111")                    ' -> -1
Debug.Print cB2I("0101010101010101")                    ' -> 21845
Debug.Print cB2I("1010101010101010")                    ' -> -21846

Debug.Print cB2L("1")                                   ' -> 1
Debug.Print cB2L("1111111111111111")                    ' -> 65535
Debug.Print cB2L("11111111111111111111111111111111")    ' -> -1
Debug.Print cB2L("01010101010101010101010101010101")    ' -> 1431655765
Debug.Print cB2L("10101010101010101010101010101010")    ' -> -1431655766

Debug.Print cH2I("0")                                   ' -> 0
Debug.Print cH2I("A1")                                  ' -> 161
Debug.Print cH2I("A1B")                                 ' -> 2587
Debug.Print cH2I("7FFF")                                ' -> 32767
Debug.Print cH2I("A1B2")                                ' -> -24142
Debug.Print cH2I("FFFF")                                ' -> -1

Debug.Print cH2L("0")                                   ' -> 0
Debug.Print cH2L("A1")                                  ' -> 161
Debug.Print cH2L("A1B")                                 ' -> 2587
Debug.Print cH2L("A1B2")                                ' -> 41394
Debug.Print cH2L("7FFFFFFF")                            ' -> 2147483647
Debug.Print cH2L("B2A1A1B2")                            ' -> -1298030158
Debug.Print cH2L("FFFFFFFF")                            ' -> -1
```

See also :

Compress

Purpose :

Compress removes all chr$(0):ASCII NULL, chr$(9):TAB, chr$(32):SPACE from a string

Declare Syntax :

Declare Function cCompress Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal Value As Integer) As String
Declare Function cMKC Lib "t2win-16.dll" (ByVal Value As Currency) As String
Declare Function cMKD Lib "t2win-16.dll" (ByVal Value As Double) As String
Declare Function cMKI Lib "t2win-16.dll" (ByVal Value As Integer) As String
Declare Function cMKL Lib "t2win-16.dll" (ByVal Value As Long) As String
Declare Function cMKS Lib "t2win-16.dll" (ByVal Value As Single) As String

Declare Function cMKN Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 **4** (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_4
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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 **4** (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_4
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 "t2win-16.dll" (ByVal lpszExe As String, ByVal lpszParams As String)
As Integer
Declare Function cRebootSystem Lib "t2win-16.dll" () As Integer
Declare Function cRestartWindows Lib "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal lpFilename As String, ByVal mode As Integer) As Long

Call Syntax :

test = cFileCRC32(lpFilename, mode)

Where :

lpFilename	the file to proceed
mode	OPEN_MODE_BINARY (calculates the CRC on the full length of the file). This is the default mode. OPEN_MODE_TEXT (calculates the CRC until a EOF is encountered)
test	the calculated CRC 32 bits in a LONG.

Comments :

The returned value can be negative and have only a value :

-1 If the filename is not a good filename or if the filename not exist or if an error occurs when accessing the filename.

Examples :

test = cFileCRC32("C:\COMMAND.COM") &h1131ADD3 (MS-DOS 6.22)

See also : cStringCRC32, Constants and Types declaration

FileDrive

Purpose :

FileDrive extracts the drive on which the file is present.

Declare Syntax :

Declare Function cFileDrive Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (Value As String) As Integer
Declare Function cCVC Lib "t2win-16.dll" (Value As String) As Currency
Declare Function cCVD Lib "t2win-16.dll" (Value As String) As Double
Declare Function cCVI Lib "t2win-16.dll" (Value As String) As Integer
Declare Function cCVL Lib "t2win-16.dll" (Value As String) As Long
Declare Function cCVS Lib "t2win-16.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 "t2win-16.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDiskSpace Lib "t2win-16.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDiskUsed Lib "t2win-16.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDiskClusterSize Lib "t2win-16.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 "t2win-16.dll" (ByVal nFilename As String, ByVal firstnext As Integer) As String

Call Syntax :

test$ = cFilesInDirectory(nFilename, firstnext)

Where :

nFilename	the directory to proceed with the file mask (*.* for all)
firstnext	TRUE for the first file
	FALSE for each next file
test$	the returned file

Comments :

Examples :

```
Dim i         As Integer
Dim Tmp       As String

i = 0
Tmp = cFilesInDirectory("c:\*.*", True)

Debug.Print "The first 7 files in C:\ are : "

Do While (Len(Tmp) > 0)
    Debug.Print Tmp
    Tmp = cFilesInDirectory("c:\*.*", False)
    i = i + 1
    If (i >= 7) Then Exit Do
Loop
```

On my system:

```
The first 7 files in C:\ are :

863DATA
WINA20.386
AUTOEXEC.BAT
COMMAND.COM
IMAGE.DAT
BOOTSECT.DOS
ACD.IDX
```

See also : cFilesInDirOnDisk, cFilesInfoInDir, cAllSubDirectories, cSubDirectory

FileSize

Purpose :

FileSize returns the size of the specified file.

Declare Syntax :

Declare Function cFileSize Lib "t2win-16.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 "t2win-16.dll" (ByVal lpFilename As String) As Long
Declare Function cFilesSizeOnDisk Lib "t2win-16.dll" (ByVal nFileName As String) As Long
Declare Function cFilesSlack Lib "t2win-16.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.
IsFileEmpty checks if the file contains or not data (size > 0).
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 "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileEmpty Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileHidden Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFilenameValid Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileNormal Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileReadOnly Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileSubDir Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileSystem Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileVolId Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileFlag Lib "t2win-16.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer

Declare Function cFileGetAttrib Lib "t2win-16.dll" (ByVal nFilename As String, nFileAttribute As Any) As Integer
```

Call Syntax :

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

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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (Type1 As Any, Type2 As Any, ByVal lenType1 As Integer) As Integer
Declare Function cCompareTypeString Lib "t2win-16.dll" Alias "cTypesCompare" (TypeSrc As Any, ByVal Dst As String, ByVal lenTypeSrc As Integer) As Integer
Declare Function cCompareStringType Lib "t2win-16.dll" Alias "cTypesCompare" (ByVal Src As String, TypeDst As Any, ByVal lenTypeSrc As Integer) As Integer

Declare Sub cTypeClear Lib "t2win-16.dll" (TypeSrc As Any, ByVal lenTypeSrc As Integer)
Declare Function cTypeMid Lib "t2win-16.dll" (TypeSrc As Any, ByVal Offset As Integer, ByVal Length As Integer) As String

Declare Sub cTypesCopy Lib "t2win-16.dll" (TypeSrc As Any, TypeDst As Any, ByVal lenTypeSrc As Integer)
Declare Function cTypeTransfert Lib "t2win-16.dll" (TypeSrc As Any, ByVal lenTypeSrc As Integer) As String

Declare Sub cStringToType Lib "t2win-16.dll" Alias "cTypesCopy" (ByVal Src As String, TypeDst As Any, ByVal lenTypeSrc As Integer)
Declare Sub cTypeToString Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (Text As String) As String
Declare Function cFromBinary2 Lib "t2win-16.dll" (Text As String, Bin As String) As String

Declare Function cToBinary Lib "t2win-16.dll" (Text As String) As String
Declare Function cToBinary2 Lib "t2win-16.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 "t2win-16.dll" (Text As String) As String
Declare Function cToHexa Lib "t2win-16.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, GetInPart, GetInPartR, GetInR, TokenIn

Purpose :

Get extratcs 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 extracts a left sub-string delimited by a separator in a gived string.
GetInPart extracts the first left sub-string or the rest after the first sub-string delimited by a separator in a gived string.
GetInPartR extracts the first right sub-string or the rest after the first sub-string delimited by a separator in a gived string.
GetInR extracts a right sub-string delimited by a separator in a gived string.
TokenIn extracts a sub-string delimited by a separator's list in a gived string.

Declare Syntax :

Declare Function cGet Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cGetBlock Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer, ByVal Length As Integer) As String
Declare Function cGetIn Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetInPart Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetInPartR Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetInR Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cTokenIn Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String

Call Syntax :

test$ = cGet(Txt, Position)
test$ = cGetBlock(Txt, Position, Length)
test$ = cGetIn(Txt, Separator, Position)
test$ = cGetInPart(Txt, Separator, Position)
test$ = cGetInPartR(Txt, Separator, Position)
test$ = cGetInR(Txt, Separator, Position)
test$ = cTokenIn(Txt, SeparatorList, Position)

Where :

Txt	the string to proceed.
Position	the position of the sub-string or the block.
Length	the length of each block.
Separator	the delimitor for each sub-string.
SeparatorList	the separator's list for each sub-string.
test$	the result.

Comments :

•If the size of the string is 0 or if the position is < 1 or greater than the maximum block is the string or if the length is 0. The returned string is an empty string.
•The function cGet is a subset of the cGetIn function.
•The function cGetBlock is similar to MID$(Txt, 1+ ((n-1) * m), m)
•The function cTokenIn is a superset of the cGetIn function, in the fact that you can pass a separator's list.
•For the function cGetInPart, cGetInPartR, you must set Position to TRUE for first part (left or right) and to FALSE for second part (left or right).
•The function cGetInPartR is very usefull when you must isolate a file extension or the full directory and the filename function.

Examples :

```
test$ = cGet("A|BC|DEF|G", 1)                              -> "A"
```

```
test$ = cGet("A|BC|DEF|G", 3)                              -> "DEF"

test$ = cGetIn("A/BC/DEF/G", "/", 4)                       -> "G"
test$ = cGetIn("A/BC/DEF/G","D", 2)                                  -> "EF/G"

test$ = cGetInR("A/BC/DEF/G", "/", 4)                      -> "A"
test$ = cGetInR("A/BC/DEF/G","D", 2)                       -> "A/BC/"

test$ = cGetInPart("A/BC/DEF/G", "/", True)                -> "A"
test$ = cGetInPart("A/BC/DEF/G","/", False)                -> "BC/DEF/G"

test$ = cGetInPartR("c:\vberr.hnd\test.mak", ".", True)    -> "mak"
test$ = cGetInPartR("c:\vberr.hnd\test.mak", ".", False)   -> "c:\vberr.hnd\test"

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

test$ = cTokenIn("A/BC:DEF\G", "/:\", 4)                   -> "G"
test$ = cTokenIn("A/BC:DEF\G", "/:\", 3)                   -> "DEF"
```

See also : 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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" Alias "cLngMsgBox" (ByVal nLanguage As Integer, ByVal Message As String, ByVal Button As Long, ByVal Title As String)
Declare Function cLngMsgBox Lib "t2win-16.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 "t2win-16.dll" (Obj As Object, ByVal Text As String)
Declare Sub cSetCtlDataField Lib "t2win-16.dll" (Obj As Object, ByVal Text As String)
Declare Sub cSetCtlFocus Lib "t2win-16.dll" (Obj As Object)
Declare Sub cSetCtlPropString Lib "t2win-16.dll" (Obj As Object, ByVal PropIndex As Integer, ByVal Text As String)
Declare Sub cSetCtlTag Lib "t2win-16.dll" (Obj As Object, ByVal Text As String)
Declare Sub cSetCtlText Lib "t2win-16.dll" (Obj As Object, ByVal Text As String)
```

Call Syntax :

The purpose and the declare syntax are very explicite.

Where :

Obj	the name of the object to proceed

Comments :

The advantage to use these routines is that these routines doesn't generates an error if the property not exists.

Examples :

See also : cSetX, cGetX, cGetCtlX

Morse

Purpose :

Morse converts a string to a morse string.

Declare Syntax :

Declare Function cMorse Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetDataField Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetFocus Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cSetTag Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetText Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetArchive Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetHidden Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetReadOnly Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetSystem Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetFlag Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cResetCapture Lib "t2win-16.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 "t2win-16.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 (16-Bit)', you should install the file T2WIN-16.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 '**T2WIN-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (Var As Variant) As String
Declare Function cIntoBalanceFill Lib "t2win-16.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 "t2win-16.dll" (ByVal nDate As Long) As String
Declare Function cIntoDateFill Lib "t2win-16.dll" (ByVal nDate As Long) As String
Declare Function cIntoDateNull Lib "t2win-16.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

AndToken, AndTokenIn, OrToken, OrTokenIn

Purpose :

AndToken checks if all items of a list of token separated by '|' is present in a specified string.
AndTokenIn checks if all items of a list of token separated by a separator is present in a specified string.

OrToken checks if one item of a list of token separated by '|' is present in a specified string.
OrTokenIn checks if one item of a list of token separated by a separator is present in a specified string.

Declare Syntax :

Declare Function cAndToken Lib "t2win-16.dll" (ByVal Txt As String, ByVal Token As String) As Integer
Declare Function cAndTokenIn Lib "t2win-16.dll" (ByVal Txt As String, ByVal Token As String, ByVal Separator As String) As Integer

Declare Function cOrToken Lib "t2win-16.dll" (ByVal Txt As String, ByVal Token As String) As Integer
Declare Function cOrTokenIn Lib "t2win-16.dll" (ByVal Txt As String, ByVal Token As String, ByVal Separator As String) As Integer

Call Syntax :

Test% = cAndToken(Txt$, Token$)
Test% = cAndTokenIn(Txt$, Token$, Separator$)

Test% = cOrToken(Txt$, Token$)
Test% = cOrTokenIn(Txt$, Token$, Separator$)

Where :

Txt$	is the specified string.
Token$	is the list of token.
Separator$	is the specified separator (default is '|').
Test%	TRUE if one of the list of token is present, FALSE if not

Comments :

AndToken, AndTokenIn, OrToken, OrTokenIn works only with string without embedded chr$(0).
AndToken, AndTokenIn, OrToken, OrTokenIn are case-sensitive. Use UCase$ or LCase$ to perform no case-sensitivity.

Examples :

```
Dim Txt                 As String
Dim Token               As String
Dim Separator           As String
Dim Test         As Integer

Txt = "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"

Token = "THE|DOG|QUICK"
Test = cOrToken(Txt, Token)                                    -> True

Token = "the|dog|quick"
Test = cOrToken(Txt, Token)                                    -> False

Token = "the\dog\quick"
Separator = "\"
Test = cOrTokenIn(lcase$(Txt), lcase$(Token), Separator)       -> True
```

```
Token = "THE|DOG|QUICK"
Test = cAndToken(Txt, Token)                                          -> True

Token = "the|dog|quick"
Test = cAndToken(Txt, Token)                                          -> False

Token = "the\dog\quick"
Separator = "\"
Test = cAndTokenIn(lcase$(Txt), lcase$(Token), Separator)             -> True
```

See also :

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 "t2win-16.dll" (Var As Variant, ByVal Length As Integer, ByVal fillZero As Integer, ByVal Hundreds As Integer) As String
Declare Function cIntoHour Lib "t2win-16.dll" (Var As Variant) As String
Declare Function cIntoVarHour Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Long
Declare Function cFileToUpper Lib "t2win-16.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 "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsAlpha Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsAscii Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsCsym Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsCsymf Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsDigit Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsISBN Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsLower Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsPalindrome Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsPunct Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsSpace Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsUpper Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsXDigit Lib "t2win-16.dll" (Txt As String) As Integer

Declare Function cIsBalance Lib "t2win-16.dll" (ByVal nHour As Long, ByVal nMinute As Integer, ByVal nSecond As
Integer) As Integer
Declare Function cIsDate Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer)
As Integer
Declare Function cIsHour Lib "t2win-16.dll" (ByVal nHour As Integer, ByVal nMinute As Integer, ByVal nSecond As
Integer) As Integer
Declare Function cIsLeapYear Lib "t2win-16.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 "t2win-16.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 "t2win-16.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigDiv Lib "t2win-16.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigMul Lib "t2win-16.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigSub Lib "t2win-16.dll" (Num1 As String, Num2 As String) As String

Declare Function cBigFmt Lib "t2win-16.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 '1.2193263113702107l4e+029'

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

See also : cMKN

Big Numbers

GetClassName

Purpose :

GetClassName retrieves the full class name of a control.

Declare Syntax :

Declare Function cGetClassName Lib "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal lpDir As String) As Integer
Declare Function cKillDirs Lib "t2win-16.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 "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFileAll Lib "t2win-16.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 "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFilesAll Lib "t2win-16.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 "t2win-16.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, MakeMultipleDir

Purpose :

MakeDir creates the specified directory.
MakeMultipleDir creates a multiple directory in one call.

Declare Syntax :

Declare Function cMakeDir Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cMakeMultipleDir Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer

Call Syntax :

test% = cMakeDir(lpFilename)
test% = cMakeMultipleDir(lpFilename)

Where :

lpFilename	the path for the new directory
test%	TRUE if all is OK
	<> TRUE if an error has occured

Comments :

The MakeDir function creates a new directory with the specified dirname. Only one directory can be created at a time, so only the last
component of dirname can name a new directory.
The MakeDir function does not do any translation of path delimiters. All operating systems accept either " or "/ " internally as valid delimiters within paths.
This fonction is the same that MkDir but doesn't generate an VB Error if a problem occurs.

The MakeMultipleDir function creates a new multiple directory with the specified dirname. MakeMultipleDir doesn't return an error if a sub-directory in the multiple directory is already present. The only final test is the existence of the full multiple directory when it was been created.

Examples :

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

test% = cMakeMultipleDir("C:\~~TEST~~\TEST\TMP")   -> TRUE (no error, the directory has been created)

See also : cChDir, cKillDir

Max

Purpose :

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

Declare Syntax :

Declare Function cMax Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 :

ResizeString resizes the size of a string to a new length.

Declare Syntax :

```
Declare Function cResizeString Lib "t2win-16.dll" (Txt As String, ByVal newLength As Integer) As String
```

Call Syntax :

```
Test$ = cResizeString(Txt$, Length%)
```

Where :

Txt$	is the specified string.
Length%	is the new length (can be shorter than the current length).
Test$	is the new string.

Comments :

The new length can be greater than the current length. In this case, chr$(0) is used to fill the rest of the string.

Examples :

```
Test$ = cResizeString("TIME TO WIN", 7)
        -> "TIME TO"
```

See also : cResizeStringAndFill

ResizeStringAndFill

Purpose :

ResizeStringAndFill the size of a string to a new length and fill it with chars if the new length is greater than the current length.

Declare Syntax :

Declare Function cResizeStringAndFill Lib "t2win-16.dll" (Txt As String, ByVal newLength As Integer, Fill As String) As String

Call Syntax :

Test$ = cResizeStringAndFill(Txt$, Length%, Fill$)

Where :

Txt$	is the specified string.
Length%	is the new length (can be shorter than the current length).
Fill$	is a char or a string to use to fill the new string.
Test$	is the new string.

Comments :

The new length can be greater than the current length. In this case, the fill string is used to fill the rest of the string.

Examples :

```
Test$ = cResizeStringAndFill("TIME TO WIN", 21, "@")
        -> "TIME TO WIN@@@@@@@@@@"

Test$ = cResizeStringAndFill("TIME TO WIN", 21, "time")
        -> "TIME TO WINtimetimeti"
```

See also : cResizeString

RestartWindows

Purpose :

Declare Syntax :

Call Syntax :

Where :

Comments :

Reverse

Purpose :

Reverse reverses all chars in a gived string.

Declare Syntax :

```
Declare Function cReverse Lib "t2win-16.dll" (Txt As String) As String
```

Call Syntax :

```
Test$ = cReverse(Txt$)
```

Where :

Txt$	is the specified string
Test$	is the string reversed

Comments :

Examples :

```
Test$ = cReverse("TIME TO WIN")
        -> "NIW OT EMIT"
```

See also :

ReverseSortD

Purpose :

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

Declare Syntax :

Declare Function cReverseSortD Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (Txt As String, Insert As String) As String
Declare Function cInsertBlocksBy Lib "t2win-16.dll" (Txt As String, Insert As String, Delimitor As String) As String
Declare Function cInsertByMask Lib "t2win-16.dll" (Txt As String, Mask As String, Insert As String) As String
Declare Function cInsertChars Lib "t2win-16.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 "t2win-16.dll" (Txt as string) As Integer
Declare Function cCplDigit Lib "t2win-16.dll" (Txt as string) As String
Declare Function cNumDigit Lib "t2win-16.dll" (Txt as string) As Integer
Declare Function cCplAlpha Lib "t2win-16.dll" (Txt As String) As String
```

Call Syntax :

```
test% = cAddDigit(Txt)
test$ = cCplDigit(Txt)
test% = cNumDigit(Txt)
test$ = cCplAlpha(Txt)
```

Where :

Txt$	the string to proceed
test%	the result
test$	the result for CplAlpha

Comments :

For AddDigit, CplDigit, NumDigit if one or more chars are different from digit, the value for each one is 0

Examples :

```
test% = cAddDigit("12345678909876543217123456789098765432l7") -> 194
test% = cNumDigit("1234567890987654321712345678909876543217")-> 5

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

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

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

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 "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlClass Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlContainer Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlDataField Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlForm Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlIndex Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetCtlName Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlNameIndex Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlPropCaption Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetCtlPropDataField Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetCtlPropText Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetCtlTag Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlTagSized Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlText Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetHwnd Lib "t2win-16.dll" (Obj As Object) As Integer
```

Call Syntax :

The purpose and the declare syntax are very explicite.

Where :

Ctl	the name of the control to proceed

Comments :

•The advantage to use these routines is that these routines doesn't generates an error if the property not exists.

Examples :

See also : cGetX, cSetX, cSetCtlX

TrueBetween

Purpose :

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

Declare Syntax :

Declare Function cTrueBetween Lib "t2win-16.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 "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetClass Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetContainer Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetDataField Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetForm Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetIndex Lib "t2win-16.dll" (ByVal hWnd As Integer) As Integer
Declare Function cGetNameIndex Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetText Lib "t2win-16.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 "t2win-16.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

ArrayToComboBox, ArrayToListBox

Purpose :

ArrayToComboBox read an string array and append it to a Combo Box.
ArrayToListBox read an string array and append it to a List Box.

Declare Syntax :

```
Declare Function cArrayToComboBox Lib "t2win-16.dll" (ByVal hWnd As Integer, Array() As Any) As Integer
Declare Function cArrayToListBox Lib "t2win-16.dll" (ByVal hWnd As Integer, Array() As Any) As Integer
```

Call Syntax :

```
Test% = cArrayToComboBox(Combo1.hWnd, Array())
Test% = cArrayToListBox(List1.hWnd, Array())
```

Where :

Combo1.hWnd	the .hWnd of a Combo Box.
List1.hWnd	the .hWnd of a List Box.
nFile$	the filename to read.
Test%	= True, if all is ok, <> True, if an error has occured.

Comments :

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

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

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

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

Examples :

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

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

Debug.Print cArrayToListBox(List1.hWnd, AD())
Debug.Print cArrayToComboBox(Combo1.hWnd, AD())
```

See also :

Uncompact

Purpose :

Uncompact uncompacts a string composed of numeric chars.

Declare Syntax :

Declare Function cUncompact Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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

ArrayStringOnDisk

Purpose :

Put/Get full variable string array (one dimension) on/from disk ascii file.

Declare Syntax :

Declare Function cArrayStringOnDisk Lib "t2win-16.dll" (ByVal File As String, Array() As Any, ByVal GetPut As Integer, rRecords As Long) As Long

Call Syntax :

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

Where :

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

Comments :

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

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

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

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

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

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

Examples :

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

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

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

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

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

```
Next i

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

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

See also : Disk Array routines, cArrayOnDisk

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 "t2win-16.dll" (Obj As Object)
Declare Sub cDisableFI Lib "t2win-16.dll" (Obj As Object)

Call Syntax :

Call cEnableFI(Ctl)
Call cDisableFI(Ctl)

Where :

Ctl	the invisible control that you want become visible (cDisableFI) or invisible (cEnableFI).

Comments :

I use this function with a picture control which containes a timer BMP.

If the enabled state of the form is changing, a WM_ENABLE message is sent before this function returns. If a form is already disabled, all its child forms are implicitly disabled, although they are not sent a WM_ENABLE message.

After some experience, I've noted that some custom controls doesn't answers correctly to this function. In fact, all controls can't receive the input when you Call cDisableFI.

Use this with caution.

See also : cEnableForm, cDisableForm

EnableForm, DisableForm

Purpose :

EnableForm and DisableForm 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 "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cDisableForm Lib "t2win-16.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 "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cDisableRedraw Lib "t2win-16.dll" (ByVal hWnd As Integer)

Declare Sub cEnableCtlRedraw Lib "t2win-16.dll" (Obj As Object)
Declare Sub cDisableCtlRedraw Lib "t2win-16.dll" (Obj As Object)
```

Call Syntax :

```
Call cEnableRedraw(Ctl.hWnd)
Call cDisableRedraw(Ctl.hWnd)

Call cEnableCtlRedraw(Ctl)
Call cDisableCtlRedraw(Ctl)
```

Where :

Comments :

The WM_SETREDRAW message can be used to set and clear the redraw flag for a window. This message is very useful for
preventing a list box from being updated when many items are being added to it, and then allowing the list box to be redrawn when all
of the changes have been made to its contents. Using this technique prevents a list box that is currently visible from flashing
constantly as its contents are being updated.

This message sets or clears the redraw flag. If the redraw flag is cleared, the contents of the specified window will not be updated
after each change, and the window will not be repainted until the redraw flag is set. For example, an application that needs to add
several items to a list box can clear the redraw flag, add the items, and then set the redraw flag. Finally, the application can Call the
InvalidateRect function to cause the list box to be repainted.

If the custom control doesn't have a .hWnd (Label control b.e.), you must use the XCtlRedraw routine.

Fill

Purpose :

Fill fills a string with some chars.

Declare Syntax :

Declare Sub cFill Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, Filter As String) As Long
Declare Function cFileFilterNot Lib "t2win-16.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 "t2win-16.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
7.07	The following functions has been removed : cReadMnuLanguage has been included in the functions cReadCtlLanguage, cReadCtlLanguageExt cSaveMnuLanguage has been included in the functions cSaveCtlLanguage, cSaveCtlLanguageExt
7.00	Initial release of the 'TIME TO WIN (16-Bit)' Dynamic Link Library for Visual Basic 4.0 (16-Bit Edition).

New Features

See also : Revision History

Version	Comments
7.07	Conversion of a binary string into an integer variable. cB2I Conversion of a binary string into a long variable. cB2L Conversion of a hexa string into an integer variable. cH2I Conversion of a hexa string into a long variable. cH2L Access of method (by position) of OCX custom controls. cObjectMethodByPos Access of method (by name) of OCX custom controls. cObjectMethodByName Reads data in properties (by position) from OCX custom controls. cObjectGetPropertyByPos Reads data in properties (by name) from OCX custom controls. cObjectGetPropertyByName Writes data in properties (by position) in OCX custom controls. cObjectPutPropertyByPos Writes data in properties (by name) from OCX custom controls. cObjectPutPropertyByName
7.00	Initial release of the 'TIME TO WIN (16-Bit)' Dynamic Link Library for Visual Basic 4.0 (16-Bit Edition).

FileCopy

Purpose :

FileCopy copies one file to an another file.

Declare Syntax :

Declare Function cFileCopy Lib "t2win-16.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 "t2win-16.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 T2WIN-16.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 "t2win-16.dll" () As String
Declare Function cGetCountryCode Lib "t2win-16.dll" () As String
Declare Function cGetCurrency Lib "t2win-16.dll" () As String
Declare Function cGetDateFormat Lib "t2win-16.dll" () As String
Declare Function cGetDateSeparator Lib "t2win-16.dll" () As String
Declare Function cGetHourFormat Lib "t2win-16.dll" () As String
Declare Function cGetLanguage Lib "t2win-16.dll" () As String
Declare Function cGetListSeparator Lib "t2win-16.dll" () As String
Declare Function cGetTimeSeparator Lib "t2win-16.dll" () As String
Declare Function cGetWinINI Lib "t2win-16.dll" (ByVal Info As Integer) As String
```

Call Syntax :

The purpose and the declare syntax are very explicite.

Where :

Info	the number of the following 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 T2WIN-16.DLL and T2WIN-16.HLP should be copied in your   WINDOWS\SYSTEM directory.

Registered version :

The files T2WIN-16.DLL, T2WIN-16.HLP should be copied in your WINDOWS\SYSTEM directory.
The file T2WIN-16.LIC should be copied in your WINDOWS directory.

Distribution note:

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

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

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 "t2win-16.dll" (ByVal nTimer As Integer, ByVal nValue As Long)
Declare Sub cStartWait Lib "t2win-16.dll" (ByVal nTimer As Integer)
Declare Function cCheckWait Lib "t2win-16.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 "t2win-16.dll" ()
Declare Function cReadBasisTimer Lib "t2win-16.dll" () As Long
Declare Sub cStopBasisTimer Lib "t2win-16.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 "t2win-16.dll" (ByVal nTimer As Integer)
Declare Function cReadTimer Lib "t2win-16.dll" (ByVal nTimer As Integer) As Long
Declare Function cStopTimer Lib "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, Password As String, ByVal Level As Integer) As Long
Declare Function cFileDecrypt Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, Password As String, ByVal Level As Integer) As Long

Call Syntax :

test& = cFileEncrypt(file1, file2, password, level)
test& = cFileDecrypt(file1, file2, password, level)

Where :

file1$	is the source file.
file2$	is the destination file.
password	is the key to use for encryption/decryption.
level	level of the encryption/decryption.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occured.

Comments :

The password/key is case sensitive.
The level is a number between **0** and **4** (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 password is an EMPTY string.
-32720	the number of chars in a block for writing differs from the number of chars for reading.
-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.
-32760	allocation error for memory buffer 1.
-32761	allocation error for memory buffer 2.

Examples :

test& = cFileEncrypt("c:\autoexec.bat", "c:\autoexec.tb1", "Time To Win", ENCRYPT_LEVEL_4)
test& = cFileDecrypt("c:\autoexec.tb1", "c:\autoexec.tb2", "Time To Win", ENCRYPT_LEVEL_4)

See also :

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 "t2win-16.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 "t2win-16.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
cReadMnuLanguage
cSaveCtlLanguage
cSaveMnuLanguage

UnloadDLL

Purpose :

UnloadDLL unloads a DLL from the memory.

Declare Syntax :

```
Declare Sub cUnloadDLL Lib "t2win-16.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 "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Integer
Declare Function cCmpFileContents Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal sensitivity As Integer) As Integer
Declare Function cCmpFileSize Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Integer
Declare Function cCmpFileTime Lib "t2win-16.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 Sub c3D Lib "t2win-16.dll" (Obj As Object, ByVal Method As Integer, ByVal Thickness As Integer)
Declare Sub c3DMeter Lib "t2win-16.dll" (hObj As Object, Meter As tag3DMeter)
Declare Function cAddD Lib "t2win-16.dll" (array() As Double, ByVal nValue As Double) As Integer
Declare Function cAddDigit Lib "t2win-16.dll" (Txt as string) As integer
Declare Function cAddI Lib "t2win-16.dll" (array() As Integer, ByVal nValue As Integer) As Integer
Declare Function cAddL Lib "t2win-16.dll" (array() As Long, ByVal nValue As Long) As Integer
Declare Function cAddS Lib "t2win-16.dll" (array() As Single, ByVal nValue As Single) As Integer
Declare Function cAddTime Lib "t2win-16.dll" (ByVal Hr As Integer) As Integer
Declare Function cAddTwoTimes Lib "t2win-16.dll" (ByVal Time1 As String, ByVal Time2 As String) As String
Declare Function cAlign Lib "t2win-16.dll" (Txt As String, ByVal TypeAlign As Integer, ByVal NewLength As Integer) As String
Declare Function cAllSubDirectories Lib "t2win-16.dll" (ByVal lpBaseDirectory As String, nDir As Integer) As String
Declare Function cAndToken Lib "t2win-16.dll" (ByVal Txt As String, ByVal Token As String) As Integer
Declare Function cAndTokenIn Lib "t2win-16.dll" (ByVal Txt As String, ByVal Token As String, ByVal Separator As String) As Integer
Declare Function cArabicToRoman Lib "t2win-16.dll" (Var As Variant) As String
Declare Sub cArrangeDesktopIcons Lib "t2win-16.dll" ()
Declare Function cArrayOnDisk Lib "t2win-16.dll" (ByVal File As String, Array() As Any, ByVal GetPut As Integer) As Long
Declare Function cArrayPrm Lib "t2win-16.dll" (array() As Any, nArray As Any) As Integer
Declare Function cArrayStringOnDisk Lib "t2win-16.dll" (ByVal File As String, Array() As Any, ByVal GetPut As Integer, rRecords As Long) As Long
Declare Function cArrayToComboBox Lib "t2win-16.dll" (ByVal hWnd As Integer, Array() As Any) As Integer
Declare Function cArrayToListBox Lib "t2win-16.dll" (ByVal hWnd As Integer, Array() As Any) As Integer
Declare Function cB2I Lib "t2win-16.dll" (ByVal Txt As String) As Integer
Declare Function cB2L Lib "t2win-16.dll" (ByVal Txt As String) As Long
Declare Function cBaseConversion Lib "t2win-16.dll" (ByVal Num As String, ByVal RadixIn As Integer, ByVal RadixOut As Integer) As String
Declare Function cBetween Lib "t2win-16.dll" (Var As Variant, Var1 As Variant, Var2 As Variant) As Integer
Declare Function cBigAdd Lib "t2win-16.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigDiv Lib "t2win-16.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigMul Lib "t2win-16.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigNum Lib "t2win-16.dll" (ByVal n1 As String, ByVal op As Integer, ByVal n2 As String) As String
Declare Function cBigSub Lib "t2win-16.dll" (Num1 As String, Num2 As String) As String
Declare Function cBigFmt Lib "t2win-16.dll" (Num As String, ByVal Fmt As Integer) As String
Declare Function cBlockCharFromLeft Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cBlockCharFromRight Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As String
Declare Sub cCenterWindow Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cChangeChars Lib "t2win-16.dll" (Txt As String, charSet As String, newCharSet As String)
Declare Sub cChangeCharsUntil Lib "t2win-16.dll" (Txt As String, charSet As String, newCharSet As String, nUntil As String)
Declare Sub cChangeTaskName Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Function cChDir Lib "t2win-16.dll" (ByVal lpDir As String) As Integer
Declare Function cChDrive Lib "t2win-16.dll" (ByVal lpDrive As String) As Integer
Declare Function cCheckChars Lib "t2win-16.dll" (Txt As String, charSet As String) As Integer
Declare Function cCheckNumericity Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cCheckTime Lib "t2win-16.dll" (ByVal Hr As Integer, ByVal Hr1 As Integer, ByVal Hr2 As Integer) As Integer
Declare Function cCheckWait Lib "t2win-16.dll" (ByVal nTimer As Integer) As Integer
Declare Function cCloseAllEditForm Lib "t2win-16.dll" () As Integer
Declare Function cCmpFileAttribute Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Integer
Declare Function cCmpFileContents Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal sensitivity As Integer) As Integer
Declare Function cCmpFileSize Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Integer
Declare Function cCmpFileTime Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Integer
Declare Sub cCnvASCIItoEBCDIC Lib "t2win-16.dll" (Txt As String)
Declare Sub cCnvEBCDICtoASCII Lib "t2win-16.dll" (Txt As String)
Declare Function cCombination Lib "t2win-16.dll" (ByVal nItems As Integer, ByVal mTimes As Integer) As Double
Declare Function cCompact Lib "t2win-16.dll" (Txt As String) As String

```
Declare Function cCompareTypeString Lib "t2win-16.dll" Alias "cTypesCompare" (TypeSrc As Any, ByVal Dst As String, ByVal lenTypeSrc As Integer) As Integer
Declare Function cCompareStringType Lib "t2win-16.dll" Alias "cTypesCompare" (ByVal Src As String, TypeDst As Any, ByVal lenTypeSrc As Integer) As Integer
Declare Function cCompress Lib "t2win-16.dll" (Txt As String) As String
Declare Function cCompressTab Lib "t2win-16.dll" (Txt As String, ByVal nTab As Integer) As String
Declare Function cCount Lib "t2win-16.dll" (Txt As String, Separator As String) As Integer
Declare Function cCountDirectories Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cCountFiles Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cCountI Lib "t2win-16.dll" (array() As Integer, ByVal Value As Integer) As Long
Declare Function cCountL Lib "t2win-16.dll" (array() As Long, ByVal Value As Long) As Long
Declare Function cCountS Lib "t2win-16.dll" (array() As Single, ByVal Value As Single) As Long
Declare Function cCountD Lib "t2win-16.dll" (array() As Double, ByVal Value As Double) As Long
Declare Function cCplAlpha Lib "t2win-16.dll" (Txt As String) As String
Declare Function cCplDigit Lib "t2win-16.dll" (Txt As String) As String
Declare Function cCreateAndFill Lib "t2win-16.dll" (ByVal Length As Integer, Txt As String) As String
Declare Function cCreateBits Lib "t2win-16.dll" (ByVal nBits As Integer) As String
Declare Sub cCtl3D Lib "t2win-16.dll" (Obj As Object, ByVal LeftTopColor As Long, ByVal RightBottomColor As Long, ByVal Thickness As Integer)
Declare Function cCurrentTime Lib "t2win-16.dll" () As Integer
Declare Function cCVB Lib "t2win-16.dll" (Value As String) As Integer
Declare Function cCVC Lib "t2win-16.dll" (Value As String) As Currency
Declare Function cCVD Lib "t2win-16.dll" (Value As String) As Double
Declare Function cCVI Lib "t2win-16.dll" (Value As String) As Integer
Declare Function cCVL Lib "t2win-16.dll" (Value As String) As Long
Declare Function cCVS Lib "t2win-16.dll" (Value As String) As Single
Declare Function cDAClear Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY) As Integer
Declare Function cDAClearCol Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, ByVal Sheet As Long) As Integer
Declare Function cDAClearRow Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Sheet As Long) As Integer
Declare Function cDAClearSheet Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Sheet As Long) As Integer
Declare Sub cDAClose Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal DeleteFile As Integer)
Declare Function cDACreate Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal CreateOrUse As Integer) As Integer
Declare Function cDAGet Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long) As Variant
Declare Sub cDAGetType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)
Declare Sub cDAPut Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, Var As Variant)
Declare Sub cDAPutType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)
Declare Sub cDArGet Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, Var As Variant)
Declare Sub cDArGetType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, nType As Any)
Declare Sub cDArPut Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, Var As Variant)
Declare Sub cDArPutType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, nType As Any)
Declare Function cDAsClearCol Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long) As Integer
Declare Function cDAsClearRow Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long) As Integer
Declare Sub cDAsGet Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, Var As Variant)
Declare Sub cDAsGetType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)
Declare Sub cDAsPut Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, Var As Variant)
Declare Sub cDAsPutType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)
Declare Function cDateToScalar Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Long
Declare Function cDayOfWeek Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer, ByVal nISO As Integer) As Integer
Declare Function cDayOfYear Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Integer
Declare Function cDaysInMonth Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer) As Integer
Declare Sub cDecrI Lib "t2win-16.dll" (Value As Integer)
Declare Sub cDecrL Lib "t2win-16.dll" (Value As Long)
Declare Function cDecrypt Lib "t2win-16.dll" (Txt As String, password As String, ByVal level As Integer) As String
Declare Function cDeviationD Lib "t2win-16.dll" (array() As Double) As Double
Declare Function cDeviationI Lib "t2win-16.dll" (array() As Integer) As Double
Declare Function cDeviationL Lib "t2win-16.dll" (array() As Long) As Double
Declare Function cDeviationS Lib "t2win-16.dll" (array() As Single) As Double
Declare Sub cDisableCtlRedraw Lib "t2win-16.dll" (Obj As Object)
Declare Sub cDisableFI Lib "t2win-16.dll" (Obj As Object)
Declare Sub cDisableForm Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cDisableRedraw Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Function cDOSGetMediaID Lib "t2win-16.dll" (ByVal nDrive As String, MEDIAID As tagMEDIAID) As Integer
Declare Function cDOSGetVolumeLabel Lib "t2win-16.dll" (ByVal nDrive As String) As String
Declare Function cDOSSetMediaID Lib "t2win-16.dll" (ByVal nDrive As String, MEDIAID As tagMEDIAID) As Integer
Declare Function cDOSSetVolumeLabel Lib "t2win-16.dll" (ByVal nDrive As String, ByVal nVolumeLabel As String) As Integer
Declare Sub cEnableCtlRedraw Lib "t2win-16.dll" (Obj As Object)
Declare Sub cEnableFI Lib "t2win-16.dll" (Obj As Object)
Declare Sub cEnableForm Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cEnableRedraw Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Function cEncrypt Lib "t2win-16.dll" (Txt As String, password As String, ByVal level As Integer) As String
Declare Function cEXEnameActiveWindow Lib "t2win-16.dll" () As String
Declare Function cEXEnameTask Lib "t2win-16.dll" (ByVal nFileName As String) As String
Declare Function cEXEnameWindow Lib "t2win-16.dll" (ByVal hModule As Integer) As String
Declare Function cExitWindowsAndExecute Lib "t2win-16.dll" (ByVal lpszExe As String, ByVal lpszParams As String) As Integer
Declare Function cExpandTab Lib "t2win-16.dll" (Txt As String, ByVal nTab As Integer) As String
Declare Function cFileChangeChars Lib "t2win-16.dll" (ByVal nFilename As String, CharSet As String, NewCharSet As String, ByVal nFileTemp As String) As Long
Declare Function cFileCompress Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Long
Declare Function cFileCompressTab Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal nTab As Integer) As Long
Declare Function cFileCopy Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Long
Declare Function cFileCRC32 Lib "t2win-16.dll" (ByVal lpFilename As String, ByVal mode As Integer) As Long
Declare Function cFileDateCreated Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileDecrypt Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal password As String, ByVal level As Integer) As Long
Declare Function cFileDrive Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileEncrypt Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal password As String, ByVal level As Integer) As Long
Declare Function cFileExpand Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Long
Declare Function cFileExpandTab Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal nTab As Integer) As Long
Declare Function cFileFilter Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal Filter As String) As Long
Declare Function cFileFilterNot Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal Filter As String) As Long
Declare Function cFileGetAttrib Lib "t2win-16.dll" (ByVal nFilename As String, nFileAttribute As Any) As Integer
Declare Function cFileLastDateAccess Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastDateModified Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastTimeAccess Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileLastTimeModified Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileLineCount Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cFileMerge Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String, ByVal fileTo As String) As Long
Declare Function cFilePathExists Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cFileResetAllAttrib Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetArchive Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetFlag Lib "t2win-16.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer
Declare Function cFileResetHidden Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetReadOnly Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileResetSystem Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSearch Lib "t2win-16.dll" (ByVal nFileName As String, ByVal Search As String, ByVal sensitivity As Integer) As Long
Declare Function cFileSearchAndReplace Lib "t2win-16.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 "t2win-16.dll" (ByVal nFileName As String, ByVal Search As String, ByVal sensitivity As Integer) As Long
Declare Function cFileSetAllAttrib Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetArchive Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetAttrib Lib "t2win-16.dll" (ByVal nFilename As String, nFileAttribute As Any) As Integer
Declare Function cFileSetFlag Lib "t2win-16.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer
Declare Function cFileSetHidden Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetReadOnly Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetSystem Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFilesInDirectory Lib "t2win-16.dll" (ByVal nFilename As String, ByVal firstnext As Integer) As String
Declare Function cFilesInDirOnDisk Lib "t2win-16.dll" (ByVal nFile As String, ByVal nFilename As String, ByVal nAttribute As Integer) As Integer
Declare Function cFilesInDirToArray Lib "t2win-16.dll" (ByVal nFilename As String, ByVal nAttribute As Integer, array() As Any) As Integer
Declare Function cFilesInfoInDir Lib "t2win-16.dll" (ByVal nFilename As String, FILEINFO As tagFILEINFO, ByVal FirstNext As Integer) As String
Declare Function cFileSize Lib "t2win-16.dll" (ByVal lpFilename As String) As Long
Declare Function cFileSort Lib "t2win-16.dll" (ByVal FileIn As String, ByVal FileOut As String, ByVal SortMethod As Integer) As Long
Declare Function cFilesSize Lib "t2win-16.dll" (ByVal nFilename As String) As Long
Declare Function cFilesSizeOnDisk Lib "t2win-16.dll" (ByVal nDrive As String, ByVal nFileName As String) As Long
Declare Function cFilesSlack Lib "t2win-16.dll" (ByVal nDrive As String, ByVal nFileName As String, Size1 As Long, Size2 As Long) As Integer
Declare Function cFileStatistics Lib "t2win-16.dll" (ByVal nFilename As String, nLines As Long, nWords As Long, nChars As Long) As Long
Declare Function cFileTimeCreated Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cFileToComboBox Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal nFile As String) As Integer
Declare Function cFileToListBox Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal nFile As String) As Integer
Declare Function cFileToLower Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Long
Declare Function cFileToUpper Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Long
Declare Sub cFill Lib "t2win-16.dll" (Txt As String, Fill As String)
Declare Function cFillD Lib "t2win-16.dll" (array() As Double, ByVal nValue As Double) As Integer
Declare Function cFillI Lib "t2win-16.dll" (array() As Integer, ByVal nValue As Integer) As Integer
Declare Function cFillL Lib "t2win-16.dll" (array() As Long, ByVal nValue As Long) As Integer
Declare Function cFillS Lib "t2win-16.dll" (array() As Single, ByVal nValue As Single) As Integer
Declare Function cFillIncrD Lib "t2win-16.dll" (Array() As Double, ByVal nValue As Double, ByVal Increment As Double) As Integer
Declare Function cFillIncrI Lib "t2win-16.dll" (Array() As Integer, ByVal nValue As Integer, ByVal Increment As Integer) As Integer
Declare Function cFillIncrL Lib "t2win-16.dll" (Array() As Long, ByVal nValue As Long, ByVal Increment As Long) As Integer
Declare Function cFillIncrS Lib "t2win-16.dll" (Array() As Single, ByVal nValue As Single, ByVal Increment As Single) As Integer
Declare Function cFilterBlocks Lib "t2win-16.dll" (Txt As String, Delimitor As String) As String
Declare Function cFilterChars Lib "t2win-16.dll" (Txt As String, charSet As String) As String
Declare Function cFilterFirstChars Lib "t2win-16.dll" (Txt As String, charSet As String) As String
Declare Function cFilterNotChars Lib "t2win-16.dll" (Txt As String, charSet As String) As String
Declare Function cFindBitReset Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As Integer
Declare Function cFindBitSet Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As Integer
Declare Function cFindFileInEnv Lib "t2win-16.dll" (ByVal lpFilename As String, ByVal lpEnv As String) As Integer
Declare Function cFindFileInPath Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cFloppyInfo Lib "t2win-16.dll" (ByVal nDrive As String, nHeads As Integer, nCylinders As Integer, nSectors As Integer) As Integer
Declare Function cFraction Lib "t2win-16.dll" (ByVal nValue As Double, nNumerator As Double, nDenominator As Double) As Double
```

Declare Function cFromBinary Lib "t2win-16.dll" (Text As String) As String
Declare Function cFromBinary2 Lib "t2win-16.dll" (Text As String, Bin As String) As String
Declare Function cFromHexa Lib "t2win-16.dll" (Text As String) As String
Declare Function cFullPath Lib "t2win-16.dll" (ByVal nFilename As String) As String
Declare Function cFXpicture Lib "t2win-16.dll" (ByVal method As Integer, ByVal hdc1 As Integer, ByVal hbitmap As Integer, ByVal parameter As Integer, ByVal delay As Integer) As Integer
Declare Function cGet Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cGetAscTime Lib "t2win-16.dll" (ByVal nLanguage As Integer) As String
Declare Function cGetBit Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As Integer
Declare Function cGetBlock Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer, ByVal Length As Integer) As String
Declare Function cGetCaption Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetChangeTaskName Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String) As String
Declare Function cGetClass Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetClassName Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetContainer Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetCountry Lib "t2win-16.dll" () As String
Declare Function cGetCountryCode Lib "t2win-16.dll" () As String
Declare Function cGetCtlCaption Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlClass Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlContainer Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlDataField Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlForm Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlIndex Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetCtlName Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlNameIndex Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlPropCaption Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetCtlPropDataField Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetCtlPropText Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetCtlTag Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlTagSized Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCtlText Lib "t2win-16.dll" (Obj As Object) As String
Declare Function cGetCurrency Lib "t2win-16.dll" () As String
Declare Function cGetCurrentDrive Lib "t2win-16.dll" () As String
Declare Function cGetDataField Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetDateFormat Lib "t2win-16.dll" () As String
Declare Function cGetDateSeparator Lib "t2win-16.dll" () As String
Declare Function cGetDefaultCurrentDir Lib "t2win-16.dll" () As String
Declare Function cGetDefaultPrinter Lib "t2win-16.dll" () As String
Declare Function cGetDevices Lib "t2win-16.dll" () As String
Declare Function cGetDiskClusterSize Lib "t2win-16.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDiskFree Lib "t2win-16.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDiskSpace Lib "t2win-16.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDiskUsed Lib "t2win-16.dll" (ByVal lpDrive As String) As Long
Declare Function cGetDriveCurrentDir Lib "t2win-16.dll" (ByVal lpDrive As String) As String
Declare Function cGetDriveType Lib "t2win-16.dll" (ByVal lpDrive As String) As Integer
Declare Function cGetFileVersion Lib "t2win-16.dll" (ByVal filename As String, ByVal nFonction As Integer) As String
Declare Function cGetFileVersionInfo Lib "t2win-16.dll" (ByVal filename As String, FILEVERSIONINFO As Any) As Integer
Declare Function cGetForm Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetFullNameInEnv Lib "t2win-16.dll" (ByVal lpFilename As String, ByVal lpEnv As String) As String
Declare Function cGetFullNameInPath Lib "t2win-16.dll" (ByVal lpFilename As String) As String
Declare Function cGetHourFormat Lib "t2win-16.dll" () As String
Declare Function cGetHwnd Lib "t2win-16.dll" (Obj As Object) As Integer
Declare Function cGetIn Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetIndex Lib "t2win-16.dll" (ByVal hWnd As Integer) As Integer
Declare Function cGetIni Lib "t2win-16.dll" (ByVal AppName As String, ByVal szItem As String, ByVal szDefault As String, ByVal InitFile As String) As String
Declare Function cGetInPart Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetInPartR Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String

```
Declare Function cGetInR Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cGetLanguage Lib "t2win-16.dll" () As String
Declare Function cGetListSeparator Lib "t2win-16.dll" () As String
Declare Function cGetLongDay Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetLongMonth Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As String
Declare Function cGetName Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetNameIndex Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetNetConnection Lib "t2win-16.dll" (ByVal lpDrive As String, ErrCode As Integer) As String
Declare Function cGetPid Lib "t2win-16.dll" () As Integer
Declare Function cGetPrinterPorts Lib "t2win-16.dll" () As String
Declare Function cGetSectionItems Lib "t2win-16.dll" (ByVal Section As String, ByVal InitFile As String, nItems As Integer) As String
Declare Function cGetSmallDay Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetShortDay Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetShortMonth Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As String
Declare Function cGetSystemDirectory Lib "t2win-16.dll" () As String
Declare Function cGetTaskName Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetText Lib "t2win-16.dll" (ByVal hWnd As Integer) As String
Declare Function cGetTimeSeparator Lib "t2win-16.dll" () As String
Declare Function cGetTinyDay Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetTinyMonth Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As String
Declare Function cGetVersion Lib "t2win-16.dll" () As Single
Declare Function cGetWindowsDirectory Lib "t2win-16.dll" () As String
Declare Function cGetWinINI Lib "t2win-16.dll" (ByVal Info As Integer) As String
Declare Function cGetWinSection Lib "t2win-16.dll" (ByVal Section As String) As String
Declare Function cGiveBitPalindrome Lib "t2win-16.dll" () As String
Declare Function cH2I Lib "t2win-16.dll" (ByVal Txt As String) As Integer
Declare Function cH2L Lib "t2win-16.dll" (ByVal Txt As String) As Long
Declare Function cHashMD5 Lib "t2win-16.dll" (Text As String) As String
Declare Function cHideAllEditForm Lib "t2win-16.dll" () As Integer
Declare Function cHideDebugForm Lib "t2win-16.dll" () As Integer
Declare Function cHMAClear Lib "t2win-16.dll" (HMA As tagHMA) As Integer
Declare Function cHMAClearCol Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long, ByVal sheet As Long) As Integer
Declare Function cHMAClearRow Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal sheet As Long) As Integer
Declare Function cHMAClearSheet Lib "t2win-16.dll" (HMA As tagHMA, ByVal sheet As Long) As Integer
Declare Function cHMACreate Lib "t2win-16.dll" (HMA As tagHMA) As Integer
Declare Function cHMAFree Lib "t2win-16.dll" (HMA As tagHMA) As Integer
Declare Function cHMAGet Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal sheet As Long) As Variant
Declare Sub cHMAGetType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal sheet As Long, nType As Any)
Declare Sub cHMAPut Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal sheet As Long, Var As Variant)
Declare Sub cHMAPutType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal sheet As Long, nType As Any)
Declare Sub cHMArGet Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long, Var As Variant)
Declare Sub cHMArGetType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long, nType As Any)
Declare Sub cHMArPut Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long, Var As Variant)
Declare Sub cHMArPutType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long, nType As Any)
Declare Sub cHMAsGet Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, Var As Variant)
Declare Sub cHMAsGetType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, nType As Any)
Declare Sub cHMAsPut Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, Var As Variant)
Declare Sub cHMAsPutType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, nType As Any)
Declare Function cHMAsClearCol Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long) As Integer
Declare Function cHMAsClearRow Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long) As Integer
Declare Function cHMAOnDisk Lib "t2win-16.dll" (HMA As tagHMA, ByVal hsFile As String, ByVal hsGetPut As Integer) As Long
Declare Function cHourTo Lib "t2win-16.dll" (Txt As String) As Variant
Declare Function cHugeStrAdd Lib "t2win-16.dll" (ByVal hsHandle As Integer, hsText As String) As Integer
Declare Function cHugeStrAddress Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
Declare Function cHugeStrAppend Lib "t2win-16.dll" (ByVal hsHandle As Integer, hsText As String) As Integer
Declare Function cHugeStrBlocks Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
Declare Function cHugeStrClear Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Integer
Declare Function cHugeStrCreate Lib "t2win-16.dll" (ByVal hsSize As Long) As Integer
Declare Function cHugeStrFree Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Integer
Declare Function cHugeStrGetNP Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
Declare Function cHugeStrGetWP Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
Declare Function cHugeStrLength Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
Declare Function cHugeStrMid Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsStart As Long, ByVal hsLength As Long) As String
Declare Function cHugeStrNext Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsNext As Long) As String
Declare Function cHugeStrOnDisk Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsFile As String, ByVal hsGetPut As Integer) As Long
Declare Function cHugeStrRead Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsBlock As Long) As String
Declare Function cHugeStrSetNP Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsPtr As Long) As Integer
Declare Function cHugeStrSetWP Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsPtr As Long) As Integer
Declare Function cHugeStrSize Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
Declare Sub cIncrI Lib "t2win-16.dll" (Value As Integer)
Declare Sub cIncrL Lib "t2win-16.dll" (Value As Long)
Declare Function cInsertBlocks Lib "t2win-16.dll" (Txt As String, Insert As String) As String
Declare Function cInsertBlocksBy Lib "t2win-16.dll" (Txt As String, Insert As String, Delimitor As String) As String
Declare Function cInsertByMask Lib "t2win-16.dll" (Txt As String, Mask As String, Insert As String) As String
Declare Function cInsertChars Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer, Insert As String) As String
Declare Function cIntoBalance Lib "t2win-16.dll" (Var As Variant) As String
Declare Function cIntoBalanceFill Lib "t2win-16.dll" (Var As Variant) As String
Declare Function cIntoDate Lib "t2win-16.dll" (ByVal nDate As Long) As String
Declare Function cIntoDateFill Lib "t2win-16.dll" (ByVal nDate As Long) As String
Declare Function cIntoDateNull Lib "t2win-16.dll" (ByVal nDate As Long) As String
Declare Function cIntoFixHour Lib "t2win-16.dll" (Var As Variant, ByVal Length As Integer, ByVal fillZero As Integer, ByVal Hundreds As Integer) As String
Declare Function cIntoHour Lib "t2win-16.dll" (Var As Variant) As String
Declare Function cIntoVarHour Lib "t2win-16.dll" (Var As Variant) As String
Declare Function cIsAlnum Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsAlpha Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsAscii Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsBalance Lib "t2win-16.dll" (ByVal nHour As Long, ByVal nMinute As Integer, ByVal nSecond As Integer) As Integer
Declare Function cIsBitPalindrome Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsCsym Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsCsymf Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsDate Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Integer
Declare Function cIsDigit Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsFileArchive Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileFlag Lib "t2win-16.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer
Declare Function cIsFileHidden Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileNormal Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFilenameValid Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileReadOnly Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileSubDir Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileSystem Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFileVolId Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cIsFormEnabled Lib "t2win-16.dll" (ByVal hWnd As Integer) As Integer
Declare Function cIsHour Lib "t2win-16.dll" (ByVal nHour As Integer, ByVal nMinute As Integer, ByVal nSecond As Integer) As Integer
Declare Function cIsISBN Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsLeapYear Lib "t2win-16.dll" (ByVal nYear As Integer) As Integer
Declare Function cIsLower Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsPalindrome Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsPunct Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsSerial Lib "t2win-16.dll" (ByVal File1 As String) As Integer
Declare Function cIsSpace Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsUpper Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cIsXdigit Lib "t2win-16.dll" (Txt As String) As Integer
Declare Function cKillDir Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillDirFilesAll Lib "t2win-16.dll" (ByVal lpDir As String, ByVal lpMask As String) As Integer
Declare Function cKillDirs Lib "t2win-16.dll" (ByVal lpDir As String, ByVal HeaderDirectory As Integer) As Integer
Declare Function cKillFile Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFileAll Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFiles Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cKillFilesAll Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Sub cKillFocus Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cLngBoxMsg Lib "t2win-16.dll" Alias "cLngMsgBox" (ByVal nLanguage As Integer, ByVal Message As String, ByVal Button As Long, ByVal Title As String)
Declare Function cLngInpBox Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal Message As String, ByVal Title As String, ByVal Default As String) As String
Declare Function cLngMsgBox Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal Message As String, ByVal Button As Long, ByVal Title As String) As Integer
Declare Function cLrc Lib "t2win-16.dll" (Txt As String) As String
Declare Function cMakeDir Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cMakeMultipleDir Lib "t2win-16.dll" (ByVal lpFilename As String) As Integer
Declare Function cMakePath Lib "t2win-16.dll" (ByVal nDrive As String, ByVal nDir As String, ByVal nFilename As String, ByVal Ext As String) As String
Declare Sub cMatrixAdd Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
Declare Function cMatrixCoFactor Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal Row As Integer, ByVal Col As Integer) As Double
Declare Function cMatrixCompare Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
Declare Sub cMatrixCopy Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double)
Declare Function cMatrixDet Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double) As Double
Declare Function cMatrixFill Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal nInit As Integer) As Integer
Declare Function cMatrixInv Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
Declare Function cMatrixMinor Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal Row As Integer, ByVal Col As Integer) As Double
Declare Sub cMatrixMul Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
Declare Sub cMatrixSub Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
Declare Function cMatrixSymToeplitz Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
Declare Sub cMatrixTranspose Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double)
Declare Function cMax Lib "t2win-16.dll" (Var1 As Variant, Var2 As Variant) As Variant
Declare Function cMaxD Lib "t2win-16.dll" (array() As Double) As Double
Declare Function cMaxI Lib "t2win-16.dll" (array() As Integer) As Integer
Declare Function cMaxL Lib "t2win-16.dll" (array() As Long) As Long
Declare Function cMaxS Lib "t2win-16.dll" (array() As Single) As Single
Declare Function cMDAClear Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY) As Integer
Declare Function cMDAClearCol Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long, ByVal sheet As Long) As Integer
Declare Function cMDAClearRow Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal sheet As Long) As Integer
Declare Function cMDAClearSheet Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal sheet As Long) As Integer
Declare Sub cMDAClose Lib "t2win-16.dll" (MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal DeleteFile As Integer)
Declare Function cMDACreate Lib "t2win-16.dll" (MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal CreateOrUse As Integer) As Integer
Declare Function cMDAGet Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal sheet As Long) As Variant
Declare Sub cMDAGetType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal sheet As Long, nType As Any)
Declare Sub cMDAPut Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal sheet As Long, Var As Variant)
Declare Sub cMDAPutType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal sheet As Long, nType As Any)
Declare Sub cMDArGet Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Col As Long, Var As Variant)
Declare Sub cMDArGetType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Col As Long, nType As Any)
Declare Sub cMDArPut Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Col As Long, Var As Variant)
Declare Sub cMDArPutType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Col As Long, nType As Any)
Declare Function cMDAsClearCol Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Col As Long) As Integer
Declare Function cMDAsClearRow Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long) As Integer
Declare Sub cMDAsGet Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, Var As Variant)
Declare Sub cMDAsGetType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)
Declare Sub cMDAsPut Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, Var As Variant)
Declare Sub cMDAsPutType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)
Declare Function cMeanD Lib "t2win-16.dll" (array() As Double) As Double
Declare Function cMeanI Lib "t2win-16.dll" (array() As Integer) As Double
Declare Function cMeanL Lib "t2win-16.dll" (array() As Long) As Double
Declare Function cMeanS Lib "t2win-16.dll" (array() As Single) As Double
Declare Function cMin Lib "t2win-16.dll" (Var1 As Variant, Var2 As Variant) As Variant
Declare Function cMinD Lib "t2win-16.dll" (array() As Double) As Double
Declare Function cMinI Lib "t2win-16.dll" (array() As Integer) As Integer
Declare Function cMinL Lib "t2win-16.dll" (array() As Long) As Long
Declare Function cMinS Lib "t2win-16.dll" (array() As Single) As Single
Declare Function cMixChars Lib "t2win-16.dll" (Txt As String) As String
Declare Function cMKB Lib "t2win-16.dll" (ByVal Value As Integer) As String
Declare Function cMKC Lib "t2win-16.dll" (ByVal Value As Currency) As String
Declare Function cMKD Lib "t2win-16.dll" (ByVal Value As Double) As String
Declare Function cMKI Lib "t2win-16.dll" (ByVal Value As Integer) As String
Declare Function cMKL Lib "t2win-16.dll" (ByVal Value As Long) As String
Declare Function cMKN Lib "t2win-16.dll" (ByVal Value As Double) As String
Declare Function cMKS Lib "t2win-16.dll" (ByVal Value As Single) As String
Declare Function cModuleFind Lib "t2win-16.dll" (MODULEENTRY As Any, ByVal ModuleName As String) As Integer
Declare Function cModules Lib "t2win-16.dll" (MODULEENTRY As Any, ByVal firstnext As Integer) As Integer
Declare Function cMorse Lib "t2win-16.dll" (ByVal morse As String) As String
Declare Function cNextHwnd Lib "t2win-16.dll" (ByVal hWnd As Integer) As Integer
Declare Function cNumDigit Lib "t2win-16.dll" (Txt as string) As integer
Declare Sub cObjectMethodByPos Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer, lpPut As Variant)
Declare Function cObjectGetPropertyByPos Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer) As Variant
Declare Sub cObjectPutPropertyByPos Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer, lpPut As Variant)
Declare Sub cObjectMethodByName Lib "t2win-16.dll" (Obj As Object, ByVal Property As String, lpPut As Variant)
Declare Function cObjectGetPropertyByName Lib "t2win-16.dll" (Obj As Object, ByVal Property As String) As Variant
Declare Sub cObjectPutPropertyByName Lib "t2win-16.dll" (Obj As Object, ByVal Property As String, lpPut As
Variant)
Declare Function cOneCharFromLeft Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cOneCharFromRight Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cOrToken Lib "t2win-16.dll" (ByVal Txt As String, ByVal Token As String) As Integer
Declare Function cOrTokenIn Lib "t2win-16.dll" (ByVal Txt As String, ByVal Token As String, ByVal Separator As
String) As Integer
```

```
Declare Function cPatternExtMatch Lib "t2win-16.dll" (ByVal Txt As String, ByVal Pattern As String) As Integer
Declare Function cPatternMatch Lib "t2win-16.dll" (ByVal Txt As String, ByVal Pattern As String) As Integer
Declare Function cProperName Lib "t2win-16.dll" (Txt As String) As String
Declare Function cProperName2 Lib "t2win-16.dll" (Txt As String, ByVal TokenToUse As String, ByVal Options As Integer) As String
Declare Sub cPutIni Lib "t2win-16.dll" (ByVal AppName As String, ByVal szItem As String, ByVal szDefault As String, ByVal InitFile As String)
Declare Function cRcsCountFileDir Lib "t2win-16.dll" (ByVal FileOrDir As Integer, ByVal FirstFileOrDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Integer
Declare Function cRcsFilesSize Lib "t2win-16.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Long
Declare Function cRcsFilesSizeOnDisk Lib "t2win-16.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Long
Declare Function cRcsFilesSlack Lib "t2win-16.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer, Size1 As Long, Size2 As Long) As Integer
Declare Function cReadBasisTimer Lib "t2win-16.dll" () As Long
Declare Function cReadCtlLanguage Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
Declare Function cReadMnuLanguage Lib "t2win-16.dll" (hCtlFirstMenu As Control, ByVal FileLanguage As String) As Integer
Declare Function cReadTimer Lib "t2win-16.dll" (ByVal nTimer As Integer) As Long
Declare Function cRebootSystem Lib "t2win-16.dll" () As Integer
Declare Function cRegistrationKey Lib "t2win-16.dll" (ByVal RegString As String, ByVal RegCode As Long) As Long
Declare Function cRemoveBlockChar Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer, ByVal Length As Integer) As String
Declare Function cRemoveOneChar Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer) As String
Declare Function cRenameFile Lib "t2win-16.dll" (ByVal lpFilename1 As String, ByVal lpFilename2 As String) As Integer
Declare Sub cResetCapture Lib "t2win-16.dll" ()
Declare Sub cResetFocus Lib "t2win-16.dll" (ByVal hWnd1 As Integer, ByVal hWnd2 As Integer)
Declare Function cResizeString Lib "t2win-16.dll" (Txt As String, ByVal newLength As Integer) As String
Declare Function cResizeStringAndFill Lib "t2win-16.dll" (Txt As String, ByVal newLength As Integer, Fill As String) As String
Declare Function cRestartWindows Lib "t2win-16.dll" () As Integer
Declare Function cReverse Lib "t2win-16.dll" (Txt As String) As String
Declare Sub cReverseAllBits Lib "t2win-16.dll" (Txt As String)
Declare Sub cReverseAllBitsByChar Lib "t2win-16.dll" (Txt As String)
Declare Function cReverseSortD Lib "t2win-16.dll" (array() As Double) As Integer
Declare Function cReverseSortI Lib "t2win-16.dll" (array() As Integer) As Integer
Declare Function cReverseSortL Lib "t2win-16.dll" (array() As Long) As Integer
Declare Function cReverseSortS Lib "t2win-16.dll" (array() As Single) As Integer
Declare Function cReverseSortStr Lib "t2win-16.dll" (Txt As String, ByVal nItem As Integer, ByVal ItemLength As Integer) As Integer
Declare Sub cRndInit Lib "t2win-16.dll" (ByVal nRnd As Long)
Declare Function cRnd Lib "t2win-16.dll" () As Double
Declare Function cRndD Lib "t2win-16.dll" () As Double
Declare Function cRndI Lib "t2win-16.dll" () As Integer
Declare Function cRndL Lib "t2win-16.dll" () As Long
Declare Function cRndS Lib "t2win-16.dll" () As Single
Declare Function cRomanToArabic Lib "t2win-16.dll" (Txt As String) As Variant
Declare Function cSaveCtlLanguage Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer, ByVal FileLanguage As String) As Integer
Declare Function cSaveMnuLanguage Lib "t2win-16.dll" (hCtlFirstMenu As Control, ByVal FileLanguage As String) As Integer
Declare Sub cScalarToDate Lib "t2win-16.dll" (ByVal Scalar As Long, nYear As Integer, nMonth As Integer, nDay As Integer)
Declare Sub cScalarToTime Lib "t2win-16.dll" (ByVal Scalar As Long, nHour As Integer, nMin As Integer, nSec As Integer)
Declare Function cScrollL Lib "t2win-16.dll" (Txt As String) As String
Declare Function cScrollR Lib "t2win-16.dll" (Txt As String) As String
Declare Function cSearchI Lib "t2win-16.dll" (array() As Integer, ByVal Value As Integer) As Long
Declare Function cSearchL Lib "t2win-16.dll" (array() As Long, ByVal Value As Long) As Long
Declare Function cSearchS Lib "t2win-16.dll" (array() As Single, ByVal Value As Single) As Long
Declare Function cSearchD Lib "t2win-16.dll" (array() As Double, ByVal Value As Double) As Long
Declare Function cSerialGet Lib "t2win-16.dll" (ByVal file As String, SERIALDATA As tagSERIALDATA) As Integer
Declare Function cSerialInc Lib "t2win-16.dll" (ByVal file As String, ByVal Increment As Long) As Integer
Declare Function cSerialPut Lib "t2win-16.dll" (ByVal file As String, SERIALDATA As tagSERIALDATA) As Integer
Declare Function cSerialRmv Lib "t2win-16.dll" (ByVal File As String) As Integer
Declare Sub cSetAllBits Lib "t2win-16.dll" (Txt As String, ByVal Value As Integer)
Declare Sub cSetBit Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer, ByVal Value As Integer)
Declare Sub cSetBitToFalse Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer)
Declare Sub cSetBitToTrue Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer)
Declare Sub cSetCaption Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetCapture Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Sub cSetCtlCaption Lib "t2win-16.dll" (Obj As Object, ByVal Text As String)
Declare Sub cSetCtlDataField Lib "t2win-16.dll" (Obj As Object, ByVal Text As String)
Declare Sub cSetCtlFocus Lib "t2win-16.dll" (Obj As Object)
Declare Sub cSetCtlPropString Lib "t2win-16.dll" (Obj As Object, ByVal PropIndex As Integer, ByVal Text As String)
Declare Sub cSetCtlTag Lib "t2win-16.dll" (Obj As Object, ByVal Text As String)
Declare Sub cSetCtlText Lib "t2win-16.dll" (Obj As Object, ByVal Text As String)
Declare Function cSetD Lib "t2win-16.dll" (array() As Double, ByVal nValue As Double) As Integer
Declare Sub cSetDataField Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetDefaultSeparator Lib "t2win-16.dll" (Separator As String)
Declare Sub cSetFocus Lib "t2win-16.dll" (ByVal hWnd As Integer)
Declare Function cSetHandleCount Lib "t2win-16.dll" (ByVal nHandle As Integer) As Integer
Declare Function cSetI Lib "t2win-16.dll" (array() As Integer, ByVal nValue As Integer) As Integer
Declare Function cSetL Lib "t2win-16.dll" (array() As Long, ByVal nValue As Long) As Integer
Declare Function cSetS Lib "t2win-16.dll" (array() As Single, ByVal nValue As Single) As Integer
Declare Sub cSetTag Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetText Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Text As String)
Declare Sub cSetWait Lib "t2win-16.dll" (ByVal nTimer As Integer, ByVal nValue As Long)
Declare Sub cShowWindow Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal method As Integer, ByVal interval As
Integer)
Declare Function cSleep Lib "t2win-16.dll" (ByVal Delay As Long) As Integer
Declare Function cSortD Lib "t2win-16.dll" (array() As Double) As Integer
Declare Function cSortI Lib "t2win-16.dll" (array() As Integer) As Integer
Declare Function cSortL Lib "t2win-16.dll" (array() As Long) As Integer
Declare Function cSortS Lib "t2win-16.dll" (array() As Single) As Integer
Declare Function cSortStr Lib "t2win-16.dll" (Txt As String, ByVal nItem As Integer, ByVal ItemLength As Integer) As
Integer
Declare Function cSpellMoney Lib "t2win-16.dll" (ByVal Value As Double, ByVal Units As String, ByVal Cents As
String) As String
Declare Sub cSplitPath Lib "t2win-16.dll" (ByVal nFilename As String, SPLITPATH As Any)
Declare Sub cStartBasisTimer Lib "t2win-16.dll" ()
Declare Sub cStartTimer Lib "t2win-16.dll" (ByVal nTimer As Integer)
Declare Sub cStartWait Lib "t2win-16.dll" (ByVal nTimer As Integer)
Declare Sub cStopBasisTimer Lib "t2win-16.dll" ()
Declare Function cStopTimer Lib "t2win-16.dll" (ByVal nTimer As Integer) As Long
Declare Function cStringCompress Lib "t2win-16.dll" (Txt As String) As String
Declare Function cStringCRC32 Lib "t2win-16.dll" (Txt As String) As Long
Declare Function cStringExpand Lib "t2win-16.dll" (Txt As String) As String
Declare Function cStringSAR Lib "t2win-16.dll" (ByVal Txt As String, ByVal Search As String, ByVal Replace As
String, ByVal Sensitivity As Integer) As String
Declare Sub cStringToType Lib "t2win-16.dll" Alias "cTypesCopy" (ByVal Src As String, TypeDst As Any, ByVal
lenTypeSrc As Integer)
Declare Function cSubDirectory Lib "t2win-16.dll" (ByVal nFilename As String, ByVal firstnext As Integer) As String
Declare Function cSumD Lib "t2win-16.dll" (array() As Double) As Double
Declare Function cSumI Lib "t2win-16.dll" (array() As Integer) As Double
Declare Function cSumL Lib "t2win-16.dll" (array() As Long) As Double
Declare Function cSumS Lib "t2win-16.dll" (array() As Single) As Double
Declare Sub cSwapD Lib "t2win-16.dll" (swap1 As Double, swap2 As Double)
Declare Sub cSwapI Lib "t2win-16.dll" (swap1 As Integer, swap2 As Integer)
Declare Sub cSwapL Lib "t2win-16.dll" (swap1 As Long, swap2 As Long)
Declare Sub cSwapS Lib "t2win-16.dll" (swap1 As Single, swap2 As Single)
Declare Sub cSwapStr Lib "t2win-16.dll" (swap1 As String, swap2 As String)
Declare Sub cSysMenuChange Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal Position As Integer, ByVal
NewMessage As String)
Declare Function cTaskFind Lib "t2win-16.dll" (TASKENTRY As Any, ByVal hTask As Integer) As Integer
Declare Function cTasks Lib "t2win-16.dll" (TASKENTRY As Any, ByVal firstnext As Integer) As Integer
Declare Function cTimeBetween Lib "t2win-16.dll" (ByVal Hr1 As Integer, ByVal Hr2 As Integer) As Integer
Declare Function cTimerClose Lib "t2win-16.dll" (ByVal TimerHandle As Integer) As Integer
Declare Function cTimerOpen Lib "t2win-16.dll" () As Integer
Declare Function cTimerRead Lib "t2win-16.dll" (ByVal TimerHandle As Integer) As Long
Declare Function cTimerStart Lib "t2win-16.dll" (ByVal TimerHandle As Integer) As Integer
Declare Function cTimeToScalar Lib "t2win-16.dll" (ByVal nHour As Integer, ByVal nMin As Integer, ByVal nSec As
Integer) As Long
Declare Function cToBinary Lib "t2win-16.dll" (Text As String) As String
Declare Function cToBinary2 Lib "t2win-16.dll" (Text As String, Bin As String) As String
Declare Sub cToggleAllBits Lib "t2win-16.dll" (Txt As String)
Declare Sub cToggleBit Lib "t2win-16.dll" (Txt As String, ByVal Position As Integer)
Declare Function cToHexa Lib "t2win-16.dll" (Text As String) As String
Declare Function cTokenIn Lib "t2win-16.dll" (Txt As String, Separator As String, ByVal Position As Integer) As String
Declare Function cTrueBetween Lib "t2win-16.dll" (Var As Variant, Var1 As Variant, Var2 As Variant) As Integer
Declare Function cTruncatePath Lib "t2win-16.dll" (ByVal nFilename As String, ByVal NewLength As Integer) As
String
Declare Sub cTypeClear Lib "t2win-16.dll" (TypeSrc As Any, ByVal lenTypeSrc As Integer)
Declare Function cTypeMid Lib "t2win-16.dll" (TypeSrc As Any, ByVal Offset As Integer, ByVal Length As Integer) As
String
Declare Function cTypesCompare Lib "t2win-16.dll" (Type1 As Any, Type2 As Any, ByVal lenType1 As Integer) As
Integer
Declare Sub cTypesCopy Lib "t2win-16.dll" (TypeSrc As Any, TypeDst As Any, ByVal lenTypeSrc As Integer)
Declare Function cTypeTransfert Lib "t2win-16.dll" (TypeSrc As Any, ByVal lenTypeSrc As Integer) As String
Declare Sub cTypeToString Lib "t2win-16.dll" Alias "cTypesCopy" (TypeSrc As Any, ByVal Dst As String, ByVal
lenTypeSrc As Integer)
Declare Function cUncompact Lib "t2win-16.dll" (Txt As String) As String
Declare Function cUniqueFileName Lib "t2win-16.dll" (Txt As String) As String
Declare Function cUnHideAllEditForm Lib "t2win-16.dll" () As Integer
Declare Function cUnHideDebugForm Lib "t2win-16.dll" () As Integer
Declare Sub cUnloadDLL Lib "t2win-16.dll" (ByVal hMod As Integer)
Declare Function cWalkThruWindow Lib "t2win-16.dll" (Class As String, Caption As String, OwnerHwnd As Integer,
OwnerClass As String, OwnerCaption As String, ByVal FirstNext As Integer) As Integer
Declare Function cWeekOfYear Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As
Integer, ByVal nISO As Integer) As Integer

' file input/output using the C routines
Declare Function cFopen Lib "t2win-16.dll" (ByVal File As String, ByVal Mode As String) As Long
Declare Function cFclose Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Function cFgetc Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Function cFputc Lib "t2win-16.dll" (ByVal char As Integer, ByVal IOstream As Long) As Integer
Declare Function cFputs Lib "t2win-16.dll" (ByVal Txt As String, ByVal IOstream As Long) As Integer
Declare Function cFgets Lib "t2win-16.dll" (Txt As String, ByVal Length As Integer, ByVal IOstream As Long) As
Integer
Declare Function cFwrite Lib "t2win-16.dll" (Txt As String, ByVal IOstream As Long) As Integer
Declare Function cFread Lib "t2win-16.dll" (Txt As String, ByVal Length As Integer, ByVal IOstream As Long) As
Integer
Declare Function cFcloseall Lib "t2win-16.dll" () As Integer
Declare Function cFflush Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Function cFflushall Lib "t2win-16.dll" () As Integer
Declare Function cFeof Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Function cFerror Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Sub cFclearerr Lib "t2win-16.dll" (ByVal IOstream As Long)
Declare Function cFseek Lib "t2win-16.dll" (ByVal IOstream As Long, ByVal offset As Long, ByVal Origin As Integer)
As Integer
Declare Function cFtell Lib "t2win-16.dll" (ByVal IOstream As Long) As Long
Declare Sub cFrewind Lib "t2win-16.dll" (ByVal IOstream As Long)
```

```
' functions for calculating 2-D geometry
Declare Sub cV2Add Lib "t2win-16.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
Declare Sub cV2Sub Lib "t2win-16.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
Declare Sub cV2Combine Lib "t2win-16.dll" (u As tagVECTOR2, ByVal c1 As Double, v As tagVECTOR2, ByVal c2
As Double, w As tagVECTOR2)
Declare Sub cV2Copy Lib "t2win-16.dll" (u As tagVECTOR2, w As tagVECTOR2)
Declare Function cV2Dot Lib "t2win-16.dll" (u As tagVECTOR2, v As tagVECTOR2) As Double
Declare Function cV2Length Lib "t2win-16.dll" (u As tagVECTOR2) As Double
Declare Function cV2LengthSquared Lib "t2win-16.dll" (u As tagVECTOR2) As Double
Declare Sub cV2LinearIp Lib "t2win-16.dll" (lo As tagVECTOR2, hi As tagVECTOR2, ByVal alpha As Double, w As
tagVECTOR2)
Declare Sub cV2Mul Lib "t2win-16.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
Declare Sub cV2Neg Lib "t2win-16.dll" (u As tagVECTOR2)
Declare Sub cV2Normalized Lib "t2win-16.dll" (u As tagVECTOR2)
Declare Sub cV2Ortho Lib "t2win-16.dll" (u As tagVECTOR2, w As tagVECTOR2)
Declare Sub cV2ScaledNewLength Lib "t2win-16.dll" (u As tagVECTOR2, ByVal newlen As Double)
Declare Function cV2SegmentLength Lib "t2win-16.dll" (p As tagVECTOR2, q As tagVECTOR2) As Double

' functions for calculating 3-D geometry
Declare Sub cV3Add Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
Declare Sub cV3Sub Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
Declare Sub cV3Combine Lib "t2win-16.dll" (u As tagVECTOR3, ByVal c1 As Double, v As tagVECTOR3, ByVal c2
As Double, w As tagVECTOR3)
Declare Sub cV3Copy Lib "t2win-16.dll" (u As tagVECTOR3, w As tagVECTOR3)
Declare Sub cV3Cross Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
Declare Function cV3Dot Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3) As Double
Declare Function cV3Length Lib "t2win-16.dll" (u As tagVECTOR3) As Double
Declare Function cV3LengthSquared Lib "t2win-16.dll" (u As tagVECTOR3) As Double
Declare Sub cV3LinearIp Lib "t2win-16.dll" (lo As tagVECTOR3, hi As tagVECTOR3, ByVal alpha As Double, w As
tagVECTOR3)
Declare Sub cV3Mul Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
Declare Sub cV3Neg Lib "t2win-16.dll" (u As tagVECTOR3)
Declare Sub cV3Normalized Lib "t2win-16.dll" (u As tagVECTOR3)
Declare Sub cV3ScaledNewLength Lib "t2win-16.dll" (u As tagVECTOR3, ByVal newlen As Double)
Declare Function cV3SegmentLength Lib "t2win-16.dll" (p As tagVECTOR3, q As tagVECTOR3) As Double
```

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 "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetSmallDay Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetShortDay Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetLongDay Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nDay As Integer) As String
Declare Function cGetTinyMonth Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As String
Declare Function cGetShortMonth Lib "t2win-16.dll" (ByVal nLanguage As Integer, ByVal nMonth As Integer) As
String
Declare Function cGetLongMonth Lib "t2win-16.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

Put/Get full array on/from disk

cArrayOnDisk cArrayStringOnDisk

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

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

cFillIncrD cFillIncrI cFillIncrL cFillIncrS

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

Count a specific value in a array

cCountD cCountI cCountL cCountS

Search a specific value in a array

cSearchD cSearchI cSearchL cSearchS

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

cAlllSubDirectories
cChDir
cChDrive
cCmpFileAttribute
cCmpFileContents
cCmpFileSize
cCmpFileTime
cCountDirectories
cCountFiles
cDOSGetMediaID
cDOSGetVolumeLabel
cDOSSetMediaID
cDOSSetVolumeLabel
cFileChangeChars
cFileCompress
cFileCompressTab
cFileCopy
cFileCRC32
cFileDecrypt
cFileEncrypt
cFileExpand
cFileExpandTab
cFileFilter
cFileFilterNot
cFileDateCreated
cFileDrive
cFileGetAttrib
cFileLastDateAccess
cFileLastDateModified
cFileLastTimeAccess
cFileLastTimeModified
cFileLineCount
cFileMerge
cFilePathExists
cFileResetAllAttrib
cFileResetArchive
cFileResetFlag
cFileResetHidden
cFileResetReadOnly
cFileResetSystem
cFileSearch
cFileSearchAndReplace
cFileSearchCount
cFileSetAllAttrib
cFileSetArchive
cFileSetAttrib
cFileSetFlag
cFileSetHidden
cFileSetReadOnly
cFileSetSystem
cFilesInDirectory
cFilesInDirOnDisk
cFilesInDirToArray
cFilesInfoInDir
cFileSize
cFileSort
cFilesSize
cFilesSizeOnDisk
cFilesSlack

cFileStatistics
cFileTimeCreated
cFileToLower
cFileToUpper
cFindFileInEnv
cFindFileInPath
cFloppyInfo
cFullPath
cGetCurrentDrive
cGetDefaultCurrentDir
cGetDiskClusterSize
cGetDiskFree
cGetDiskSpace
cGetDiskUsed
cGetDriveCurrentDir
cGetDriveType
cGetFullNameInEnv
cGetFullNameInPath
cGetNetConnection
cIsFileArchive
cIsFileFlag
cIsFileHidden
cIsFileNormal
cIsFileReadOnly
cIsFileSubDir
cIsFileSystem
cIsFileVolId
cKillDir
cKillDirFilesAll
cKillDirs
cKillFile
cKillFileAll
cKillFiles
cKillFilesAll
cMakeDir
cMakeMultipleDir
cMakePath
cRcsCountFileDir
cRcsFilesSize
cRcsFilesSizeOnDisk
cRcsFilesSlack
cRenameFile
cSplitPath
cSubDirectory
cTruncatePath
cUniqueFileName

IsX Family Test routines

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.

cRomanToArabic
cScrollL
cScrollR
cSpellMoney
cStringCompress
cStringExpand
cStringSAR
cToBinary
cToBinary2
cToHexa
cTokenIn
cUncompact

Timer functions

Timer functions performs timing functions for your application. These functions are divided in two parts :

1) Timing which use the GetTickCount() have an accuracy of **55** ms, these functions are available for all applications in memory and share the same memory space. You can have 32 timers. Be carefully, when distributing the DLL on an other computer did use the same DLL.

cCheckWait
cReadBasisTimer
cReadTimer
cSetWait
cSleep
cStartBasisTimer
cStartTimer
cStartWait
cStopBasisTimer
cStopTimer

2) Timing which use the TimerCountt() have an accuracy of **1** ms, these functions use the concept of handle to permit to have many different application which can use the DLL. You can have 64 handles.

cTimerClose
cTimerOpen
cTimerRead
cTimerStart

Type functions

VB Control Specific routines

c3D
c3DMeter
cCloseAllEditForm
cCtl3D
cDisableCtlRedraw
cDisableFI
cDisableForm
cDisableRedraw
cEnableCtlRedraw
cEnableFI
cEnableForm
cEnableRedraw
cGetCaption
cGetClass
cGetContainer
cGetCtlCaption
cGetCtlClass
cGetCtlContainer
cGetCtlDataField
cGetCtlForm
cGetCtlIndex
cGetCtlName
cGetCtlNameIndex
cGetCtlPropCaption
cGetCtlPropDataField
cGetCtlPropText
cGetCtlRect
cGetCtlRectTwips
cGetCtlTag
cGetCtlTagSized
cGetCtlText
cGetDataField
cGetForm
cGetHwnd
cGetIndex
cGetName
cGetNameIndex
cGetText
cHideAllEditForm
cHideDebugForm
cKillFocus
cObjectMethodByName
cObjectMethodByPos
cObjectGetPropertyByName
cObjectGetPropertyByPos
cObjectPutPropertyByName
cObjectPutPropertyByPos
cResetCapture
cResetFocus
cSetCaption
cSetCapture
cSetCtlCaption
cSetCtlDataField
cSetCtlFocus
cSetCtlPropString
cSetCtlTag
cSetCtlText
cSetDataField
cSetFocus

cSetTag
cSetText
cHideAllEditForm
cUnHideDebugForm

Windows Specific routines

cArrangeDesktopIcons
cArrayToComboBox
cArrayToListBox
cChangeTaskName
cEXEnameActiveWindow
cEXEnameTask
cEXEnameWindow
cExitWindowsAndExecute
cFileToComboBox
cFileToListBox
cFXPicture
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
cShowWindow
cTaskFind
cTasks
cUnloadDLL
cWalkThruWindow

Constants and Types declaration

```
Option Explicit

' definition for win.ini section
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

' definition for drive type
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

' definition for file attributes
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 A_NORMAL_ARCHIVE = &HFE                 'Normal, Archive
Global Const A_ALL = &HFF                            'Normal, Archive, Read-Only, Hidden,
System

' definition for encrypt/decrypt
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 ENCRYPT_LEVEL_4 = 4

' definition for FILECRC32
Global Const OPEN_MODE_BINARY = 0
Global Const OPEN_MODE_TEXT = 1

' definition for ARRAYONDISK
Global Const PUT_ARRAY_ON_DISK = 0
Global Const GET_ARRAY_ON_DISK = 1

' definition for big numbers
Global Const BIG_ADD = 0
Global Const BIG_SUB = 1
Global Const BIG_MUL = 2

' definition for file version information
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

' definition for language in multi-language management
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 LNG_CATALAN = 7
Global Const LNG_POLISH = 8

' definition for message position in multi-language message box
Global Const MB_MESSAGE_LEFT = 0
Global Const MB_MESSAGE_CENTER = 8192
Global Const MB_MESSAGE_RIGHT = 16384

' definition for timeout management in multi-language message box
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

' definition for properties for language management
Global Const RS_CAPTION = 1
Global Const RS_TEXT = 2
Global Const RS_DATAFIELD = 4
Global Const RS_DATASOURCE = 8
Global Const RS_TAG = 16
Global Const RS_MENU = 32
Global Const RS_ALL = 255

' definition for accessing properties in OBJECT controls (OCX or VBX)
Global Const OBJ_CAPTION = 0
Global Const OBJ_CLASS = 1
Global Const OBJ_CONTAINER = 2
Global Const OBJ_DATAFIELD = 3
Global Const OBJ_FORM = 4
Global Const OBJ_INDEX = 5
Global Const OBJ_NAME = 6
Global Const OBJ_NAMEINDEX = 7
Global Const OBJ_TAG = 8
Global Const OBJ_TEXT = 9

' definition for error type for PATTERNMATCHEXT
```

```
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

' definition for error type for ISFILENAMEVALID
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

' definition for variable type in DISK ARRAY
Global Const DA_BYTE = 1
Global Const DA_TYPE = 0
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

' definition for error type in DISK ARRAY
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
Global Const DA_CAN_NOT_SEEK = 17
Global Const DA_INVALID_HANDLE = 18
Global Const DA_CAN_NOT_READ_PART = 19
Global Const DA_CAN_NOT_READ_REMAIN = 20

' definition for error type in HUGE MEMORY ARRAY
Global Const HMA_NO_ERROR = True
Global Const HMA_NO_MEMORY = 1
Global Const HMA_BAD_TYPE = 2
Global Const HMA_BAD_ROWS = 3
Global Const HMA_BAD_COLS = 4
Global Const HMA_BAD_SHEETS = 5
Global Const HMA_INVALID_HANDLE = 6

' definition for error type in SERIAL DATA
Global Const SD_SERIAL_NOT_FOUND = 1
Global Const SD_CAN_NOT_OPEN_FILE = 2

' definition for File I/O
Global Const EOFILE = -1
Global Const SEEK_CUR = 1
Global Const SEEK_END = 2
Global Const SEEK_SET = 0

' definition for file sort
Global Const SORT_ASCENDING = 1
Global Const SORT_DESCENDING = 2
Global Const SORT_CASE_SENSITIVE = 4
Global Const SORT_CASE_INSENSITIVE = 8

' definition for compress/expand
Global Const LZH_ENCODE = True
Global Const LZH_DECODE = False

' definition for PROPERNAME2
Global Const PN_UPPERCASE = 1
Global Const PN_PUNCTUATION = 2
Global Const PN_KEEP_ORIGINAL = 4
Global Const PN_ONLY_LEADING_SPACE = 8

' definition for matrix fill
Global Const MATRIX_ZERO = 0
Global Const MATRIX_UNIT = 1

' definition for FX picture
Global Const FX_HORIZONTAL = 1
Global Const FX_VERTICAL = 2
Global Const FX_DIAGONAL_SQUARE = 3
Global Const FX_RECTANGLE = 4

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

' structure for file version information
Type tagFILEVERSIONINFO
        VersionProduct          As String
        VersionFile             As String
        CompanyName             As String
        FileDescription         As String
        FileVersion             As String
        InternalName            As String
        LegalCopyright          As String
        LegalTrademarks As String
        Comments                As String
        ProductName             As String
        ProductVersion          As String
End Type

' structure for file attributes
Type FileAttributeType
        ErrNo                   As Integer
        Archive                 As Integer
        Hidden                  As Integer
        Normal                  As Integer
        ReadOnly                As Integer
        SubDir                  As Integer
        System                  As Integer
        VolId                   As Integer
End Type

' structure for VB array
Type ArrayType
        Bounds                  As Long
        LBound                  As Integer
        UBound                  As Integer
        ElemSize                As Integer
        IndexCount              As Integer
        TotalElem               As Integer
End Type

' structure for modules
Type tagMODULEENTRY
        dwSize                  As Long
        szModule                As String * 10
        hModule         As Integer
        wcUsage                 As Integer
        szExePath               As String * 256
        wNext                   As Integer
End Type

' structure for tasks
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

' structure for disk array
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 for the last correct transaction
        nIsTyped                As Integer                      'is nType a type'd variable
        dummy                   As String * 7                   'reserved for future use
End Type

' structure for huge memory array
Type tagHMA
        daSize                  As Integer                      'size of the type'd
        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 Long                         'returned size of a element
        rMemorySize             As Long                         'returned size of the memory used
        rParts                  As Long                         'returned total part
        rRemain         As Long                         'returned size of the remain part
        rSheetSize              As Long                         'size of a sheet
        rOffset                 As Long                         'returned offset
        nIsTyped                As Integer                      'is nType a type'd variable
        dummy                   As String * 20                  'reserved for future use
End Type

' structure for serialization
Type tagSERIALDATA
        Description1            As String * 50                  'serialization description 1
        Description2            As String * 50                  'serialization description 2
        Number                  As Long                         'serialization number
        dummy                   As String * 50                  'reserved for future use
End Type

' structure for ARRAYSTRINGONDISK and FILESINDIRTOARRAY
Type tagVARSTRING
        Contents                As String
End Type

' structure for 2-D geometry types
Type tagVECTOR2
        x                       As Double
        y                       As Double
End Type

' structure for 3-D geometry types
Type tagVECTOR3
        x                       As Double
        y                       As Double
        z                       As Double
End Type

' structure for get/set Media ID
Type tagMEDIAID
        InfoLevel               As Integer
        SerialNumber            As Long
```

```
	VolLabel		As String * 11
	FileSysType			As String * 8
End Type

' structure for Get Control Rectangle
Type tagRECT
	Left				As Integer
	Top				As Integer
	Right				As Integer
	Bottom				As Integer
End Type

' structure for 3D-Meter
Type tag3DMeter
	CrtValue		As Long		'current value
	MaxValue			As Long		'maximum value
	BackColor			As Long
	ForeColor			As Long
	Polygon				As Integer	'0 : rectangle, 1 : triangle, 2 : trapezium, 3 : ellipse , 4 : bars
	BarSize				As Integer	'size of a bar (polygon = 4) (in pixel :
min=1,max=20,default=10)
	SpaceBars			As Integer	'space between bars (polygon = 4) (in pixel :
min=1,max=4,default=2)
	Direction		As Integer	'0   : horizontal, other : vertical
	ThreeD				As Integer	'-1 : indented, 1 : raised
	Thickness			As Integer
	Percent				As Integer	'internal use, do not change
	OldPolygon			As Integer	'internal use, do not change
	OldDirection			As Integer	'internal use, do not change
	OldThreeD			As Integer	'internal use, do not change
	HatchBrush			As Integer	'-1 : solid brush, 0 : hor., 1 : ver., 2 : downward diag., 3 :
upward diag., 4 : cross, 5 : diag.cross
End Type

' structure for File Information
Type tagFILEINFO
	fSize				As Long		'size of the file
	fDate				As Long		'date of the file (scalar date)
	fTime				As Long		'time of the file (scalar time)
	fAttribute			As Integer	'attribute of the file
End Type
```

EXEnameActiveWindow

Purpose :

EXEnameActiveWindow retrieves the full filename (path and file) of the active window.

Declare Syntax :

Declare Function cEXEnameActiveWindow Lib "t2win-16.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 "t2win-16.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 "t2win-16.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

Align

Purpose :

Align aligns a give string (left, center, right) into an another new string.

Declare Syntax :

Declare Function cAlign Lib "t2win-16.dll" (Txt As String, ByVal TypeAlign As Integer, ByVal NewLength As Integer) As String

Call Syntax :

Test$ = cAlign(Txt$, TypeAlign%, NewLength%)

Where :

Txt$	is the specified string
TypeAlign%	< 0 : left align, = 0 : center align, > 0 : right align.
NewLength%	the length of the new string
Test$	is the string aligned

Comments :

If NewLength is below that the length of the string, the left part of the string is returned.
The new string is padded with spaces.

Examples :

```
Test$ = cAlign("TIME TO WIN", -1, 20)
        -> "TIME TO WIN         "

Test$ = cAlign("TIME TO WIN", 0, 20)
        -> "     TIME TO WIN    "

Test$ = cAlign("TIME TO WIN", 1, 20)
        -> "         TIME TO WIN"
```

See also :

Date, Hour and Time routines

IEEE Conversion routines

Miscellaneous routines

Technical Support

Only registered users can receive support and update.

To receive support, you must specify your registration ID.

However, any report on any problem are the welcome.

The following information may be of help to you in streamlining your efforts to resolve any technical problems you may have with 'TIME TO WIN (16-Bit)' Dynamic Link Library for Visual Basic® 4.0 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
Internet : 100042.3646@compuserve.com

I'm on CompuServe one time a day.

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

Special thanks to J. Kercheval, Michael M. Dodd, Ray Gardner, Bob Stout, Thad Smith.
Special thanks to Brian Pirie for REGISTRATION KEY SYSTEM FOR C PROGRAMMERS.
Special thanks to Andy Brown for MD5 HASH ALGORITHM. (derived from the RSA ** ** Data Security, Inc. MD5 Message-Digest Algorithm).

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

Overview

'TIME TO WIN (16-Bit)' is a DLL (**D**ynamic **L**ink **L**ibrary) only for use with Visual Basic® 4.0 for Windows®.

I'm an Engineer in Electricity and Electronic and I've writed 'TIME TO WIN (16-Bit)' to help any users of VB to find a solution at some missing functions in VB. VB is a powerfull product but by some aspects it is very limited.

I hope that 'TIME TO WIN (16-Bit)' will be a great advantage for you and for your application.

'TIME TO WIN (16-Bit)' contains more over **620** functions or subroutines. You can find functions or routines over the following sections :

• 2-D Geometry
• 3-D Geometry
• 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
• File Input/Output from C
• Financial (interest rate)
• Huge Memory Arrays
• Huge Strings
• IEEE Conversion routines
• IsX Family Test routines
• Matrix
• Miscellaneous routines
• Multi-Language support
• Multiple Disk Array routines
• Serialization
• String Manipulation routines
• Timer functions
• Type functions
• VB Control Specific routines
• Windows Specific routines

Registering 'TIME TO WIN (16-Bit)'

The easiest way to Register 'TIME TO WIN (16-Bit)' is through CompuServe's SWREG forum.

1) GO SWREG
2) Choose Register Shareware.
3) 'TIME TO WIN (16-Bit)' 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 a license file for one site (only if lastest version is available (not currently in test)) if not you receive the license file for one site.

You also qualify to receive new versions of 'TIME TO WIN (16-Bit)' during one year.

The price for 'TIME TO WIN (16-Bit)' is fixed at $61.00

This price is much a contribution to my works that a payment. When you register 'TIME TO WIN (16-Bit)', you help me to develop better products and others products.

'TIME TO WIN (16-Bit)' is written in C and has been compiled using Visual C++ 1.51.
The code has been optimized for 80386 use with the 'maximize speed' option.

'TIME TO WIN (16-Bit)' can only be used with Visual Basic 4.0.

Others products :

1) TIME TO WIN Light : Light version of 'TIME TO WIN' (360 routines).

1.1) TIME TO WIN Light is $25.00. SWREG ID is : #**5808**.

2) VB/Error Handler : Add/Remove error's management into a VB application by treatment of all files (.FRM, .BAS, .INC) in the .MAK project.

2.1) VB/Error Handler for ONLY REGISTERED USER of 'TIME TO WIN (16-Bit)' is $20.00. SWREG ID is : #**4379**.
2.2) VB/Error Handler for UN-REGISTERED USER of 'TIME TO WIN (16-Bit)' is $30.00. SWREG ID is : #**4380**.

3) VB/Tracer-Profiler : Add/Remove trace/profile information into VB application by treatment of all files (.FRM, .BAS, .INC) in the .MAK project.

3.1) VB/Tracer-Profiler for ONLY REGISTERED USER of 'TIME TO WIN (16-Bit)' is $25.00. SWREG ID is : #**5295**.
3.2) VB/Tracer-Profiler UN-REGISTERED USER of 'TIME TO WIN (16-Bit)' is $34.00. SWREG ID is : #**5294**.

4) Bundle of TIME TO WIN, VB/Error Handler, VB/Tracer-Profiler

Three products : TIME TO WIN, VB/Error Handler, VB/Tracer Profiler for the INCREDIBLE price of $99.00. SWREG ID is : #**5499**.

SwapD

Purpose :

SwapD swaps two Double values.

Declare Syntax :

Declare Sub cSwapD Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetArchive Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetHidden Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetReadOnly Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetSystem Lib "t2win-16.dll" (ByVal nFilename As String) As Integer
Declare Function cFileSetFlag Lib "t2win-16.dll" (ByVal nFilename As String, ByVal nStatus As Integer) As Integer

Declare Function cFileSetAttrib Lib "t2win-16.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 "t2win-16.dll" (ByVal nFileName As String, ByVal Search As String, ByVal sensitivity As Integer) As Long
Declare Function cFileSearchCount Lib "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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 "t2win-16.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.
cDAClearCol	clear a single col on on a sheet in a big sized array.
cDAClearRow	clear a single row on a sheet in a big sized array.
cDAClearSheet	clear a single sheet in a big sized array.
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.
cDAsClearCol	clear a single col on on a sheet in a big sized array with only one sheet.
cDAsClearRow	clear a single row on a sheet in a big sized array with only one sheet.
cDAsGet	read an element from a big sized array on disk with only one sheet.
cDAsGetType	read a type'd variable from a big sized array on disk with only one sheet.
cDAsPut	save an element to a big sized array on disk with only one sheet.
cDAsPutType	save a type'd variable to a big sized array on disk with only one sheet.
cDArGet	read an element from a big sized array on disk with only one sheet and one row.
cDArGetType	read a type'd variable from a big sized array on disk with only one sheet and one row.
cDArPut	save an element to a big sized array on disk with only one sheet and one row.
cDArPutType	save a type'd variable to a big sized array on disk with only one sheet and one row.

To minimize the use of too many functions for the different variable type in VB, 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.

To handle type'd variable, you must use cDAGetType, cDAPutType.

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.

Actually, the maximum number of chars for a string element or for a type'd variable is 4096.

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 "t2win-16.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 maxixum number of Rows is 2147483647
The maxixum number of Cols is 2147483647
The maxixum number of Sheets is 2147483647

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

The length of the filename can be 64 chars maximum.

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

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

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

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

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

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

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

Examples :

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

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

ErrCode = cDACreate(DA, True)                           'create a new big sized array on disk

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

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

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

Var(5) = cDAGet(DA, 1, 1, 2)                            'read the string in Row 1, Col 1, Sheet 2
Var(6) = cDAGet(DA, 1, DA.nCols, 2)                     'read the string in Row 1, Col 500, Sheet 2
Var(7) = cDAGet(DA, DA.nRows, 1, 2)                     'read the string in Row 500, Col 1, Sheet 2
Var(8) = cDAGet(DA, DA.nRows, DA.nCols, 2)              'read the string in Row 500, Col 500, Sheet 2

Call cDAClose(DA, False)                                'close the file without delete it.
```

On my system :

```
ErrCode = -1                                            'no error

DA.daSize = 128                                         'internal header size
DA.Signature   = "MCR_347"                              'internal signature
DA.nFilename = "c:\t2w_tmp\dastring.tmp"                'name fo the file
DA.nType = 50                                           'fixed string of 50 chars
DA.nRows = 500                                          '500 rows
DA.nCols = 500                                          '500 cols
DA.nSheets = 2                                          '2 sheets
DA.rHandle = 0                                          'internal handle
DA.rElementSize = 50                                    'internal size of a element
DA.rFileSize = 25000128                                 'internal size of the file
DA.rParts = 762                                         'internal number of parts (block of 32768
chars)
DA.rRemain = 30784                                      'internal remain chars
DA.rSheetSize = 250000                                  'internal size of one sheet
DA.rTime = 26639                                        'internal time to perform the operation

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

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

See also : Disk Array 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 "t2win-16.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, DArGet, DAsGet

Purpose :

DAGet reads an element from a big sized array on disk.
DArGet have the same functionnality but with a big sized array with only one sheet and only one row.
DAsGet have the same functionnality but with a big sized array with only one sheet.

Declare Syntax :

Declare Function cDAGet Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long) As Variant
Declare Function cDArGet Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long) As Variant
Declare Function cDAsGet Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long) As Variant

Call Syntax :

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

Where :

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

Comments :

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

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.
If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.
If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

Examples :

see cDACreate

See also : Disk Array routines, cDAPut

DAPut, DArPut, DAsPut

Purpose :

DAPut saves an element to a big sized array on disk.
DArPut have the same functionnality but with a big sized array with only one sheet and only one row.
DAsPut have the same functionnality but with a big sized array with only one sheet.

Declare Syntax :

Declare Sub cDAPut Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, Var As Variant)
Declare Sub cDArPut Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long) As Variant
Declare Sub cDAsPut Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col 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 1, the Sheet 1 is used.

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.
If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.
If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

Examples :

see cDACreate

See also : Disk Array routines, cDAGet

DAPutType, DArPutType, DAsPutType

Purpose :

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

Declare Syntax :

```
Declare Sub cDAPutType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As
Long, ByVal Sheet As Long, nType As Any)
Declare Sub cDArPutType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, nType As Any)
Declare Sub cDAsPutType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As
Long, nType As Any)
```

Call Syntax :

```
Call cDAPutType(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 1, the Sheet 1 is used.

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.
If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.
If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

Examples :

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

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

ErrCode = cDACreate(DA, True)                           'create a new big sized array on disk

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

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

See also : Disk Array routines, cDAGetType

DAGetType, DArGetType, DAsGetType

Purpose :

DAGetType reads a type'd variable from a big sized array on disk.
DArGetType have the same functionnality but with a big sized array with only one sheet and only one row.
DAsGetType have the same functionnality but with a big sized array with only one sheet.

Declare Syntax :

Declare Sub cDAGetType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)
Declare Sub cDArGetType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, nType As Any)
Declare Sub cDAsGetType Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)

Call Syntax :

Call cDAGetType(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 1, the Sheet 1 is used.

If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.
If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.
If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

Examples :

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

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

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

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

See also : Disk Array routines, cDAPutType

DAClear

Purpose :

DAClear clears a big sized array (fill it with chr$(0) or chr$(32) (for string array)).

Declare Syntax :

```
Declare Function cDAClear Lib "t2win-16.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"                'name of the file to use
DA.nType = 50                                           'positive value for a string
DA.nIsTyped = False                                     'init the array with spaces
DA.nRows = 500                                          '500 rows
DA.nCols = 500                                          '500 cols
DA.nSheets = 2                                          '2 sheets

ErrCode = cDACreate(DA, True)                           'create a new big sized array on disk

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

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

ErrCode = cDAClear(DA)                                  'clear all elements in the big sized array on
disk
```

See also : Disk Array routines, cDACreate, cDAClearSheet

DAClearSheet

Purpose :

DAClearSheet clears a single Sheet in a big sized array (fill it with chr$(0) or chr$(32) (for string array)).

Declare Syntax :

Declare Function cDAClearSheet Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Sheet As Long) As Integer

Call Syntax :

ErrCode% = cDAClearSheet(DISKARRAY, Sheet&)

Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
Sheet&	is the desired Sheet.
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.

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

If the Sheet is -1 then all Sheets are used. This parameter have the same functionnality that cDAClear
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

Examples :

```
Dim ErrCode                  As Integer
Dim DA                       As tagDISKARRAY

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

ErrCode = cDACreate(DA, True)                           'create a new big sized array on disk

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

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

ErrCode = cDAClearSheet(DA, 1)                          'clear the Sheet 1 in the big sized array on
disk
```

See also : Disk Array routines, cDACreate; cDAClear

Need assistance for some translations in different languages

Actually, 'TIME TO WIN (16-Bit)' supports 8 languages :

French	
Dutch	
English	
German	translated by Andreas Thoele.
Italian	translated by Silvio Sorrentino.
Spanish	translated by Manuel Tobarra Narro.
Polish	translated by Pawel Mandalian.
Catalan	translated by Joan Ludevid.

If you're fluent in an another language, can you translate the following texts that I can include in my product :

```
long month      :
"January","February","March","April","May","June","July","August","September","October","November","December"
short month     : "Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"
tiny month      : "J","F","M","A","M","J","J","A","S","O","N","D"

long day        : "Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"
short day       : "Sun","Mon","Tue","Wed","Thu","Fri","Sat"
small day       : "Su","Mo","Tu","We","Th","Fr","Sa"
tiny day        : "S","M","T","W","T","F","S"

system menu     : "&Restore","&Move","&Size","Mi&nimize","Ma&ximize","&Close\tAlt+F4","S&witch To...\tCtrl+Esc"
message box     : "&Move","&Close\tAlt+F4","OK","Cancel","&Abort","&Retry","&Ignore","&Yes","&No"
```

Thanks you for any translation.

You can post any translations on CompuServe :

Name : Michaël RENARD
CIS : 100042,3646
Internet : 100042.3646@compuserve.com

DAClearCol, DAsClearCol

Purpose :

DAClearCol clears a single Col on one Sheet or on all Sheets in a big sized array (fill it with chr$(0) or chr$(32) (for string array)).
DAsClearCol have the same functionnality but with a big sized array with only one sheet.

Declare Syntax :

Declare Function cDAClearCol Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long, ByVal Sheet As Long) As Integer
Declare Function cDAsClearCol Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Col As Long) As Integer

Call Syntax :

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

Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
Col&	is the desired Col.
Sheet&	is the desired Sheet.
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.

If the Col is below 1, the Col 1 is used.
If the Col is greater than DISKARRAY.nCols, the Col DISKARRAY.nCols is used.

If the Sheet is -1 then all Sheets are used.
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

Examples :

```
Dim ErrCode          As Integer
Dim DA               As tagDISKARRAY

DA.nFilename = "c:\t2w_tmp\dastring.tmp"                    'name of the file to use
DA.nType = 50                                               'positive value for a string
DA.nIsTyped = False                                         'init the array with spaces
DA.nRows = 500                                              '500 rows
DA.nCols = 500                                              '500 Cols
DA.nSheets = 2                                              '2 Sheets

ErrCode = cDACreate(DA, True)                               'create a new big sized array on disk

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

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

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

See also : Disk Array routines, cDACreate; cDAClear, cDAClearRow

DAClearRow, DAsClearRow

Purpose :

DAClearRow clears a single Row on one Sheet or on all Sheets in a big sized array (fill it with chr$(0) or chr$(32) (for string array)).
DAsClearRow have the same functionnality but with a big sized array with only one sheet.

Declare Syntax :

Declare Function cDAClearRow Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long, ByVal Sheet As Long) As Integer
Declare Function cDAsClearRow Lib "t2win-16.dll" (DISKARRAY As tagDISKARRAY, ByVal Row As Long) As Integer

Call Syntax :

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

Where :

DISKARRAY	is a type'd variable (tagDISKARRAY).
Row&	is the desired Row.
Sheet&	is the desired Sheet.
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.

If the Row is below 1, the Row 1 is used.
If the Row is greater than DISKARRAY.nRows, the Row DISKARRAY.nRows is used.

If the Sheet is -1 then all Sheets are used.
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than DISKARRAY.nSheets, the Sheet DISKARRAY.nSheets is used.

Examples :

```
Dim ErrCode          As Integer
Dim DA               As tagDISKARRAY

DA.nFilename = "c:\t2w_tmp\dastring.tmp"                  'name of the file to use
DA.nType = 50                                             'positive value for a string
DA.nIsTyped = False                                       'init the array with spaces
DA.nRows = 500                                            '500 Rows
DA.nCols = 500                                            '500 cols
DA.nSheets = 2                                            '2 Sheets

ErrCode = cDACreate(DA, True)                             'create a new big sized array on disk

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

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

```
ErrCode = cDAClearRow(DA, DA.nRows, 1)                    'clear the last Row in Sheet 1 in the big sized
array on disk
```

See also : Disk Array routines, cDACreate; cDAClear, cDAClearCol

Combination

Purpose :

Combination computes C(n,m) which is the number of combinations of n items, taken m at a time.

Declare Syntax :

```
Declare Function cCombination Lib "t2win-16.dll" (ByVal nItems As Integer, ByVal mTimes As Integer) As Double
```

Call Syntax :

```
Test# = cCombination(nItems%, mTimes%)
```

Where :

nItems	the number of items.
mTimes%	the number taken.
Test#	the result.

Comments :

If nItems is below 0 or if mTimes is not between 0 and nItems, the result is -1.
Beware of using to big nItems and/or mTimes, this gives an overflow.

Examples :

```
Debug.Print cCombination(42, 0)          -> 1
Debug.Print cCombination(42, 1)          -> 42
Debug.Print cCombination(42, 2)          -> 861

Debug.Print cCombination(42, 42)         -> 1
Debug.Print cCombination(42, 41)         -> 42
Debug.Print cCombination(42, 40)         -> 861
```

See also :

Affected routines

The routines below are affected by the new method of allocation temporary memory to handle string :

cCompact
cCompress
cCompressTab
cCplAlpha
cCplDigit
cCreateAndFill
cCreateBits
cExpandTab
cFileDateCreated
cFileDrive
cFileLastDateAccess
cFileLastDateModified
cFileLastTimeAccess
cFileLastTimeModified
cFileTimeCreated
cFilterBlocks
cFilterChars
cFilterFirstChars
cFilterNotChars
cFromBinary
cFromBinary2
cFromHexa
cGetCurrentDrive
cGetNetConnection
cGiveBitPalindrome
cInsertBlocks
cInsertBlocksBy
cInsertByMask
cInsertChars
cIntoDate
cIntoDateFill
cIntoDateNull
cIntoVarHour
cLrc
cOneCharFromLeft
cOneCharFromRight
cRemoveBlockChar
cRemoveOneChar
cResizeString
cResizeStringAndFill
cReverse
cScrollL
cScrollR
cToBinary
cToBinary2
cToHexa
cUnCompact

internal FixHour
internal GetWinINI
internal GetWinINI2

ScrollL, ScrollR

Purpose :

ScrollL scrolls one char to the left of a specified string.
ScrollR scrolls one char to the right of a specified string.

Declare Syntax :

```
Declare Function cScrollL Lib "t2win-16.dll" (Txt As String) As String
Declare Function cScrollR Lib "t2win-16.dll" (Txt As String) As String
```

Call Syntax :

```
test$ = cScrollL(Txt$)
test$ = cScrollR(Txt$)
```

Where :

Txt$	is the string to scroll.
test$	is the string scrolled to the left or to the right.

Comments :

The size of the string must be greater than 1.

Examples :

```
Txt$ = "TIME TO WIN "

test$ = cScrollL(Txt$)          "IME TO WIN T"
test$ = cScrollR(Txt$)          " TIME TO WIN"
```

See also :

RegistrationKey

Purpose :

RegistrationKey performs the calculation of a key from a name and a code.

Declare Syntax :

```
Declare Function cRegistrationKey Lib "t2win-16.dll" (ByVal RegString As String, ByVal RegCode As Long) As Long
```

Call Syntax :

```
Key& = cRegistrationKey(RegString$, RegCode&)
```

Where :

RegString$	the name for the registration.
RegCode&	the basis code for the registration.
Key&	= 0, if length of RegString is < 10 or if RegCode is 0, <>0, the key calculated from RegString and RegCode.

Comments :

Using this registration key system, you can easily and quickly generate and verify the validity of numerical registration keys that correspond to a person who has purchased your program. Thus, when someone who already has a shareware or demo version of your program wishes to purchase the program, you need only send them a simple registration key number, instead of sending an entire registered version. You can simply use this package to generate a unique registration key number which corresponds to the user's name (or any other string you wish to use). The user will then be able to enter this number into your software's configuration file / configuration program. When your program begins, it will be able to read this number from the configuration file, and again using this package, determine whether it is a valid registration key corresponding to the user's name. If the registration key is valid, your program can switch into "registered mode", and if not, can run in its unregistered "unregistered mode". (Source from Brian Pirie).

Examples :

```
Dim Key          As Long

Key = cRegistration("TIME TO WIN", 12345)
        -> 39951

Key = cRegistration(LCase$("TIME TO WIN"), 12345)
        -> -1119769877
```

See also : cHashMD5

ObjectMethod, ObjectGetProperty, ObjectPutProperty

Purpose :

ObjectMethodByPos give the access of method (by position) of OCX custom controls.
ObjectMethodByName give the access of method (by name) of OCX custom controls.
ObjectGetPropertyByPos read data in properties (by position) from OCX custom controls.
ObjectGetPropertyByName read data in properties (by name) from OCX custom controls.
ObjectPutPropertyByPos write data in properties (by position) in OCX custom controls.
ObjectPutPropertyByName write data in properties (by name) from OCX custom controls.

Declare Syntax :

```
Declare Sub cObjectMethodByPos Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer, lpPut As Variant)
Declare Function cObjectGetPropertyByPos Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer) As Variant
Declare Sub cObjectPutPropertyByPos Lib "t2win-16.dll" (Obj As Object, ByVal Property As Integer, lpPut As Variant)
Declare Sub cObjectMethodByName Lib "t2win-16.dll" (Obj As Object, ByVal Property As String, lpPut As Variant)
Declare Function cObjectGetPropertyByName Lib "t2win-16.dll" (Obj As Object, ByVal Property As String) As Variant
Declare Sub cObjectPutPropertyByName Lib "t2win-16.dll" (Obj As Object, ByVal Property As String, lpPut As
Variant)
```

Call Syntax :

```
Call cObjectMethodByPos(Obj, Property%, varPut)
Call cObjectMethodByName(Obj, Property$, varPut)
varGet = cObjectGetPropertyByPos(Obj, Property%)
varGet = cObjectGetPropertyByName(Obj, Property$)
Call cObjectPutPropertyByPos(Obj, Property%, varPut)
Call cObjectPutPropertyByName(Obj, Property$, varPut)
```

Where :

Obj	is a valid object (Form, OCX custom control, VBX custom control);
Property%	is a constant for accessing the data (see Constants and Types declaration);
Property$	is a valid property;
varPut	is a data in a type variant;
varGet	is the returned data in a type variant.

Comments :

For cObjectGetProperty?, if the property don't exist the returned variant is EMPTY

Examples :

```
Dim varGet                    As Variant

Call cObjectPutPropertyByPos(Frame1, OBJ_CAPTION, "this is a test")
varGet = cObjectGetPropertyByPos(Frame1, OBJ_CAPTION)                    '---> this is a test

Call cObjectPutPropertyByName(Frame1, "caption", "this is an another test")
varGet = cObjectGetPropertyByName(Frame1, "caption")                     '---> this is an another test

Call cObjectMethodByName(List1, "clear", Empty)
```

See also :

CloseAllEditForm

Purpose :

CloseAllEditForm closes all VB edit form in the design environnement (windows with code only, the others are already closed by VB himself).

Declare Syntax :

```
Declare Function cCloseAllEditForm Lib "t2win-16.dll" () As Integer
```

Call Syntax :

```
test% = cCloseAllEditForm()
```

Where :

test%	TRUE if all is correct, FALSE if an error has occured.

Comments :

CloseAllEditForm use the Windows Enumeration to find which window class is an VB edit form.

Examples :

```
Dim Test		As Integer

Test = cCloseAllEditForm()
```

See also : cHideAllEditForm, cUnHideAllEditForm, cHideDebugForm, cUnHideDebugForm

Thanks you to register 'TIME TO WIN (16-Bit)'.
SWREG #4045, price $61.00

DecrI, DecrL

Purpose :

DecrI auto-decrement an integer value by 1.
DecrL auto-decrement a long value by 1.

Declare Syntax :

Declare Sub cDecrI Lib "t2win-16.dll" (Value As Integer)
Declare Sub cDecrL Lib "t2win-16.dll" (Value As Long)

Call Syntax :

cDecrI Value%
cDecrL Value&

Where :

Value%	is the integer value to auto-decrement.
Valeu&	is the long value to auto-decrement.

Comments :

These routines are slower than the VB equivalent : Value = Value - 1 but are shorter to type.

Examples :

```
Dim Value As Integer

Value = 5

cDecrI Value            -> 4
cDecrI Value            -> 3
```

See also : cIncrI, cIncrL

HideDebugForm, UnHideDebugForm

Purpose :

HideDebugForm hides the debug window in the design environnement.
UnHideDebugForm unhides the debug window in the design environnement.

Declare Syntax :

```
Declare Function cHideDebugForm Lib "t2win-16.dll" () As Integer
Declare Function cUnHideDebugForm Lib "t2win-16.dll" () As Integer
```

Call Syntax :

```
test% = cHideDebugForm()
test% = cUnHideDebugForm()
```

Where :

test%	TRUE if all is correct, FALSE if an error has occured.

Comments :

HideDebugForm use the Windows Enumeration to find which window class is an VB debug form.
UnHideDebugForm use the Windows Enumeration to find which window class is an VB debug form.

Examples :

```
Dim Test            As Integer

Test = cHideDebugForm()
../.. some pieces of code
Test = cUnHideDebugForm()
```

See also : cCloseAllEditForm, cHideAllEditForm, cUnHideAllEditForm

HideAllEditForm, UnHideAllEditForm

Purpose :

HideAllEditForm hides all VB edit form in the IDE (windows with code only, the others are already closed by VB himself).
UnHideAllEditForm unhides all VB edit form in the IDE (windows with code only, the others are already closed by VB himself).

Declare Syntax :

Declare Function cHideAllEditForm Lib "t2win-16.dll" () As Integer
Declare Function cUnHideAllEditForm Lib "t2win-16.dll" () As Integer

Call Syntax :

test% = cHideAllEditForm()
test% = cUnHideAllEditForm()

Where :

test%	TRUE if all is correct, FALSE if an error has occured.

Comments :

HideAllEditForm use the Windows Enumeration to find which window class is an VB edit form.
UnHideAllEditForm use the Windows Enumeration to find which window class is an VB edit form.

Examples :

Dim Test As Integer

../.. in a Form_Load event
Test = cHideAllEditForm()
../.. in a Form_UnLoad or Form_QueryUnLoad event
Test = cUnHideAllEditForm()

See also : cCloseAllEditForm, cHideDebugForm, cUnHideDebugForm

WalkThruWindow

Purpose :

WalkThruWindow walk in the window's list of all windows at a gived moment.

Declare Syntax :

Declare Function cWalkThruWindow Lib "t2win-16.dll" (Class As String, Caption As String, OwnerHwnd As Integer, OwnerClass As String, OwnerCaption As String, ByVal FirstNext As Integer) As Integer

Call Syntax :

hWnd% = cWalkThruWindow(Class$, Caption$, OwnerHwnd%, OwnerClass$, OwnerCaption$, FirstNext%)

Where :

Class$	is the returned Name of the Window's Class for the hWnd founded.
Caption$	is the returned Caption of the Window for the hWnd founded.
OwnerHwnd%	is the returned hWnd of the Owner for the hWnd founded
OwnerClass$	is the returned Name of the Window's Class for the Owner for the hWnd founded.
OwnerCaption$	is the returned Caption of the Window for the Owner for the hWnd founded.
FirstNext%	TRUE to begin the search, FALSE to continue the search.
hWnd%	is the returned hWnd founded.

Comments :

Examples :

```
        Dim nClass              As String
        Dim nCaption            As String
        Dim nOwnerClass         As String
        Dim nOwnerCaption       As String
        Dim nOwnerHwnd          As Integer

        Dim nhWnd               As Integer

        nhWnd = cWalkThruWindow(nClass, nCaption, nOwnerHwnd, nOwnerClass, nOwnerCaption, True)

        Do While (nhWnd <> 0)
                Debug.Print "Owner     = "; Hex$(nOwnerHwnd) & Chr$(9) & nOwnerCaption & " (" & nOwnerClass
& ")"
                Debug.Print "Window = "; Hex$(nhWnd) & Chr$(9) & nCaption & " (" & nClass & ")"
                nhWnd = cWalkThruWindow(nClass, nCaption, nOwnerHwnd, nOwnerClass, nOwnerCaption,
False)
        Loop
```

Part of the output on my system :

```
        Owner    = 42A4         Microsoft Visual Basic (ThunderMain)
        Window = 41BC                   Time To WIN (Demo) (ThunderForm)
        Owner    = 42A4         Microsoft Visual Basic (ThunderMain)
        Window = 5878                   (ToolsPalette)
        Owner    = 42A4         Microsoft Visual Basic (ThunderMain)
        Window = 56D4                   TIME2WIN.MAK (PROJECT)
        Owner    = 42A4         Microsoft Visual Basic (ThunderMain)
        Window = 5B20                   Debug Window [TIME2WIN.FRM] (OFEDT)
        Owner    = 42A4         Microsoft Visual Basic (ThunderMain)
```

```
    Window = 48AC               Microsoft Visual Basic [run] (wndclass_desked_gsk)
    Owner    = 4A68         Properties (wndclass_pbrs)
    Window = 59A8               (CBar)
    Owner    = 42A4         Microsoft Visual Basic (ThunderMain)
    Window = 4A68               Properties (wndclass_pbrs)
    Owner    = 42A4         Microsoft Visual Basic (ThunderMain)
    Window = 5928               (CPal)
    Owner    = 0                ()
    Window = 42A4               Microsoft Visual Basic (ThunderMain)
```

See also :

IsSerial, SerialGet, SerialInc, SerialPut, SerialRmv

Purpose :

IsSerial checks if a file has been serialized.
SerialGet gets the serialization information from a serialized file.
SerialInc increment by a value the serialized number part of a serialized file.
SerialPut puts a serialization information to a serialized file.
SerialRmv removes the serialization information from a serialized file.

Declare Syntax :

```
Declare Function cIsSerial Lib "t2win-16.dll" (ByVal File As String) As Integer
Declare Function cSerialGet Lib "t2win-16.dll" (ByVal file As String, SERIALDATA As tagSERIALDATA) As Integer
Declare Function cSerialInc Lib "t2win-16.dll" (ByVal file As String, ByVal Increment As Long) As Integer
Declare Function cSerialPut Lib "t2win-16.dll" (ByVal file As String, SERIALDATA As tagSERIALDATA) As Integer
Declare Function cSerialRmv Lib "t2win-16.dll" (ByVal File As String) As Integer
```

Call Syntax :

```
Test% = cIsSerial(File$)
Test% = cSerialGet(File$, SERIALDATA)
Test% = cSerialInc(File$, Increment&)
Test% = cSerialPut(File$, SERIALDATA)
Test% = cSerialRmv(File$)
```

Where :

File$	is the specified file.
SERIALDATA	is a type'd variable (tagSERIALDATA).
Increment&	is the increment (positive or negative).
Test%	TRUE if all is ok, <> TRUE if an error has occured.

Comments :

The length of the serialization string is maximum 50 characters (SERIALDATA.Description1, SERIALDATA.Description2).
For SerialInc, if you pass a 0 value, the serialization number is reset to 0 (be care).

Examples :

```
        Dim putSERIALDATA       As tagSERIALDATA
        Dim getSERIALDATA       As tagSERIALDATA

        putSERIALDATA.Description1 = "1234567890123456789012345"
        putSERIALDATA.Description2 = ""
        putSERIALDATA.Number = 987654321
        Debug.Print cSerialPut("c:\tmp\sample.exe", putSERIALDATA)
        Debug.Print cSerialGet("c:\tmp\sample.exe", getSERIALDATA)
        Debug.Print getSERIALDATA.Description1 & Chr$(13) & getSERIALDATA.Description2 & Chr$(13) &
getSERIALDATA.Number

        putSERIALDATA.Description2 = "ABCDEFGHIJKLMNOPQRSTUVWYZ"
        putSERIALDATA.Number = 123456789
        Debug.Print cSerialPut("c:\tmp\sample.exe", putSERIALDATA)
        Debug.Print cSerialGet("c:\tmp\sample.exe", getSERIALDATA)
        Debug.Print getSERIALDATA.Description1 & Chr$(13) & getSERIALDATA.Description2 & Chr$(13) &
getSERIALDATA.Number
```

```
        Debug.Print cSerialInc("c:\tmp\sample.exe", 123)
        Debug.Print cSerialGet("c:\tmp\sample.exe", getSERIALDATA)
        Debug.Print getSERIALDATA.Description1 & Chr$(13) & getSERIALDATA.Description2 & Chr$(13) &
getSERIALDATA.Number

        Debug.Print cSerialRmv("c:\tmp\sample.exe")
```

See also :

Serialization

Serialization is a set of routines primarily intended for developers so that they may append a serial number (or other identifier) to the end of an .exe, .dll or any static files in size, put/modify or get serial numbers or any string to 50 characters.   Users may use to initialize purchased software applications with ownership, security-related, or other identifying marks.

A unique serial number going out with each copy of an application affords the developer with a possible opportunity to identify, if need be, the
registered client of a particular copy.   The end-user is normally unaware of the existence of such a mark, its location, its method of placement or
the method of reading/verifying.   Its absence or modification may provide evidence of tampering.

The serialization of a file adds an overhead of 200 bytes to the specified file.

cIsSerial
cSerialGet
cSerialInc
cSerialPut
cSerialRmv

TimerOpen, TimerStart, TimerRead, TimerClose

Purpose :

TimerOpen opens a timer and return an handle of an available timer (1 to 64).
TimerStart starts the selected timer's handle.
TimerRead reads the current value of the selected timer's handle.
TimerClose closes the selected timer's handle.

Declare Syntax :

```
Declare Function cTimerOpen Lib "t2win-16.dll" () As Integer
Declare Function cTimerStart Lib "t2win-16.dll" (ByVal TimerHandle As Integer) As Integer
Declare Function cTimerRead Lib "t2win-16.dll" (ByVal TimerHandle As Integer) As Long
Declare Function cTimerClose Lib "t2win-16.dll" (ByVal TimerHandle As Integer) As Integer
```

Call Syntax :

```
TimerHandle% = cTimerOpen()
StartOk% = cTimerStart(TimerHandle%)
Test& = cTimerRead(TimerHandle%)
CloseOk% = cTimerClose(TimerHandle%)
```

Where :

TimerHandle%	>0 is one timer is available, = 0 if no timers available..
StartOk%	TRUE if the starting is successfully, FALSE if the starting fail.
Test&	is the current value of the specified timer handle.
CloseOk%	TRUE if the closing is successfully, FALSE if the closing fail.

Comments :

These timers functions is independant of the calling program.
The value of timers is in milliseconds.
The accuracy of timers is 1 milliseconds.

Examples :

```
Dim TimerHandle           As Integer
Dim TimerValue                    As Long

Dim i                             As Long
Dim n                             As Long
Dim StartOk                       As Integer
Dim CloseOk                       As Integer

TimerHandle = cTimerOpen()
StartOk = cTimerStart(TimerHandle)


For i = 1 To 54321
        n = i * 2
Next i

MsgBox "Time (in milliseconds) to perform the test is " & cTimerRead(TimerHandle) & " milliseconds"

CloseOk = cTimerClose(TimerHandle)
```

On my system : "Time (in milliseconds) to perform the test is 330"

See also : Timer functions

FileChangeChars

Purpose :

FileChangeChars replace all chars in a char set by a new char set.

Declare Syntax :

Declare Function cFileChangeChars Lib "t2win-16.dll" (ByVal nFileName As String, CharSet As String, NewCharSet As String, ByVal nFileTemp As String) As Long

Call Syntax :

test& = cFileChangeChars(nFilename$, CharSet$, NewCharSet$, nFileTemp$)

Where :

nFilename$	the ASCII file.
CharSet$	the string to be searched.
NewCharSet$	the replacement string.
nFileTemp$	a temporary file.
test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occured.

Comments :

If the nFilename string is an EMPTY string, the returned value is FALSE.
If the char set string is an EMPTY string, the returned value is FALSE.
If the new char set string is an EMPTY string, the returned value is FALSE.

If the length of char set is different of the length of new char set, the minimum length is used.

If the nFileTemp is an EMPTY string, a default temporary file is used.

The returned value can be negative and have the following value :

-32730	reading error for file 1.
-32740	writing error for file 2.
-32750	opening error for file 1.
-32751	opening error for file 2.

Examples :

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

test& = cFileChangeChars("c:\autoexec.tab", "path", " PATH ", "", False)

See also : cChangeChars

Ctl3D, 3D

Purpose :

Ctl3D adds 3D visibility to a VB control.
3D is a shortcut of Ctl3D.

Declare Syntax :

Declare Sub cCtl3D Lib "t2win-16.dll" (Obj As Object, ByVal LeftTopColor As Long, ByVal RightBottomColor As Long, ByVal Thickness As Integer)
Declare Sub c3D Lib "t2win-16.dll" (Obj As Object, ByVal Method As Integer, ByVal Thickness As Integer)

Call Syntax :

Call Ctl3D(Ctl, LeftTopColor&, RightBottomColor&, Thickness%)
Call 3D(Ctl, Method%, Thickness%)

Where :

Ctl	is a VB control (standard or VBX)
LeftTopColor&	is the color for the left and the top side of the 3D.
RightBottomColor&	is the color for the right and the bottom side of the 3D.
Thickness%	is the 3D depth.
Method%	0 : inner 3D.
	1 : outer 3D.

Comments :

The control Ctl can be a control directly on the form or a control in a container.

3D uses the color &h808080 for left and top side, and &hFFFFFF for right and bottom side.

Examples :

see Form_Paint in the sample.

See also :

ArrayOnDisk

Purpose :

Put/Get full array on/from disk

Declare Syntax :

Declare Function cArrayOnDisk Lib "t2win-16.dll" (ByVal File As String, Array() As Any, ByVal GetPut As Integer) As Long

Call Syntax :

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

Where :

File$	is the file to use.
Array()	is the array with any dimension.
GetPut%	PUT_ARRAY_ON_DISK to put the array on disk, GET_ARRAY_ON_DISK to get the array from disk.
test&	>=0 is the returned length of the file, < 0 is an error occurs (error n° is the negative value of all DA_x values, see Constants and Types declaration).

Comments :

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

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

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

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

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

        'This type replaces

        Dim newString                   As String * 80
```

Examples :

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

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

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

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

```
Next i

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

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

See also : Disk Array routines, cArrayStringOnDisk

ArrangeDesktopIcons

Purpose :

This function arranges all desktop icons.

Declare Syntax :

Declare Sub cArrangeDesktopIcons Lib "t2win-16.dll" ()

Call Syntax :

Call cArrangeDesktopIcons()

Where :

Comments :

Examples :

See also :

FileIO

Purpose :

Fopen opens a file for I/O.
Fclose closes an open stream.
Fgetc reads a single character from a stream.
Fputc writes a single character to a stream.
Fputs writes a line of characters to a stream.
Fgets reads a line of characters from a stream.
Fwrite writes an arbitrary number of characters to a stream.
Fread reads an arbitrary number of characters from a stream.
Fcloseall closes all files opened with fopen.
Fflush flushes buffered I/O to a particular stream to disk.
Fflushall flushes buffered I/O for all open streams to disk.
Feof tests for end-of-file on a stream.
Ferror tests for an error on a stream.
Fclearerr resets the error indicator for a stream.
Fseek moves the file pointer to a specified location.
Ftell gets the current position of a file pointer.
Frewind moves the file pointer to the beginning of a file.

Declare Syntax :

Declare Function cFopen Lib "t2win-16.dll" (ByVal File As String, ByVal Mode As String) As Long
Declare Function cFclose Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Function cFgetc Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Function cFputc Lib "t2win-16.dll" (ByVal char As Integer, ByVal IOstream As Long) As Integer
Declare Function cFputs Lib "t2win-16.dll" (ByVal Txt As String, ByVal IOstream As Long) As Integer
Declare Function cFgets Lib "t2win-16.dll" (Txt As String, ByVal Length As Integer, ByVal IOstream As Long) As Integer
Declare Function cFwrite Lib "t2win-16.dll" (Txt As String, ByVal IOstream As Long) As Integer
Declare Function cFread Lib "t2win-16.dll" (Txt As String, ByVal Length As Integer, ByVal IOstream As Long) As Integer
Declare Function cFcloseall Lib "t2win-16.dll" () As Integer
Declare Function cFflush Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Function cFflushall Lib "t2win-16.dll" () As Integer
Declare Function cFeof Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Function cFerror Lib "t2win-16.dll" (ByVal IOstream As Long) As Integer
Declare Sub cFclearerr Lib "t2win-16.dll" (ByVal IOstream As Long)
Declare Function cFseek Lib "t2win-16.dll" (ByVal IOstream As Long, ByVal offset As Long, ByVal Origin As Integer) As Integer
Declare Function cFtell Lib "t2win-16.dll" (ByVal IOstream As Long) As Long
Declare Sub cFrewind Lib "t2win-16.dll" (ByVal IOstream As Long)

Call Syntax :

see above

Where :

File$	the name to use for streaming.
Mode$	the open mode for the file (see comments).
IOstream&	the returned stream or the stream to use to perform file management.
Char%	the char to write/read in decimal.
Txt$	the string to write/read.
Length%	the length to read a string.
Offset&	the new seek position in the stream.
Origin%	the seeking method (see definition for file I/O in Constants and Types declaration)

Comments :

Code returned by these routines :

Fopen	>= 0	: I/O stream in a long integer.
Fclose	= 0	: all is OK,
	< 0	: error.
Fgetc	>= 0	: the char readed,
	< 0	: error.
Fputc	>= 0	: the char writed,
	< 0	: error.
Fputs	>= 0	: all is OK,
	< 0	: error.
Fgets	= 0	: all is OK,
	< 0	: error.
Fwrite	>= 0	: all is OK,
	< 0	: error.
Fread	>= 0	: all is OK,
	< 0	: error.
Fcloseall	= 0	: all is OK,
	< 0	: error.
Fflush	= 0	: all is OK,
	< 0	: error.
Fflushall	= 0	: all is OK
	< 0	: error.
Feof	= 0	: not EOF,
	= -1	: EOF.
Ferror	= 0	: no error,
	<>0	: error number.
Fseek	= 0	: all is OK,
	< 0	: error.
Ftell	>= 0	: the pointer position,
	< 0	: error.

The character string mode specifies the type of access requested for the file, as follows:

"**r**"	Opens for reading. If the file does not exist or cannot be found, the fopen call will fail.
"**w**"	Opens an empty file for writing. If the given file exists, its contents are destroyed.
"**a**"	Opens for writing at the end of the file (appending); creates the file first if it doesn't exist.
"**r+**"	Opens for both reading and writing. (The file must exist.)
"**w+**"	Opens an empty file for both reading and writing. If the given file exists, its contents are destroyed.
"**a+**"	Opens for reading and appending; creates the file first if it doesn't exist.

When a file is opened with the "**a**" or "**a+**" access type, all write operations occur at the end of the file. Although the file pointer can be repositioned using *cFseek* or *cFrewind*, the file pointer is always moved back to the end of the file before any write operation is carried out. Thus, existing data cannot be overwritten.

When the "**r+**", "**w+**", or "**a+**" access type is specified, both reading and writing are allowed (the file is said to be open for "update"). However, when you switch between reading and writing, there must be an intervening *cFflush*, *cFseek*, or *cFrewind* operation. The current position can be specified for the *cFseek* operation, if desired. In addition to the values listed above, the following characters can be included in mode to specify the translation mode for newline characters:

"t"

Open in text (translated) mode. In this mode, carriage-return-line-feed (CR-LF) combinations are translated into single line feeds (LF) on input and LF characters are translated to CR-LF combinations on output. Also , CTRL+Z is interpreted as an end-of-file character on input. In files opened for reading or for reading/writing, cFopen checks for a CTRL+Z at the end of the file and removes it, if possible. This is done because using the *cFseek* and *cFtell* functions to move within a file that ends with a CTRL+Z may cause cFseek to behave improperly near the end of the file.

"b"

Open in binary (untranslated) mode; the above translations are suppressed.

Examples :

see FileIO.MAK

See also :

File Input/Output from C

The routines below are a direct implementation of C File I/O. You can use the routines to perform some file manipulations. Use these routines with care.

cFopen	Opens a file for I/O	
fopen		
cFclose	Closes an open stream.	
fclose		
cFgetc	Reads a single character from a stream.	
fgetc		
cFputc	Writes a single character to a stream.	
fputc		
cFputs	Writes a line of characters to a stream.	
fputs		
cFgets	Reads a line of characters from a stream.	
fgets		
cFwrite	Writes an arbitrary number of characters to a stream.	fwrite
cFread	Reads an arbitrary number of characters from a stream.	
fread		
cFcloseall	Closes all files opened with fopen.	
_fcloseall		
cFflush	Flushes buffered I/O to a particular stream to disk.	
fflush		
cFflushall	Flushes buffered I/O for all open streams to disk.	
_flushall		
cFeof	Tests for end-of-file on a stream.	
feof		
cFerror	Tests for an error on a stream.	
ferror		
cFclearerr	Resets the error indicator for a stream.	
clearerr		
cFseek	Moves the file pointer to a specified location.	fseek
cFtell	Gets the current position of a file pointer.	
ftell		
cFrewind	Moves the file pointer to the beginning of a file.	
rewind		

Special Offer for Visual Basic User's Group

Order **4** 'TIME TO WIN (16-Bit)' in one call (swreg #4045) and receive **1** Extra in Bonus.
You pay **4** and you receive **5** licenses.
This is a Bargain of **20** %.

Order **7** 'TIME TO WIN (16-Bit)' in one call (swreg #4045) and receive **3** Extra in Bonus.
You pay **7** and you receive **10** licenses.
This is a Bargain of **30** %.

These Bargains will be available for the 32 bits version of 'TIME TO WIN (16-Bit)'.

CnvASCIItoEBCDIC, CnvEBCDICtoASCII

Purpose :

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

Declare Syntax :

```
Declare Sub cCnvASCIItoEBCDIC Lib "t2win-16.dll" (Txt As String)
Declare Sub cCnvEBCDICtoASCII Lib "t2win-16.dll" (Txt As String)
```

Call Syntax :

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

Where :

Txt$	the string to convert

Comments :

Examples :

```
Dim Tmp          As String

Tmp = "A/BC/DEF/GHIJ"

Call cCnvASCIItoEBCDIC(Tmp)
Debug.Print Tmp
        -> ÁaÂÃaÄÅÆaÇÈÉÑ

Call cCnvEBCDICtoASCII(Tmp)
Debug.Print Tmp
        -> A/BC/DEF/GHIJ
```

See also :

FileSort

Purpose :

FileSort sorts an ASCII file or a BINARY file in ascending or descending order with case sensitive or not.

Declare Syntax :

Declare Function cFileSort Lib "t2win-16.dll" (ByVal FileIn As String, ByVal FileOut As String, ByVal SortMethod As Integer, ByVal RecordLength As Long, ByVal KeyOffset As Long, ByVal KeyLength As Long, rRecords As Integer) As Long

Call Syntax :

Test% = cFileSort(FileIn$, FileOut$, SortMethod%, RecordLength&, KeyOffset&, KeyLength&, rRecords%)

Where :

FileIn$	the input file.
FileOut$	the output file.
SortMethod%	a combination of the following constants : SORT_ASCENDING SORT_DESCENDING SORT_CASE_SENSITIVE SORT_CASE_INSENSITIVE
RecordLength&	-1 for an ASCII file, > 0 for a BINARY file.
KeyOffset&	-1 for an ASCII file, >= 0 for a BINARY file.
KeyLength&	-1 for an ASCII file, > 0 for a BINARY file.
rRecords	the number of records treated.
Test&	> 0 if all is OK (the returned value is the total bytes copied), < 0 if an error has occured.

Comments :

The returned value can be negative and have the following value :

-1	file 1 is invalid (empty name).
-2	file 2 is invalid (empty name).
-3	KeyOffset must be specified (RecordLength is used).
-4	KeyOffset must be >= 0 (RecordLength is used).
-5	KeyLength must be > 0 (RecordLength is used).
-6	(KeyOffset + KeyLength) must be <= to RecordLength.
-7	filename 1 must be different of filename 2.
-8	unable to open file 1.
-9	unable to open file 2.
-10	can't allocate memory buffer for no fixed length
-11	can't allocate memory buffer for pointers.
-12	can't read first record.
-13	can't read a record.
-14	too many records (about > 16384).
-15	can't expand memory buffer for pointers.
-16	can't write a record (disk full, disk failure, ...).

FileSort uses memory to perform the sort. You're limited to the memory available and a maximum of about 16384 records.

Examples :

```
Dim rRec                    As Integer

Debug.Print cFileSort("c:\autoexec.bat", "c:\ae1.bat", SORT_ASCENDING + SORT_CASE_INSENSITIVE, -1, -1, -1,
rRec)
```

See also :

HashMD5

Purpose :

Performs the hash algorithm (MD5) to a specified string.

Declare Syntax :

```
Declare Function cHashMD5 Lib "t2win-16.dll" (Text As String) As String
```

Call Syntax :

```
Hash$ = cHashMD5(Text$)
```

Where :

Text$	the specified string (length between 1 to 32767).
Hash$	the returned hashed string.

Comments :

A hash algorithm such as MD5 is often used in cryptosystems to "reduce" a user-supplied passphrase into a sufficient number of bits to use as a key to the system. The following is taken from the Executive Summary section of the Internet RFC that proposes MD5 as a standard.

The [MD5] algorithm takes as input an input message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. (Source from Andy Brown).

HashMD5 is derived from the RSA ** ** Data Security, Inc. MD5 Message-Digest Algorithm.

Examples :

```
Dim Hash            As String

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

See also : cRegistrationKey

Financial : interest rate

Purpose :

AtoF	:	annuity to future value.
AtoFC	:	annuity to future value continuous compounding.
AtoP	:	annuity to present value.
AtoPC	:	annuity to present value continuous compounding.
FtoA	:	future value to annuity.
FtoAC	:	future value to annuity continuous compounding.
FtoP	:	future value to present value.
FtoPC	:	future value to present value continuous compounding.
PtoA	:	present value to annuity.
PtoAC	:	present value to annuity continuous compounding.
PtoF	:	present value to future value.
PtoFC	:	present value to future value continuous compounding.

Declare Syntax :

```
Declare Function cAtoF Lib "t2win-16.dll" (ByVal Interest As Double, ByVal N As Integer) As Double
Declare Function cAtoFC Lib "t2win-16.dll" (ByVal Rates As Double, ByVal N As Integer) As Double
Declare Function cAtoP Lib "t2win-16.dll" (ByVal Interest As Double, ByVal N As Integer) As Double
Declare Function cAtoPC Lib "t2win-16.dll" (ByVal Rates As Double, ByVal N As Integer) As Double
Declare Function cFtoA Lib "t2win-16.dll" (ByVal Interest As Double, ByVal N As Integer) As Double
Declare Function cFtoAC Lib "t2win-16.dll" (ByVal Rates As Double, ByVal N As Integer) As Double
Declare Function cFtoP Lib "t2win-16.dll" (ByVal Interest As Double, ByVal N As Integer) As Double
Declare Function cFtoPC Lib "t2win-16.dll" (ByVal Rates As Double, ByVal N As Integer) As Double
Declare Function cPtoA Lib "t2win-16.dll" (ByVal Interest As Double, ByVal N As Integer) As Double
Declare Function cPtoAC Lib "t2win-16.dll" (ByVal Rates As Double, ByVal N As Integer) As Double
Declare Function cPtoF Lib "t2win-16.dll" (ByVal Interest As Double, ByVal N As Integer) As Double
Declare Function cPtoFC Lib "t2win-16.dll" (ByVal Rates As Double, ByVal N As Integer) As Double
```

Call Syntax :

Where :

In all functions, N is the number of periods.

AtoF	:	Interest	is the effective interest rate per period.
AtoFC	:	Interest	is the nominal interest rate per period.
AtoP	:	Interest	is effective interest rate per period.
AtoPC	:	Interest	is the nominal interest rate per period.
FtoA	:	Interest	is the effective interest rate per period.
FtoAC	:	Interest	is the nominal interest rate per period.
FtoP	:	Interest	is the effective interest rate per period.
FtoPC	:	Interest	is the nominal interest rate per period.
PtoA	:	Interest	is the effective interest rate per period.
PtoAC	:	Interest	is the nominal interest rate per period.
PtoF	:	Interest	is the effective interest rate per period.
PtoFC	:	Interest	is the nominal interest rate per period.

Comments :

If Interest is 0 or N is below or equal to 0, the returned value is -1.

Examples :

See also :

Matrix

Purpose :

MatrixAdd adds two square matrix.
MatrixCoFactor calculates the CoFactor of an element in a square matrix.
MatrixCompare compare two square matrix.
MatrixCopy copy a square matrix.
MatrixDet calculates the Determinant of a square matrix.
MatrixFill fills a square matrix (matrix zero, matrix unit).
MatrixInv inverts a square matrix (determinant can't be nul).
MatrixMinor calculates the Minor of an element in a square matrix.
MatrixMul multiply two square matrix.
MatrixSub substract two square matrix.
MatrixSymToeplitz creates a symmetrical Toeplitz matrix.
MatrixTranspose transpose a square matrix.

Declare Syntax :

Declare Sub cMatrixAdd Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
Declare Function cMatrixCoFactor Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal Row As Integer, ByVal Col As Integer) As Double
Declare Function cMatrixCompare Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
Declare Sub cMatrixCopy Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double)
Declare Function cMatrixDet Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double) As Double
Declare Function cMatrixFill Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal nInit As Integer) As Integer
Declare Function cMatrixInv Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
Declare Function cMatrixMinor Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ByVal Row As Integer, ByVal Col As Integer) As Double
Declare Sub cMatrixMul Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
Declare Sub cMatrixSub Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayB() As Double, ArrayC() As Double)
Declare Function cMatrixSymToeplitz Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double) As Integer
Declare Sub cMatrixTranspose Lib "t2win-16.dll" (ByVal Size As Integer, ArrayA() As Double, ArrayC() As Double)

Call Syntax :

```
Call cMatrixAdd(Size%, ArrayA(), ArrayB(), ArrayC())
Test# = cMatrixCoFactor(Size%, ArrayA(), Row, Col)
Test% = cMatrixCompare(Size%, ArrayA(), ArrayC())
Call cMatrixCopy(Size%, ArrayA(), ArrayC())
Test# = cMatrixDet(Size%, ArrayA())
Test% = cMatrixFill(Size%, ArrayA), nInit%)
Test% = cMatrixInv(Size%, ArrayA(), ArrayC())
Test# = cMatrixMinor(Size%, ArrayA(), Row, Col)
Call cMatrixMul(Size%, ArrayA(), ArrayB(), ArrayC())
Call cMatrixSub(Size%, ArrayA(), ArrayB(), ArrayC())
Test% = cMatrixSymToeplitz(Size%, ArrayA(), ArrayC())
Call cMatrixTranspose(Size%, ArrayA(), ArrayB(), ArrayC())
```

Where :

Size%	is the size for the matrixes.
ArrayA()	is the first square matrix (only double value).
ArrayB()	is the second square matrix (only double value).

ArrayC()	is the result square matrix (only double value).
nInit%	MATRIX_ZERO or MATRIX_UNIT.
Test%	= True, matrixes are the same, = False, matrixes are not the same.

Comments :

These matrixes functions doesn't check if the matrix is really square and if the size is ok.
All matrixes must be the same square (n x n).

Examples :

See the demo file.

See also :

ProperName

Purpose :

ProperName converts the first letter of each word separated by a space in a string to upper case.

Declare Syntax :

Declare Function cProperName Lib "t2win-16.dll" (Txt As String) As String

Call Syntax :

Test$ = cProperName(Txt$)

Where :

Txt$	is the specified string.
Test$	is the returned string.

Comments :

Examples :

macdonald	becomes	Macdonald
mac donald	becomes	Mac Donald
John fitz,jr	becomes	John Fitz,jr
john Fitz, jr	becomes	John Fitz, Jr

See also :

3-D Geometry

Purpose :

V3Add add two 3D vectors.
V3Sub substract two 3D vectors.
V3Combine combine two 3D vectors.
V3Copy copy a 3D vector into an another.
V3Dot calculate the dot of two 3D vectors.
V3Length calculate the length (magnitude) of a 3D vector.
V3Length calculate the length squared (magnitude squared) of a 3D vector.
V3LinearLp perform the linear interpolation of two 3D vector.
V3Mul multiply two 3D vector.
V3Neg perform the negate of a 3D vector.
V3Normalized normalize a 3D vector.
V3Ortho perform the orthogonal transformation of two 3D vector.
V3ScaledNewLength change the x,y of a 3D vector from a new length (magnitude).
V3SegmentLength calculate the length of the segment between the two 3D vector.

Declare Syntax :

Declare Sub cV3Add Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
Declare Sub cV3Sub Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
Declare Sub cV3Combine Lib "t2win-16.dll" (u As tagVECTOR3, ByVal c1 As Double, v As tagVECTOR3, ByVal c2 As Double, w As tagVECTOR3)
Declare Sub cV3Copy Lib "t2win-16.dll" (u As tagVECTOR3, w As tagVECTOR3)
Declare Sub cV3Cross Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
Declare Function cV3Dot Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3) As Double
Declare Function cV3Length Lib "t2win-16.dll" (u As tagVECTOR3) As Double
Declare Function cV3LengthSquared Lib "t2win-16.dll" (u As tagVECTOR3) As Double
Declare Sub cV3LinearIp Lib "t2win-16.dll" (lo As tagVECTOR3, hi As tagVECTOR3, ByVal alpha As Double, w As tagVECTOR3)
Declare Sub cV3Mul Lib "t2win-16.dll" (u As tagVECTOR3, v As tagVECTOR3, w As tagVECTOR3)
Declare Sub cV3Neg Lib "t2win-16.dll" (u As tagVECTOR3)
Declare Sub cV3Normalized Lib "t2win-16.dll" (u As tagVECTOR3)
Declare Sub cV3ScaledNewLength Lib "t2win-16.dll" (u As tagVECTOR3, ByVal newlen As Double)
Declare Function cV3SegmentLength Lib "t2win-16.dll" (p As tagVECTOR3, q As tagVECTOR3) As Double

Call Syntax :

Where :

Comments :

Examples :

See also :

2-D Geometry

Purpose :

V2Add add two 2D vectors.
V2Sub substract two 2D vectors.
V2Combine combine two 2D vectors.
V2Copy copy a 2D vector into an another.
V2Dot calculate the dot of two 2D vectors.
V2Length calculate the length (magnitude) of a 2D vector.
V2Length calculate the length squared (magnitude squared) of a 2D vector.
V2LinearLp perform the linear interpolation of two 2D vector.
V2Mul multiply two 2D vector.
V2Neg perform the negate of a 2D vector.
V2Normalized normalize a 2D vector.
V2Ortho perform the orthogonal transformation of two 2D vector.
V2ScaledNewLength change the x,y of a 2D vector from a new length (magnitude).
V2SegmentLength calculate the length of the segment between the two 2D vector.

Declare Syntax :

Declare Sub cV2Add Lib "t2win-16.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
Declare Sub cV2Sub Lib "t2win-16.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
Declare Sub cV2Combine Lib "t2win-16.dll" (u As tagVECTOR2, ByVal c1 As Double, v As tagVECTOR2, ByVal c2 As Double, w As tagVECTOR2)
Declare Sub cV2Copy Lib "t2win-16.dll" (u As tagVECTOR2, w As tagVECTOR2)
Declare Function cV2Dot Lib "t2win-16.dll" (u As tagVECTOR2, v As tagVECTOR2) As Double
Declare Function cV2Length Lib "t2win-16.dll" (u As tagVECTOR2) As Double
Declare Function cV2LengthSquared Lib "t2win-16.dll" (u As tagVECTOR2) As Double
Declare Sub cV2LinearIp Lib "t2win-16.dll" (lo As tagVECTOR2, hi As tagVECTOR2, ByVal alpha As Double, w As tagVECTOR2)
Declare Sub cV2Mul Lib "t2win-16.dll" (u As tagVECTOR2, v As tagVECTOR2, w As tagVECTOR2)
Declare Sub cV2Neg Lib "t2win-16.dll" (u As tagVECTOR2)
Declare Sub cV2Normalized Lib "t2win-16.dll" (u As tagVECTOR2)
Declare Sub cV2Ortho Lib "t2win-16.dll" (u As tagVECTOR2, w As tagVECTOR2)
Declare Sub cV2ScaledNewLength Lib "t2win-16.dll" (u As tagVECTOR2, ByVal newlen As Double)
Declare Function cV2SegmentLength Lib "t2win-16.dll" (p As tagVECTOR2, q As tagVECTOR2) As Double

Call Syntax :

Where :

Comments :

Examples :

See also :

HugeStrAdd

Purpose :

HugeStrAdd adds a VB string into a Huge String.

Declare Syntax :

```
Declare Function cHugeStrAdd Lib "t2win-16.dll" (ByVal hsHandle As Integer, hsText As String) As Integer
```

Call Syntax :

```
hsReturn% = cHugeStrAdd(hsHandle%, hsText$)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsText$	is the VB string to add into the Huge String.
hsReturn%	TRUE : if all is ok
	FALSE : if length of the VB string is 0, or if the VB string can't be fitted into the Huge String.

Comments :

The length of hsText must be between 1 and 64,000 chars.
The position of hsText into the Huge String is depending of the Write Pointer.
If you don't set manually the Write Pointer, the VB String is always appended to previous chars.

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

DOSGetMediaID, DOSSetMediaID

Purpose :

DOSGetMediaID read the media ID (serial number, volume label, ...) from a disk.
DOSSetMediaID change the existing media ID (serial number, volume label, ...) from a disk.

Declare Syntax :

```
Declare Function cDOSGetMediaID Lib "t2win-16.dll" (ByVal nDrive As String, MEDIAID As tagMEDIAID) As Integer
Declare Function cDOSSetMediaID Lib "t2win-16.dll" (ByVal nDrive As String, MEDIAID As tagMEDIAID) As Integer
```

Call Syntax :

```
Test% = cDOSGetMediaID(nDrive$, MEDIAID)
Test% = cDOSSetMediaID(nDrive$, MEDIAID)
```

Where :

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

Comments :

If nDrive is empty, the default drive is used.

Examples :

```
Dim MEDIAID As tagMEDIAID

test% = cDOSGetMediaID("A", MEDIAID)

        Drive A : no media id

test% = cDOSGetMediaID("B", MEDIAID)

        Drive B : no media id

test% = cDOSGetMediaID("C", MEDIAID)

        Drive C :
                SerialNumber is '77777777'
                VolLabel is 'MCRCOMPUTER'
                FileSysType is 'M.C.R. 7'
```

See also :

FileCompress, FileExpand

Purpose :

FileCompress compress a file into a compressed format.
FileExpand expand a compressed file into a normal format.

Declare Syntax :

```
Declare Function cFileCompress Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Long
Declare Function cFileExpand Lib "t2win-16.dll" (ByVal file1 As String, ByVal file2 As String) As Long
```

Call Syntax :

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

Where :

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

Comments :

The compression gives the better result on TEXT file.

Examples :

See also :

ProperName2

Purpose :

ProperName2 convert the first letter of some words separated by a space or punctuation in upper letter case

Declare Syntax :

Declare Function cProperName2 Lib "t2win-16.dll" (Txt As String, ByVal TokenToUse As String, ByVal Options As Integer) As String

Call Syntax :

Test$ = cProperName2(Txt$, TokenToUse$, Options%)

Where :

Txt$	is the text to convert.
TokenToUse$	is the token list that can't be converted.
Options%	PN_UPPERCASE, works with upper case text.
	PN_PUNCTUATION, separator can be a space or a punctuation.
	PN_KEEP_ORIGINAL, keep case letter in the token list.
	PN_ONLY_LEADER_SPACE, don't use the leader trailer space for search in the token list.

Comments :

TokenToUse can be empty.
TokenToUse is a list of all words (separated by '/') which can't be converted (b.e. : "the/and/a/an/or/of")

Examples :

ProperName2 of 'JOHN FITZ,JR' is 'John Fitz,Jr'
ProperName2 of 'john Fitz,jr' is 'John Fitz,Jr'
ProperName2 of 'macdonald' is 'Macdonald'
ProperName2 of 'mac donald' is 'Mac Donald'
ProperName2 of 'a.l. greene jr.' is 'A.L. Greene Jr.'
ProperName2 of 'shale and sandstone and till' is 'Shale and Sandstone and Till'
ProperName2 of 'a sandstone or a shale' is 'a Sandstone or a Shale'

See also :

StringCompress, StringExpand

Purpose :

StringCompress compress a string into a compressed format.
StringExpand expand a compressed string into a normal format.

Declare Syntax :

Declare Function cStringCompress Lib "t2win-16.dll" (Txt As String) As String
Declare Function cStringExpand Lib "t2win-16.dll" (Txt As String) As String

Call Syntax :

Test$ = cFileCompress(Txt$)
Test$ = cFileExpand(Txt$)

Where :

Txt$	is the original string.
Test$	is the compressed string.

Comments :

The compression gives the better result on TEXT string.

Examples :

See also :

FillIncrD, FillIncrI, FillIncrL, FillIncrS

Purpose :

FillIncr fills, with an automatic incremented value, all of the elements of an array (double, integer, long, single).

Declare Syntax :

Declare Function cFillIncrD Lib "t2win-16.dll" (Array() As Double, ByVal nValue As Double, ByVal Increment As Double) As Integer
Declare Function cFillIncrI Lib "t2win-16.dll" (Array() As Integer, ByVal nValue As Integer, ByVal Increment As Integer) As Integer
Declare Function cFillIncrL Lib "t2win-16.dll" (Array() As Long, ByVal nValue As Long, ByVal Increment As Long) As Integer
Declare Function cFillIncrS Lib "t2win-16.dll" (Array() As Single, ByVal nValue As Single, ByVal Increment As Single) As Integer

Call Syntax :

status = cFillIncrD(array(), nValue, Increment)

Where :

array()	is the array.
nValue	is the starting value.
Increment	is the increment.
status	is always TRUE.

Comments :

See Also :

MDAClear

Purpose :

MDAClear clears a multiple big sized array (fill it with chr$(0) or chr$(32) (for string array)).

Declare Syntax :

Declare Function cMDAClear Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY) As Integer

Call Syntax :

ErrCode% = cMDAClear(Array%, MULTIPLEDISKARRAY)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
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 multiple big sized array on disk OR after the using of an existing multiple big sized array on disk.

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

Examples :

```
Dim ErrCode                  As Integer
Dim MDA                      As tagMULTIPLEDISKARRAY

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

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

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

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

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

See also : Multiple Disk Array routines, cMDACreate, cMDAClearSheet

MDAClearCol, MDAsClearCol

Purpose :

MDAClearCol clears a single Col on one Sheet or on all Sheets in a multiple big sized array (fill it with chr$(0) or chr$(32) (for string array)).
MDAsClearCol have the same functionnality but with a multiple big sized array with only one sheet.

Declare Syntax :

Declare Function cMDAClearCol Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long, ByVal Sheet As Long) As Integer
Declare Function cMDAsClearCol Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long) As Integer

Call Syntax :

ErrCode% = cMDAClearCol(Array%, MULTIPLEDISKARRAY, Col&, Sheet&)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Col&	is the desired Col.
Sheet&	is the desired Sheet.
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 multiple big sized array on disk OR after the using of an existing multiple big sized array on disk.

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

If the Col is below 1, the Col 1 is used.
If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.

If the Sheet is -1 then all Sheets are used.
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

Examples :

```
Dim ErrCode                 As Integer
Dim MDA                     As tagMULTIPLEDISKARRAY

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

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

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

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

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

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

See also : Multiple Disk Array routines, cMDACreate, cMDAClear, cMDAClearRow

MDAClearRow, MDAsClearRow

Purpose :

MDAClearRow clears a single Row on one Sheet or on all Sheets in a multiple big sized array (fill it with chr$(0) or chr$(32) (for string array)).
MDAsClearRow have the same functionnality but with a multiple big sized array with only one sheet.

Declare Syntax :

Declare Function cMDAClearRow Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Sheet As Long) As Integer
Declare Function cMDAsClearRow Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long) As Integer

Call Syntax :

ErrCode% = cMDAClearRow(Array%, MULTIPLEDISKARRAY, Row&, Sheet&)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Row&	is the desired Row.
Sheet&	is the desired Sheet.
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 multiple big sized array on disk OR after the using of an existing multiple big sized array on disk.

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

If the Row is below 1, the Row 1 is used.
If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.

If the Sheet is -1 then all Sheets are used.
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

Examples :

```
Dim ErrCode                As Integer
Dim MDA                    As tagMULTIPLEDISKARRAY

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

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

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

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

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

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

See also : Multiple Disk Array routines, cMDACreate, cMDAClear, cMDAClearCol

Multiple Disk Array routines

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

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

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

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

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

cMDAClear	clear a multiple big sized array.
cMDAClearCol	clear a single col on on a sheet in a multiple big sized array.
cMDAClearRow	clear a single row on a sheet in a multiple big sized array.
cMDAClearSheet	clear a single sheet in a multiple big sized array.
cMDAClose	close a big sized array and keep it or close a multiple big sized array and destroy it.
cMDACreate	create a new big sized array on disk or use an existing multiple big sized array on disk.
cMDAGet	read an element from a multiple big sized array on disk.
cMDAGetType	read a type'd variable from a multiple big sized array on disk.
cMDAPut	save an element to a multiple big sized array on disk.
cMDAPutType	save a type'd variable to a multiple big sized array on disk.
cMDAsClearCol	clear a single col on on a sheet in a multiple big sized array with only one sheet.
cMDAsClearRow	clear a single row on a sheet in a multiple big sized array with only one sheet.
cMDAsGet	read an element from a multiple big sized array on disk with only one sheet.
cMDAsGetType	read a type'd variable from a multiple big sized array on disk with only one sheet.
cMDAsPut	save an element to a multiple big sized array on disk with only one sheet.
cMDAsPutType	save a type'd variable to a multiple big sized array on disk with only one sheet.
cMDArGet	read an element from a multiple big sized array on disk with only one sheet and one row.
cMDArGetType	read a type'd variable from a big sized array on disk with only one sheet and one row.
cMDArPut	save an element to a multiple big sized array on disk with only one sheet and one row.
cMDArPutType	save a type'd variable to a multiple big sized array on disk with only one sheet and one row.

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

To handle type'd variable, you must use cMDAGetType, cMDAPutType.

When you create a new multiple array on disk, a header (640 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.

Actually, the maximum number of chars for a string element or for a type'd variable is 4096.

AddTwoTimes

Purpose :

AddTwoTimes adds two time string to form a third time string.

Declare Syntax :

```
Declare Function cAddTwoTimes Lib "t2win-16.dll" (ByVal Time1 As String, ByVal Time2 As String) As String
```

Call Syntax :

```
Test$ = cAddTwoTimes(Time1$, Time2$)
```

Where :

Time1$	is the first time string (format is HH:MM:SS).
Time2$	is the second time string (format is HH:MM:SS).
Test$	is the result (format is HH:MM:SS).

Comments :

The length of each time string must be absolutely 8 characters.
The format of each time string must be absolutely HH:MM:SS.
If the sum of the two time string exceed 24:00:00, the returned string is calculated from 00:00:00.

Examples :

```
Dim Time1 As String
Dim Time2 As String
Dim Time3 As String

Time1 = "23:58:58"
Time2 = "01:02:01"

Time3 = cAddTwoTimes(Time1$, Time2$)                -> "01:00:59"
```

See also :

IncrI, IncrL

Purpose :

IncrI auto-increment an integer value by 1.
IncrL auto-increment a long value by 1.

Declare Syntax :

Declare Sub cIncrI Lib "t2win-16.dll" (Value As Integer)
Declare Sub cIncrL Lib "t2win-16.dll" (Value As Long)

Call Syntax :

cIncrI Value%
cIncrL Value&

Where :

Value%	is the integer value to auto-increment.
Valeu&	is the long value to auto-increment.

Comments :

These routines are slower than the VB equivalent : Value = Value + 1 but are shorter to type.

Examples :

Dim Value As Integer

Value = 5

```
cIncrI Value                -> 6
cIncrI Value                -> 7
```

See also : cDecrI, cDecrL

FileToComboBox, FileToListBox

Purpose :

FileToComboBox read a file and append it to a Combo Box.
FileToListBox read a file and append it to a List Box.

Declare Syntax :

Declare Function cFileToComboBox Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal nFile As String) As Integer
Declare Function cFileToListBox Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal nFile As String) As Integer

Call Syntax :

Test% = cFileToComboBox(Combo1.hWnd, nFile$)
Test% = cFileToListBox(List1.hWnd, nFile$)

Where :

Combo1.hWnd	the .hWnd of a Combo Box.
List1.hWnd	the .hWnd of a List Box.
nFile$	the filename to read.
Test%	= True, if all is ok, <> True, if an error has occured.

Comments :

Examples :

Debug.Print cFileToComboBox(Combo1.hWnd, "c:\tmp\cmb_001.txt")
Debug.Print cFileToListBox(List1.hWnd, "c:\tmp\lst_001.txt")

See also :

FXpicture

Purpose :

FXpicture performs some specials effects on two Picture Box.

Declare Syntax :

Declare Function cFXpicture Lib "t2win-16.dll" (ByVal method As Integer, ByVal hdc1 As Integer, ByVal hbitmap As Integer, ByVal parameter As Integer, ByVal delay As Integer) As Integer

Call Syntax :

Test% = cFXpicture(method%, Picture1.hDC, Picture2.Picture, parameter%, delay%)

Where :

method%	FX_HORIZONTAL FX_VERTICAL FX_DIAGONAL_SQUARE FX_RECTANGLE
Picture1.hDC	is the .hDC of the first Picture Box.
Picture2.Picture	is the .Picture of the second Picture Box.
parameter%	= 0, default value will be 1, >0, the size of a line for special effect.
delay%	= 0, default value will be 10, >0, the delay between two lines for special effect.

Comments :

Normally, the .Visible property of the Picture2 must be set to False
Don't forget that the special effect works directly on the form not into the picture.

Examples :

```
Debug.Print cFXpicture(FX_HORIZONTAL, Picture1.hDC, Picture2.Picture, 0, 0)
Picture1.Picture = Picture2.Picture
```

See also :

DOSGetVolumeLabel, DOSSetVolumeLabel

Purpose :

DOSGetVolumeLabel read the volume label of any disk.
DOSSetVolumeLabel create/change/delete the volume label of any disk.

Declare Syntax :

Declare Function cDOSGetVolumeLabel Lib "t2win-16.dll" (ByVal nDrive As String) As String
Declare Function cDOSSetVolumeLabel Lib "t2win-16.dll" (ByVal nDrive As String, ByVal nVolumeLabel As String) As Integer

Call Syntax :

VolLbl$ = cDOSGetVolumeLabel(nDrive$)
Test% = cDOSSetVolumeLabel(nDrive$, NewVolLbl$)

Where :

nDrive$	is the drive to use.
VolLbl$	is the readed volume label.
NewVolLbl$	is the new volume label.
Test%	= True, if all is ok
	<> True, if an error has occured.

Comments :

The length of a volume label can be 11 chars maximum.
The description of a volume label must respect the DOS filename convention.

Examples :

```
Dim VolLbl         As String
Dim Test As Integer

VolLbl = cDOSGetVolumeLabel("A")

VolLbl             -> "TIME_TO_WIN"

Test = cDOSSetVolumeLabel("A", "NEW_VOLUME")

Test               -> -1 (True)
```

See also :

FloppyInfo

Purpose :

FloppyInfo gives some informations on the selected floppy drive.

Declare Syntax :

Declare Function cFloppyInfo Lib "t2win-16.dll" (ByVal nDrive As String, nHeads As Integer, nCylinders As Integer, nSectors As Integer) As Integer

Call Syntax :

Size% = cFloppyInfo(nDrive$, nHeads%, nCylinders%, nSectors%)

Where :

nDrive$	is the drive letter ('A' or 'B')
nHeads%	is the returned number of Heads.
nCylinders%	is the returned number of Cylinders/Tracks.
nSectors%	is the returned number of Sectors by Cylinders/Tracks.
Size%	is the floppy size (360, 720, 1200, 1440, 2880).

Comments :

Examples :

```
Dim nSize           As Integer
Dim nHeads          As Integer
Dim nCylinders      As Integer
Dim nSectors        As Integer

nSize = cFloppyInfo("A", nHeads, nCylinders, nSectors)

nSize         -> 1440
nHeads        -> 2
nCylinders    -> 80
nSectors-> 18
```

See also :

3DMeter

Purpose :

3DMeter use a Picture Box to perform a 3D-Meter. The indicator can be a Rectangle, a Triangle, a Trapezium, an Ellipse, a Bar Graph.

Declare Syntax :

Declare Sub c3DMeter Lib "t2win-16.dll" (hObj As Object, Meter As tag3DMeter)

Call Syntax :

Call c3DMeter(Picture1, Meter1)

Where :

Picture1	is a picture box
Meter1	is a type'd variable tag3DMeter (see Constants and Types declaration)

Comments :

All precautions have been taken to avoid flickering when normal progress (not random). However, using the Hatch Brush Pattern cause some flickering.

Meter1.CrtValue	the current value in the meter
Meter1.MaxValue	the maximum value of the meter
Meter1.BackColor	the back color
Meter1.ForeColor	the fore color for the current value
Meter1.Polygon	0 or default : rectangle, 1 : triangle, 2 : trapezium, 3 : ellipse, 4 : bar graph
Meter1.BarSize	size of a bar for polygon = 4 (in pixel : min=1, max=20, default : 10)
Meter1.SpaceBars	space between two bars for polygon = 4 (in pixel : min=1, max = 4, default = 2)
Meter1.Direction	0 : horizontal, 1 : vertical
Meter1.ThreeD	0 : none, -1 : indented, 1 : raised
Meter1.Thickness	thickness for the three dimension
Meter1.Percent	internal use : return the percent
Meter1.OldPolygon	internal use
Meter1.OldDirection	internal use
Meter1.OldThreeD	internal use
Meter1.HatchBrush	-1 : solid brush, 0 : horizontal, 1 : vertical, 2 : downward diagonal, 3 : upward diagonal, 4 : cross, 5 : diagonal.cross

Examples :

see T2W_3DM.MAK project

See also :

MDAClearSheet

Purpose :

MDAClearSheet clears a single Sheet in a multiple big sized array (fill it with chr$(0) or chr$(32) (for string array)).

Declare Syntax :

Declare Function cMDAClearSheet Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Sheet As Long) As Integer

Call Syntax :

ErrCode% = cMDAClearSheet(Array%, MULTIPLEDISKARRAY, Sheet&)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Sheet&	is the desired Sheet.
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 multiple big sized array on disk OR after the using of an existing multiple big sized array on disk.

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

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

If the Sheet is -1 then all Sheets are used. This parameter have the same functionnality that cMDAClear
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

Examples :

```
Dim ErrCode             As Integer
Dim MDA                 As tagMULTIPLEDISKARRAY

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

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

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

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

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

See also : Multiple Disk Array routines, cMDACreate; cMDAClear

MDAClose

Purpose :

Close a multiple big sized array and keep it or close a multiple big sized array and destroy it.

Declare Syntax :

Declare Sub cMDAClose Lib "t2win-16.dll" (MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal DeleteFile As Integer)

Call Syntax :

Call cMDAClose(MULTIPLEDISKARRAY, DeleteFile%)

Where :

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

Comments :

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

Examples :

see cMDACreate

See also : Multiple Disk Array routines, cMDACreate

MDACreate

Purpose :

MDACreate creates a multiple new big sized array on disk or use an existing multiple big sized array on disk.

Declare Syntax :

Declare Function cMDACreate Lib "t2win-16.dll" (MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal CreateOrUse As Integer) As Integer

Call Syntax :

ErrCode% = cMDACreate(MDA, CreateOrUse%)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
CreateOrUse%	TRUE : if you want to create a new big sized array on disk, FALSE : if you want to re-use an existing big sized array on disk.
ErrCode%	is the returned error code, see Constants and Types declaration. (DA_x)

Comments :

In theory :

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

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

The length of the filename can be 64 chars maximum.

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

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

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

```
DA.nFilename = "c:\t2w_tmp\dastring.tmp"     'name of the file (you must have enough space on the drive).
DA.nType(1) = 50                             'the type of the variable to use, see Constants and Types declaration. (DA_x)
DA.nIsTyped(1) = False                       'Must be True for a type'd variable for Array 1.
DA.nRows(1) = 500                            'the number of rows to use for Array 1.
DA.nCols(1) = 500                            'the number of cols to use for Array 1.
DA.nSheets(1) = 2                            'the number of sheets to use for Array 1.
.../...
DA.nType(20) = 25                            'the type of the variable to use, see Constants and Types declaration. (DA_x)
DA.nIsTyped(20) = False                      'Must be True for a type'd variable for Array 20.
DA.nRows(20) = 500                           'the number of rows to use for Array 20.
DA.nCols(20) = 500                           'the number of cols to use for Array 20.
DA.nSheets(20) = 2                           'the number of sheets to use for Array 20.
```

YOU CAN'T CHANGE THESE PARAMETERS AFTER THE CREATION OF THE MULTIPLE BIG SIZED ARRAY.

YOU CAN'T CHANGE THE OTHER VALUES IN THE TYPE'D VARIABLE.

Don't forget that you create the multiple array of maximum 20 arrays in one call. The order is not important, but you must take in mind that if you use only 3 arrays on the 20, there are only initialization for these 3 arrays and you can't insert other arrays.

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

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

Examples :

```
Dim ErrCode                 As Integer
Dim MDA                     As tagMULTIPLEDISKARRAY
Dim Var(1 To 8)             As Variant

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

DA.nType(9) = 25                                                      'positive value for a string (array 9)
DA.nIsTyped(9) = False                                                'init the array with spaces (array 9)
DA.nRows(9) = 100                                                     '100 rows (array 9)
DA.nCols(9) = 100                                                     '100 cols (array 9)
DA.nSheets(9) = 5                                                     '5 sheets (array 9)

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

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

Call cMDAPut(9, MDA, 1, 1, 5, "D:2, 1234567890")                      'save the string in Row 1, Col 1, Sheet 5, Array 9
Call cMDAPut(9, MDA, 1, MDA.nCols(9), 5, "D:2, 0987654321")           'save the string in Row 1, Col 100, Sheet 5, Array 9
Call cMDAPut(9, MDA, MDA.nRows(9), 1, 5, "D:2, 12345ABCDE")           'save the string in Row 100, Col 1, Sheet 5, Array 9
Call cMDAPut(9, MDA, MDA.nRows(9), MDA.nCols(9), 5, "D:2, VWXYZ54321") 'save the string in Row 100, Col 100, Sheet 5, Array 9

Var(1) = cMDAGet(1, MDA, 1, 1, 1)                                     'read the string in Row 1, Col 1, Sheet 1, Array 1
Var(2) = cMDAGet(1, MDA, 1, MDA.nCols(1), 1)                          'read the string in Row 1, Col 500, Sheet 1, Array 1
Var(3) = cMDAGet(1, MDA, MDA.nRows(1), 1, 1)                          'read the string in Row 500, Col 1, Sheet 1, Array 1
Var(4) = cMDAGet(1, MDA, MDA.nRows(1), MDA.nCols(1), 1)                 'read the string in Row 500, Col
500, Sheet 1, Array 1

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

Call cMDAClose(MDA, False)                                      'close the file without delete it.
```

See also : Multiple Disk Array routines, cMDAClose

MDAGet, MDArGet, MDAsGet

Purpose :

MDAGet reads an element from a multiple big sized array on disk.
MDArGet have the same functionnality but with a multiple big sized array with only one sheet and only one row.
MDAsGet have the same functionnality but with a multiple big sized array with only one sheet.

Declare Syntax :

Declare Function cMDAGet Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long) As Variant
Declare Function cMDArGet Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long) As Variant
Declare Function cMDAsGet Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long) As Variant

Call Syntax :

Var = cMDAGet(Array%, MULTIPLEDISKARRAY, Row&, Col&, Sheet&)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
Var	is the readed variant value depending of the variable type used in the creation.

Comments :

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

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.
If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.
If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

Examples :

see cMDACreate

See also : Multiple Disk Array routines, cMDAPut

MDAGetType, MDArGetType, MDAsGetType

Purpose :

MDAGetType reads a type'd variable from a multiple big sized array on disk.
MDArGetType have the same functionnality but with a multiple big sized array with only one sheet and only one row.
MDAsGetType have the same functionnality but with a multiple big sized array with only one sheet.

Declare Syntax :

Declare Sub cMDAGetType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)
Declare Sub cMDArGetType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long, nType As Any)
Declare Sub cMDAsGetType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)

Call Syntax :

Call cMDAGetType(Array%, MULTIPLEDISKARRAY, Row&, Col&, Sheet&, nType)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
nType	is the readed type'd variable depending of the variable type used in the creation.

Comments :

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

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.
If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.
If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

Examples :

```
Dim ErrCode             As Integer
Dim MDA                 As tagMULTIPLEDISKARRAY
Dim TE(1 To 4)          As tagTASKENTRY

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

ErrCode = cMDACreate(MDA, False)                 'use a created multiple big sized array on
disk

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

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

See also : Multiple Disk Array routines, cMDAPutType

MDAPut, MDArPut, MDAsPut

Purpose :

MDAPut saves an element to a big sized array on disk.
MDArPut have the same functionnality but with a multiple big sized array with only one sheet and only one row.
MDAsPut have the same functionnality but with a multiple big sized array with only one sheet.

Declare Syntax :

Declare Sub cMDAPut Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, Var As Variant)
Declare Sub cMDArPut Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Col As Long) As Variant
Declare Sub cMDAsPut Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, Var As Variant)

Call Syntax :

Call cMDAPut(Array%, MULTIPLEDISKARRAY, Row&, Col&, Sheet&, Var)

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
Var	is the variant value to save depending of the variable type used in the creation.

Comments :

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

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.
If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.
If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

Examples :

see cMDACreate

See also : Multiple Disk Array routines, cMDAGet

MDAPutType, MDArPutType, MDAsPutType

Purpose :

MDAPutType saves a type'd variable from a big sized array on disk.
MDArPutType have the same functionnality but with a big sized array with only one sheet and only one row.
MDAsPutType have the same functionnality but with a big sized array with only one sheet.

Declare Syntax :

```
Declare Sub cMDAPutType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)
Declare Sub cMDArPutType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Col As Long, nType As Any)
Declare Sub cMDAsPutType Lib "t2win-16.dll" (ByVal Array As Integer, MULTIPLEDISKARRAY As
tagMULTIPLEDISKARRAY, ByVal Row As Long, ByVal Col As Long, nType As Any)
```

Call Syntax :

```
Call cMDAPutType(Array%, MULTIPLEDISKARRAY, Row&, Col&, Sheet&, nType)
```

Where :

MULTIPLEDISKARRAY	is a type'd variable (tagMULTIPLEDISKARRAY).
Array%	is the array in the multiple array (must be between 1 to 20).
Row&	is the row.
Col&	is the col.
Sheet&	is the sheet.
nType	is the type'd variable to save depending of the variable type used in the creation.

Comments :

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

If the Row is greater than MULTIPLEDISKARRAY.nRows(Array%), the Row MULTIPLEDISKARRAY.nRows(Array%) is used.
If the Col is greater than MULTIPLEDISKARRAY.nCols(Array%), the Col MULTIPLEDISKARRAY.nCols(Array%) is used.
If the Sheet is greater than MULTIPLEDISKARRAY.nSheets(Array%), the Sheet MULTIPLEDISKARRAY.nSheets(Array%) is used.

Examples :

```
Dim ErrCode                 As Integer
Dim MDA                     As tagMULTIPLEDISKARRAY
Dim TE                      As tagTASKENTRY

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

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

ErrCode = cTasks(TE, True)
Call cMDAPutType(1, MDA, 1, 1, 1, TE)                    'save the type'd variable in Row 1, Col 1,
```

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

See also : Multiple Disk Array routines, cMDAGetType

DayOfWeek

Purpose :

DayofWeek calculate the day of the week.

Declare Syntax :

Declare Function cDayOfWeek Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer, ByVal nISO As Integer) As Integer

Call Syntax :

Test% = cDayOfWeek(nYear%, nMonth%, nDay%, nISO%)

Where :

nYear%	is the year.
nMonth%	is the month.
nDay%	is the day.
nISO%	= True, for ISO specification,
	= False, for non-ISO specification.
Test%	is the returned day of the week.

Comments :

Following the ISO specification, the returned day of the week will be 0 (Monday) to 6 (Sunday).
Following the non-ISO specification, the returned day of the week will be 0 (Sunday) to 6 (Saturday).

If the parameters are incorrect, the returned value is -1.

Examples :

```
Dim Test           As Integer

'For ISO spefication

Test = cDayOfWeek(1995, 3, 25, True)          -> 5 (Saturday)
Test = cDayOfWeek(1995, 3, 26, True)          -> 6 (Sunday)
Test = cDayOfWeek(1995, 3, 27, True)          -> 0 (Monday)

'For non-ISO specification

Test = cDayOfWeek(1995, 3, 25, False)         -> 6 (Saturday)
Test = cDayOfWeek(1995, 3, 26, False)         -> 0 (Sunday)
Test = cDayOfWeek(1995, 3, 27, False)         -> 1 (Monday)
```

See also :

DateToScalar

Purpose :

DateToScalar compute a scalar from all date parts.

Declare Syntax :

Declare Function cDateToScalar Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Long

Call Syntax :

Test& = cDateToScalar(nYear%, nMonth%, nDay%)

Where :

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

Comments :

If the parameters are not correct, the returned value is -1.

Examples :

```
Dim Test            As Long

Test = cDateToScalar(1995, 3, 25)               -> 728377
```

See also : cScalarToDate

DayOfYear

Purpose :

DayOfYear calculates the day of the year.

Declare Syntax :

Declare Function cDayOfYear Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer) As Integer

Call Syntax :

Test% = cDayOfYear(nYear%, nMonth%, nDay%)

Where :

nYear%	is the year.
nMonth%	is the month.
nDay%	is the day.
Test%	is the returned day of the year.

Comments :

The returned value is 365 or 366 (for a leap year).

If the parameters are incorrect, the returned value is -1.

Examples :

```
Dim Test As Integer

Test = cDayOfYear(1995, 1, 1)          -> 1
Test = cDayOfYear(1995, 3, 25)         -> 84
Test = cDayOfYear(1995, 12, 31)        -> 365
Test = cDayOfYear(1996, 12, 31)        -> 366
```

See also :

ScalarToDate

Purpose :

ScalarToDate decompose a scalar date into these components.

Declare Syntax :

Declare Sub cScalarToDate Lib "t2win-16.dll" (ByVal Scalar As Long, nYear As Integer, nMonth As Integer, nDay As Integer)

Call Syntax :

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

Where :

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

Comments :

Examples :

```
Dim nYear       As Integer
Dim nMonth      As Integer
Dim nDay        As Integer

Call cScalarToDate(728377, nYear%, nMonth%, nDay%)

        nYear%          -> 1995
        nMonth%         -> 3
        nDay%           -> 25
```

See also : cDateToScalar

WeekOfYear

Purpose :

WeekOfYear calculates the week of the year.

Declare Syntax :

Declare Function cWeekOfYear Lib "t2win-16.dll" (ByVal nYear As Integer, ByVal nMonth As Integer, ByVal nDay As Integer, ByVal nISO As Integer) As Integer

Call Syntax :

Test% = cWeekOfYear(nYear%, nMonth%, nDay%)

Where :

nYear%	is the year.
nMonth%	is the month.
nDay%	is the day.
nISO%	= True, for ISO specification, = False, for non-ISO specification.
Test%	is the returned week of the year.

Comments :

ISO defines the first week with 4 or more days in it to be week #1

Following the ISO specification, the returned week of the year will be 0 to 52.
Following the non-ISO specification, the returned week of the year will be 1 to 53.

If the parameters are incorrect, the returned value is -1.

Examples :

```
Dim Test            As Integer

'Following the ISO specification

Test = cWeekOfYear(1995, 12, 31, True)          -> 52
Test = cWeekOfYear(1995, 1, 1, True)            -> 0
Test = cWeekOfYear(1995, 1, 2, True)            -> 1
Test = cWeekOfYear(1995, 3, 25, True)           -> 12
Test = cWeekOfYear(1995, 3, 26, True)           -> 12
Test = cWeekOfYear(1995, 12, 31, True)          -> 52
Test = cWeekOfYear(1996, 1, 1, True)            -> 1

'Following the non-ISO specification

Test = cWeekOfYear(1995, 12, 31, False)         -> 53
Test = cWeekOfYear(1995, 1, 1, False)           -> 1
Test = cWeekOfYear(1995, 1, 2, False)           -> 1
Test = cWeekOfYear(1995, 3, 25, False)          -> 12
Test = cWeekOfYear(1995, 3, 26, True)           -> 13
Test = cWeekOfYear(1995, 12, 31, False)         -> 53
Test = cWeekOfYear(1996, 1, 1, False)           -> 1
```

See also :

GetVersion

Purpose :

GetVersion returns the version number of 'TIME TO WIN (16-Bit)'

Declare Syntax :

Declare Function cGetVersion Lib "t2win-16.dll" () As Single

Call Syntax :

version% = cGetVersion()

Where :

Comments :

This is usefull to avoid version conflict with old version.

Examples :

```
version% = cGetVersion()                3.50
```

See also :

HugeStrAddress

Purpose :

HugeStrAddress returns the memory address of a Huge String.

Declare Syntax :

```
Declare Function cHugeStrAddress Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
```

Call Syntax :

```
hsAddress& = cHugeStrLength(hsHandle%)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsAddress&	is the memory address of the Huge String.

Comments :

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsAddress           As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")

hsAddress = cHugeStrAddress(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had an address of " & hsAddress

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrAppend

Purpose :

HugeStrAppend appends a VB string into a Huge String.

Declare Syntax :

```
Declare Function cHugeStrAppend Lib "t2win-16.dll" (ByVal hsHandle As Integer, hsText As String) As Integer
```

Call Syntax :

```
hsReturn% = cHugeStrAppend(hsHandle%, hsText$)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsText$	is the VB string to append into the Huge String.
hsReturn%	TRUE : if all is ok
	FALSE : if length of the VB string is 0, or if the VB string can't be fitted into the Huge String.

Comments :

The length of hsText must be between 1 and 64,000 chars.
The position of hsText into the Huge String is NOT depending of the Write Pointer. The VB string is appended without regards and whitout change of the Write Pointer.

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
hsReturn = cHugeStrSetWP(hsHandle, 10)
hsReturn = cHugeStrAppend(hsHandle, ", No price change.")

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrBlocks

Purpose :

HugeStrBlocks returns the number of blocks of 64,000 chars into a Huge String.

Declare Syntax :

```
Declare Function cHugeStrBlocks Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
```

Call Syntax :

```
hsBlocks& = cHugeStrBlocks(hsHandle%)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsBlocks&	is the number of blocks of 64,000 chars.

Comments :

If the size of a Huge String is.a multiple of 64.000, the returned blocks will be always the quotient of the division.
If the size of a Huge String is not a multiple of 64.000, the returned blocks will be the quotient of the division plus one.

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsBlocks            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, String$(64000, "A"))
hsReturn = cHugeStrAdd(hsHandle, String$(64000, "B"))
hsReturn = cHugeStrAdd(hsHandle, String$(32000, "C"))

hsBlocks = cHugeStrBlocks(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had " & hsBlocks & " blocks"

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrClear

Purpose :

HugeStrClear clears the contents of a Huge String.

Declare Syntax :

```
Declare Function cHugeStrClear Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Integer
```

Call Syntax :

```
hsReturn% = cHugeStrClear(hsHandle%)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsReturn%	is the returned code, TRUE : the Huge String has been cleared. FALSE : the Huge String can't be cleared.

Comments :

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrClear(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been cleared."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be cleared."
End If

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrCreate

Purpose :

HugeStrCreate creates and reserves enough memory space for the required Huge String.

Declare Syntax :

```
Declare Function cHugeStrCreate Lib "t2win-16.dll" (ByVal hsSize As Long) As Integer
```

Call Syntax :

```
hsHandle% = cHugeStrCreate(hsSize&)
```

Where :

hsSize&	is the size for the Huge String (TIME2WIN add 12 bytes for header).
hsHandle%	is the Handle for all functions for Huge String.

Comments :

The Handle can be '0' if the Huge String can't be created. In this case, you can't use any functions for Huge String.

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of   " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of   " & hsSize & " bytes can't be created."
End If
```

See also :

HugeStrFree

Purpose :

HugeStrFree frees a Huge String created with cHugeStrCreate.

Declare Syntax :

```
Declare Function cHugeStrFree Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Integer
```

Call Syntax :

```
hsReturn% = cHugeStrFree(hsHandle%)
```

Where :

```
hsHandle%       is a handle returned by the cHugeStrCreate function.
hsReturn%       is the returned code,
                TRUE    : the Huge String has been destroyed.
                FALSE   : the Huge String can't be destroyed.
```

Comments :

In the case of the Huge String can't be destroyed, the memory will be restablish when 'TIME TO WIN (16-Bit)' will be unloaded.

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of   " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of   " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrGetNP

Purpose :

HugeStrGetNP returns the Next Pointer of a Huge String.

Declare Syntax :

```
Declare Function cHugeStrGetNP Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
```

Call Syntax :

```
hsPtr& = cHugeStrGetNP(hsHandle%)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsPtr&	is the readed Next Pointer.

Comments :

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
hsReturn = cHugeStrSetWP(hsHandle, 9)

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

MsgBox "The contents of the next 11 chars is " & cHugeStrNext(hsHandle, 11)

MsgBox "The Next Pointer is " & cHugeStrGetNP(hsHandle)

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrGetWP

Purpose :

HugeStrGetWP returns the Write Pointer of a Huge String.

Declare Syntax :

```
Declare Function cHugeStrGetWP Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
```

Call Syntax :

```
hsPtr& = cHugeStrGetWP(hsHandle%)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsPtr&	is the readed Write Pointer.

Comments :

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
hsReturn = cHugeStrSetWP(hsHandle, 9)
hsReturn = cHugeStrAdd(hsHandle, "time to win")

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

MsgBox "The contents of the first block is " & cHugeStrRead(hsHandle, 1)

MsgBox "The Write Pointer is " & cHugeStrGetWP(hsHandle)

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrLength

Purpose :

HugeStrLength returns the length of used chars in a Huge String.

Declare Syntax :

```
Declare Function cHugeStrLength Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
```

Call Syntax :

```
hsLength% = cHugeStrLength(hsHandle%)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsLength%	is the length of used chars.

Comments :

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrMid

Purpose :

HugeStrMid returns the X chars from a position from a Huge String.

Declare Syntax :

```
Declare Function cHugeStrMid Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsStart As Long, ByVal hsLength As Long) As String
```

Call Syntax :

```
hsText$ = cHugeStrMid(hsHandle%, hsStart&, hsLength&)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsStart&	is the starting position (1 to Length of the Huge String).
hsLength&	is the length of the desired string (1 to 64,000).
hsText$	is the readed string.

Comments :

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

MsgBox "The contents of the 11 chars from the position 9 is " & cHugeStrMid(hsHandle, 9, 11)

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrNext

Purpose :

HugeStrNext returns the X next chars from the Next Pointer in a Huge String.

Declare Syntax :

```
Declare Function cHugeStrNext Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsNext As Long) As String
```

Call Syntax :

```
hsText$ = cHugeStrNext(hsHandle%, hsNext&)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsNext&	is the number of next chars to read (1 to 64,000).
hsText$	is the readed string.

Comments :

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
hsReturn = cHugeStrSetWP(hsHandle, 9)

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

MsgBox "The contents of the next 11 chars is " & cHugeStrNext(hsHandle, 11)

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrOnDisk

Purpose :

HugeStrOnDisk reads/writes a Huge String from/to a file.

Declare Syntax :

Declare Function cHugeStrOnDisk Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsFile As String, ByVal hsGetPut As Integer) As Long

Call Syntax :

hsFileLength& = cHugeStrOnDisk(hsHandle%, hsFile$, hsGetPut%)

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsFile$	is the name of the file to read/write the Huge String.
hsGetPut%	PUT_ARRAY_ON_DISK to put the array on disk, GET_ARRAY_ON_DISK to get the array from disk.
hsFileLength&	>=0 is the returned length of the file, < 0 is an error occurs (error n° is the negative value of all DA_x values, see Constants and Types declaration).

Comments :

The file length is the size of the Huge String plus the 12 bytes header.

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

MsgBox "The length of the saved file is " & cHugeStrOnDisk(hsHandle, "c:\hugestr.tmp", PUT_ARRAY_ON_DISK)

hsReturn = cHugeStrClear(hsHandle)

MsgBox "The length of the readed file is " & cHugeStrOnDisk(hsHandle, "c:\hugestr.tmp", GET_ARRAY_ON_DISK)

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
```

```
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrRead

Purpose :

HugeStrRead reads a block of 64,000 chars or a part of block in a Huge String.

Declare Syntax :

```
Declare Function cHugeStrRead Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsBlock As Long) As String
```

Call Syntax :

```
hsText$ = cHugeStrRead(hsHandle%, hsBlock&)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsBlock&	is a block number for reading into Huge String (must be between 1 and cHugeStrBlocks).
hsText$	is the returned string (maximum 64,000 chars).

Comments :

The length of hsText will be between 0 and 64,000 chars.

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

MsgBox "The contents of the first block is " & cHugeStrRead(hsHandle, 1)

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrSetNP

Purpose :

HugeStrSetNP sets the Next Pointer of a Huge String.

Declare Syntax :

```
Declare Function cHugeStrSetNP Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsPtr As Long) As Integer
```

Call Syntax :

```
hsReturn% = cHugeStrSetNP(hsHandle% , hsPtr&)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsPtr&	is the new Next Pointer.
hsReturn%	TRUE : if all is ok
	FALSE : if hsPtr is <=0 or > Length of the Huge String.

Comments :

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsLength            As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
hsReturn = cHugeStrSetNP(hsHandle, 9)

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

MsgBox "The contents of the next 11 chars is " & cHugeStrNext(hsHandle, 11)

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrSetWP

Purpose :

HugeStrSetWP sets the Write Pointer into a Huge String.

Declare Syntax :

```
Declare Function cHugeStrSetWP Lib "t2win-16.dll" (ByVal hsHandle As Integer, ByVal hsPtr As Long) As Integer
```

Call Syntax :

```
hsReturn% = cHugeStrSetWP(hsHandle%, hsPtr&)
```

Where :

hsHandle%	is the Handle for all functions for Huge String.
hsPtr&	is the new Write Pointer.
hsReturn%	TRUE : if all is ok
	FALSE : if hsPtr is <=0 or > Length of the Huge String.

Comments :

Examples :

```
Dim hsHandle          As Integer
Dim hsSize            As Long
Dim hsReturn          As Integer
Dim hsLength          As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReturn = cHugeStrAdd(hsHandle, "This is TIME TO WIN version 4.0")
hsReturn = cHugeStrSetWP(hsHandle, 9)
hsReturn = cHugeStrAdd(hsHandle, "time to win")

hsLength = cHugeStrLength(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a length of " & hsLength

MsgBox "The contents of the first block is " & cHugeStrRead(hsHandle, 1)

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HugeStrSize

Purpose :

HugeStrSize returns the size of a Huge String.

Declare Syntax :

```
Declare Function cHugeStrSize Lib "t2win-16.dll" (ByVal hsHandle As Integer) As Long
```

Call Syntax :

```
hsReadSize& = cHugeStrSize(hsHandle%)
```

Where :

hsHandle%	is a handle returned by the cHugeStrCreate function.
hsReadSize&	is the size of the Huge String.

Comments :

The returned size is the size specified in the cHugeStrCreate function.

Examples :

```
Dim hsHandle            As Integer
Dim hsSize              As Long
Dim hsReturn            As Integer
Dim hsReadSize          As Long

hsSize = 512& * 1024
hsHandle = cHugeStrCreate(hsSize)

If (hsHandle <> 0) Then
        MsgBox "Huge String of " & hsSize & " bytes has been created with handle (" & hsHandle & ")"
Else
        MsgBox "Huge String of " & hsSize & " bytes can't be created."
End If

hsReadSize = cHugeStrSize(hsHandle)

MsgBox "Huge String (" & hsHandle & ") had a size of " & hsReadSize

hsReturn = cHugeStrFree(hsHandle)

If (hsReturn = TRUE) Then
        MsgBox "Huge String (" & hsHandle & ") has been destroyed."
Else
        MsgBox "Huge String (" & hsHandle & ") can't be destroyed."
End If
```

See also :

Huge Strings

The functions/subs usen in the Huge String routines handle Huge String. Huge String is a string from 1 to 16,711,680 chars.

An bigger advantage of Huge String is the speed. The functions for adding or appending chars in a Huge String is faster than VB equivalent (20 times faster).

The maximum number of Huge String is 8192. This number is a theorical maximum and is depending of any application loaded in memory.

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

cHugeStrAdd	Adds a VB string into a Huge String.
cHugeStrAddress	Returns a pointer for the first char of a Huge String.
cHugeStrAppend	Appends a VB string into a Huge String.
cHugeStrBlocks	Returns the number of block of 64,000 chars from a Huge String.
cHugeStrClear	Clears a Huge String.
cHugeStrCreate	Creates a Huge String.
cHugeStrFree	Free a Huge String (destroy it).
cHugeStrGetNP	Gets the Next Pointer of a Huge String.
cHugeStrGetWP	Gets the Write Pointer of a Huge String.
cHugeStrLength	Returns the length of data in a Huge String.
cHugeStrMid	Extracts a VB sub-string from a Huge String.
cHugeStrNext	Reads the next part of a Huge String.
cHugeStrOnDisk	Get/Put a Huge String from/to a file on disk.
cHugeStrRead	Read a block of 64,000 chars or minder from a Huge String.
cHugeStrSetNP	Sets the Next Pointer of a Huge String.
cHugeStrSetWP	Sets the Write Pointer of a Huge String.
cHugeStrSize	Returns the full size of a Huge String.

Don't forget that any Huge String must be destroyed before quitting the application. If you not destroy all Huge String that you've created, the memory used will be only released when T2WIN-16.DLL will be unloaded from memory.

HMAOnDisk

Purpose :

HMAOnDisk reads/writes a Huge Array from/to a file.

Declare Syntax :

Declare Function cHMAOnDisk Lib "t2win-16.dll" (HMA As tagHMA, ByVal hsFile As String, ByVal hsGetPut As Integer) As Long

Call Syntax :

hsFileLength& = cHMAOnDisk(HMA, hsFile$, hsGetPut%)

Where :

HMA	is a type'd variable (tagHMA).
hsFile$	is the name of the file to read/write the Huge Array.
hsGetPut%	PUT_ARRAY_ON_DISK to put the array on disk, GET_ARRAY_ON_DISK to get the array from disk.
hsFileLength&	>=0 is the returned length of the file, < 0 is an error occurs (error n° is the negative value of all HMA_x values, see Constants and Types declaration).

Comments :

The file length is the size of the Huge Array.

Examples :

```
Dim HMA                 As tagHMA
Dim ErrCode             As Integer

HMA.nType = 50                                          'positive value for a string
HMA.nIsTyped = False                                    'init the array with spaces
HMA.nRows = 50                                          '50 rows
HMA.nCols = 50                                          '50 cols
HMA.nSheets = 2                                         '2 sheets

ErrCode = cHMACreate(HMA)

If (ErrCode <> 0) Then
        MsgBox "Huge Array of " & HMA.rMemorySize & " bytes has been created with handle (" & HMA.rHandle &
")"
Else
        MsgBox "Huge Array of " & HMA.rMemorySize & " bytes can't be created."
End If

Call cHMAPut(HMA, 1, 1, 1, "D:1, ABCDEFGHIJ")                   'save the string in Row 1, Col 1, Sheet 1
Call cHMAPut(HMA, 1, HMA.nCols, 1, "D:1, abcdefghij")           'save the string in Row 1, Col 50, Sheet 1
Call cHMAPut(HMA, HMA.nRows, 1, 1, "D:1, OPQRSTUVWXYZ")         'save the string in Row 50, Col 1, Sheet 1
Call cHMAPut(HMA, HMA.nRows, HMA.nCols, 1, "D:1, oprqstuvwxyz") 'save the string in Row 50, Col 50, Sheet 1

MsgBox "The length of the saved file is " & cHMAOnDisk(HMA, "c:\hugestr.tmp", PUT_ARRAY_ON_DISK)

ErrCode = cHMAClear(HMA)

MsgBox "The length of the readed file is " & cHMAOnDisk(HMA, "c:\hugestr.tmp", GET_ARRAY_ON_DISK)

ErrCode = cHMAFree(HMA)
```

```
If (ErrCode = TRUE) Then
	MsgBox "Huge Array (" & hsHandle & ") has been destroyed."
Else
	MsgBox "Huge Array (" & hsHandle & ") can't be destroyed."
End If
```

See also :

HMAPutType, HMArPutType, HMAsPutType

Purpose :

HMAPutType saves a type'd variable from a huge array.
HMArPutType have the same functionnality but with a huge array with only one sheet and only one row.
HMAsPutType have the same functionnality but with a huge array with only one sheet.

Declare Syntax :

Declare Sub cHMAPutType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)
Declare Sub cHMArPutType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long, nType As Any)
Declare Sub cHMAsPutType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, nType As Any)

Call Syntax :

Call cHMAPutType(HMA, Row&, Col&, Sheet&, nType)

Where :

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

Comments :

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

If the Row is greater than HMA.nRows, the Row HMA.nRows is used.
If the Col is greater than HMA.nCols, the Col HMA.nCols is used.
If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

Examples :

```
Dim ErrCode              As Integer
Dim HMA                  As tagHMA
Dim TE                   As tagTASKENTRY

HMA.nType = Len(TE)                               'positive value for a type'd variable
HMA.nIsTyped = True                               'init the array with chr$(0) because type'd
variable
HMA.nRows = 500                                   '500 rows
HMA.nCols = 500                           '500 cols
HMA.nSheets = 2                                   '2 sheets

ErrCode = cHMACreate(HMA)                         'create a new huge array

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

See also : Huge Memory Arrays, cHMAGetType

HMAPut, HMArPut, HMAsPut

Purpose :

HMAPut saves an element to a huge array.
HMArPut have the same functionnality but with a huge array with only one sheet and only one row.
HMAsPut have the same functionnality but with a huge array with only one sheet.

Declare Syntax :

Declare Sub cHMAPut Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, Var As Variant)
Declare Sub cHMArPut Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long) As Variant
Declare Sub cHMAsPut Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, Var As Variant)

Call Syntax :

Call cHMAPut(HMA, Row&, Col&, Sheet&, Var)

Where :

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

Comments :

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

If the Row is greater than HMA.nRows, the Row HMA.nRows is used.
If the Col is greater than HMA.nCols, the Col HMA.nCols is used.
If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

Examples :

see cHMACreate

See also : Huge Memory Arrays, cHMAGet

HMAGetType, HMArGetType, HMAsGetType

Purpose :

HMAGetType reads a type'd variable from a huge array.
HMArGetType have the same functionnality but with a huge array with only one sheet and only one row.
HMAsGetType have the same functionnality but with a huge array with only one sheet.

Declare Syntax :

Declare Sub cHMAGetType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long, nType As Any)
Declare Sub cHMArGetType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long, nType As Any)
Declare Sub cHMAsGetType Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, nType As Any)

Call Syntax :

Call cHMAGetType(HMA, Row&, Col&, Sheet&, nType)

Where :

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

Comments :

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

If the Row is greater than HMA.nRows, the Row HMA.nRows is used.
If the Col is greater than HMA.nCols, the Col HMA.nCols is used.
If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

Examples :

```
Dim ErrCode             As Integer
Dim HMA                 As tagHMA
Dim TE(1 To 4)          As tagTASKENTRY

HMA.nType = Len(TE(1))                                  'positive value for a type'd variable
HMA.nIsTyped = True                                     'init the array with chr$(0) because type'd
variable
HMA.nRows = 500                                         '500 rows
HMA.nCols = 500                                 '500 cols
HMA.nSheets = 2                                         '2 sheets

ErrCode = cHMACreate(HMA)                               'use a created huge array

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

See also : Huge Memory Arrays, cHMAPutType

HMAGet, HMArGet, HMAsGet

Purpose :

HMAGet reads an element from a huge array.
HMArGet have the same functionnality but with a huge array with only one sheet and only one row.
HMAsGet have the same functionnality but with a huge array with only one sheet.

Declare Syntax :

Declare Function cHMAGet Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long, ByVal Sheet As Long) As Variant
Declare Function cHMArGet Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long) As Variant
Declare Function cHMAsGet Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Col As Long) As Variant

Call Syntax :

Var = cHMAGet(HMA, Row&, Col&, Sheet&)

Where :

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

Comments :

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

If the Row is greater than HMA.nRows, the Row HMA.nRows is used.
If the Col is greater than HMA.nCols, the Col HMA.nCols is used.
If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

Examples :

see cHMACreate

See also : Huge Memory Arrays, cHMAPut

HMAFree

Purpose :

Free the memory used by a huge array.

Declare Syntax :

Declare Function cHMAFree Lib "t2win-16.dll" (HMA As tagHMA) As Integer

Call Syntax :

ErrCode = cHMAFree(HMA)

Where :

HMA	is a type'd variable (tagHMA).
ErrCode%	is the returned error code, see Constants and Types declaration. (HMA_x)

Comments :

Examples :

see cHMACreate

See also : Huge Memory Arrays, cHMACreate

HMACreate

Purpose :

HMACreate creates a new huge array.

Declare Syntax :

Declare Function cHMACreate Lib "t2win-16.dll" (HMA As tagHMA) As Integer

Call Syntax :

ErrCode% = cHMACreate(HMA)

Where :

HMA	is a type'd variable (tagHMA).
ErrCode%	is the returned error code, see Constants and Types declaration. (HMA_x)

Comments :

In theory :

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

You are only limited by the size of the memory.

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

When you create a new huge array, the only parameters that you must initialize are :

```
        HMA.nType = 50                'the type of the variable to use, see Constants and Types declaration. (HMA_x)
        HMA.nIsTyped = False          'Must be True for a type'd variable.
        HMA.nRows = 50                'the number of rows to use.
        HMA.nCols = 50                'the number of cols to use.
        HMA.nSheets = 2               'the number of sheets to use.
```

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

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

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

Examples :

```
Dim ErrCode          As Integer
Dim HMA              As tagHMA
Dim Var(1 To 8)      As Variant

HMA.nType = 50                              'positive value for a string
HMA.nIsTyped = False                        'init the array with spaces
HMA.nRows = 50                                                  '50 rows
HMA.nCols = 50                                                  '50 cols
HMA.nSheets = 2                                                 '2 sheets

ErrCode = cHMACreate(HMA)                                       'create a new huge array

Call cHMAPut(HMA, 1, 1, 1, "D:1, ABCDEFGHIJ")                   'save the string in Row 1, Col 1, Sheet 1
Call cHMAPut(HMA, 1, HMA.nCols, 1, "D:1, abcdefghij")           'save the string in Row 1, Col 50, Sheet 1
Call cHMAPut(HMA, HMA.nRows, 1, 1, "D:1, OPQRSTUVWXYZ")         'save the string in Row 50, Col 1, Sheet 1
Call cHMAPut(HMA, HMA.nRows, HMA.nCols, 1, "D:1, oprqstuvwxyz") 'save the string in Row 50, Col 50, Sheet 1

Call cHMAPut(HMA, 1, 1, 2, "D:2, 1234567890")                   'save the string in Row 1, Col 1, Sheet 2
Call cHMAPut(HMA, 1, HMA.nCols, 2, "D:2, 0987654321")           'save the string in Row 1, Col 50, Sheet 2
Call cHMAPut(HMA, HMA.nRows, 1, 2, "D:2, 12345ABCDE")           'save the string in Row 50, Col 1, Sheet 2
Call cHMAPut(HMA, HMA.nRows, HMA.nCols, 2, "D:2, VWXYZ54321")   'save the string in Row 50, Col 50, Sheet 2

Var(1) = cHMAGet(HMA, 1, 1, 1)                                  'read the string in Row 1, Col 1, Sheet 1
Var(2) = cHMAGet(HMA, 1, HMA.nCols, 1")                         'read the string in Row 1, Col 50, Sheet 1
Var(3) = cHMAGet(HMA, HMA.nRows, 1, 1)                          'read the string in Row 50, Col 1, Sheet 1
Var(4) = cHMAGet(HMA, HMA.nRows, HMA.nCols, 1)                  'read the string in Row 50, Col 50, Sheet 1

Var(5) = cHMAGet(HMA, 1, 1, 2)                                  'read the string in Row 1, Col 1, Sheet 2
Var(6) = cHMAGet(HMA, 1, HMA.nCols, 2)                          'read the string in Row 1, Col 50, Sheet 2
Var(7) = cHMAGet(HMA, HMA.nRows, 1, 2)                          'read the string in Row 50, Col 1, Sheet 2
Var(8) = cHMAGet(HMA, HMA.nRows, HMA.nCols, 2)                  'read the string in Row 50, Col 50, Sheet 2

ErrCode = cHMAFree(HMA)                                         'free the memory used.
```

On my system :

```
ErrCode = -1                                                    'no error

HMA.daSize = 64                                                 'internal header size
HMA.nType = 50                                                  'fixed string of 50 chars
HMA.nRows = 50                                                  '50 rows
HMA.nCols = 50                                                  '50 cols
HMA.nSheets = 2                                                 '2 sheets
HMA.rHandle = 0                                                 'internal handle
HMA.rElementSize = 50                                           'internal size of a element
HMA.rFileSize = 250000                                          'internal size of the memory used
HMA.rParts = 3                                                  'internal number of parts (block of 64000
chars)
HMA.rRemain = 58000                                             'internal remain chars
HMA.rSheetSize = 2500                                           'internal size of one sheet

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

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

See also : Huge Memory Arrays, cHMAFree

HMAClearSheet

Purpose :

HMAClearSheet clears a single Sheet in a huge array (fill it with chr$(0) or chr$(32) (for string array)).

Declare Syntax :

Declare Function cHMAClearSheet Lib "t2win-16.dll" (HMA As tagHMA, ByVal Sheet As Long) As Integer

Call Syntax :

ErrCode% = cHMAClearSheet(HMA, Sheet&)

Where :

HMA	is a type'd variable (tagHMA).
Sheet&	is the desired Sheet.
ErrCode%	is the returned error code, see Constants and Types declaration. (HMA_x)

Comments :

This function must be used only after you've created a huge array.

If you've created a huge array, the array is already cleared.

If the huge array have a single Sheet, this routine have the same effect that cHMAClear.

If the Sheet is -1 then all Sheets are used. This parameter have the same functionnality that cHMAClear
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

Examples :

```
Dim ErrCode             As Integer
Dim HMA                 As tagHMA

HMA.nType = 50                                                  'positive value for a string
HMA.nIsTyped = False                                            'init the array with spaces
HMA.nRows = 500                                                 '500 rows
HMA.nCols = 500                                         '500 cols
HMA.nSheets = 2                                                 '2 Sheets

ErrCode = cHMACreate(HMA, True)                                 'create a new huge array

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

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

ErrCode = cHMAClearSheet(HMA, 1)                                'clear the Sheet 1 in the huge array
```

See also : Huge Memory Arrays, cHMACreate, cHMAClear

HMAClearRow, HMAsClearRow

Purpose :

HMAClearRow clears a single Row on one Sheet or on all Sheets in a huge array (fill it with chr$(0) or chr$(32) (for string array)).
HMAsClearRow have the same functionnality but with a huge array with only one sheet.

Declare Syntax :

Declare Function cHMAClearRow Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long, ByVal Sheet As Long) As Integer
Declare Function cHMAsClearRow Lib "t2win-16.dll" (HMA As tagHMA, ByVal Row As Long) As Integer

Call Syntax :

ErrCode% = cHMAClearRow(HMA, Row&, Sheet&)

Where :

HMA	is a type'd variable (tagHMA).
Row&	is the desired Row.
Sheet&	is the desired Sheet.
ErrCode%	is the returned error code, see Constants and Types declaration. (HMA_x)

Comments :

This function must be used only after you've created a huge array.

If you've created a huge array, the array is already cleared.

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

If the Sheet is -1 then all Sheets are used.
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

Examples :

```
Dim ErrCode                 As Integer
Dim HMA                     As tagHMA

HMA.nType = 50                                                   'positive value for a string
HMA.nIsTyped = False                                             'init the array with spaces
HMA.nRows = 500                                                  '500 Rows
HMA.nCols = 500                                          '500 col
HMA.nSheets = 2                                                  '2 Sheets

ErrCode = cHMACreate(HMA)                                        'create a new huge array

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

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

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

See also : Huge Memory Arrays, cHMACreate, cHMAClear, cHMAClearCol

HMAClearCol, HMAsClearCol

Purpose :

HMAClearCol clears a single Col on one Sheet or on all Sheets in a huge array (fill it with chr$(0) or chr$(32) (for string array)).
HMAsClearCol have the same functionnality but with a huge array with only one sheet.

Declare Syntax :

Declare Function cHMAClearCol Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long, ByVal Sheet As Long) As Integer
Declare Function cHMAsClearCol Lib "t2win-16.dll" (HMA As tagHMA, ByVal Col As Long) As Integer

Call Syntax :

ErrCode% = cHMAClearCol(HMA, Col&, Sheet&)

Where :

HMA	is a type'd variable (tagHMA).
Col&	is the desired Col.
Sheet&	is the desired Sheet.
ErrCode%	is the returned error code, see Constants and Types declaration. (HMA_x)

Comments :

This function must be used only after you've created a huge array.

If you've created a huge array, the array is already cleared.

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

If the Sheet is -1 then all Sheets are used.
If the Sheet is below 1 and different of -1, the Sheet 1 is used.
If the Sheet is greater than HMA.nSheets, the Sheet HMA.nSheets is used.

Examples :

```
Dim ErrCode             As Integer
Dim HMA                 As tagHMA

HMA.nType = 50                                              'positive value for a string
HMA.nIsTyped = False                                        'init the array with spaces
HMA.nRows = 500                                             '500 rows
HMA.nCols = 500                                     '500 Cols
HMA.nSheets = 2                                             '2 Sheets

ErrCode = cHMACreate(HMA)                                   'create a new huge array

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

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

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

See also : Huge Memory Arrays, cHMACreate, cHMAClear, cHMAClearRow

HMAClear

Purpose :

HMAClear clears a huge array (fill it with chr$(0) or chr$(32) (for string array)).

Declare Syntax :

Declare Function cHMAClear Lib "t2win-16.dll" (HMA As tagHMA) As Integer

Call Syntax :

ErrCode% = cHMAClear(HMA)

Where :

HMA	is a type'd variable (tagHMA).
ErrCode%	is the returned error code, see Constants and Types declaration. (HMA_x)

Comments :

This function must be used only after you've created a huge array.

If you've created a huge array, the array is already cleared.

Examples :

```
Dim ErrCode               As Integer
Dim HMA                   As tagHMA

HMA.nType = 50                                            'positive value for a string
HMA.nIsTyped = False                                      'init the array with spaces
HMA.nRows = 500                                           '500 rows
HMA.nCols = 500                                  '500 cols
HMA.nSheets = 2                                           '2 sheets

ErrCode = cHMACreate(HMA)                                 'create a new huge array

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

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

ErrCode = cHMAClear(HMA)                                  'clear all elements in the big sized array on
disk
```

See also : Huge Memory Arrays, cHMACreate, cHMAClearSheet

Huge Memory Arrays

The functions/subs usen in the Huge Memory Arrays routines handle Huge Arrays. Huge Arrays is based on the same principle that DISK ARRAY.

An bigger advantage of Huge Arrays is the speed.

The maximum number of Huge Arrays is 8192. This number is a theorical maximum and is depending of any application loaded in memory.

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

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

Don't forget that any Huge Memory Arrays must be destroyed before quitting the application. If you not destroy all Huge Memory Arrays that you've created, the memory used will be only released when T2WIN-16.DLL will be unloaded from memory.

ScalarToTime

Purpose :

ScalarToTime decompose a scalar time into these components.

Declare Syntax :

Declare Sub cScalarToTime Lib "t2win-16.dll" (ByVal Scalar As Long, nHour As Integer, nMin As Integer, nSec As Integer)

Call Syntax :

Call cScalarToTime(Scalar&, nHour%, nMin%, nSec%)

Where :

Scalar&	is a scalar time.
nHour%	is the returned hour.
nMin%	is the returned minute.
nSec%	is the returned second.

Comments :

Examples :

```
Dim nHour       As Integer
Dim nMin        As Integer
Dim nSec        As Integer

Call cScalarToTime(60630, nHour%, nMin%, nSec%)

        nHour%          -> 16
        nMin%           -> 50
        nSec%           -> 30
```

See also : cTimeToScalar

ShowWindow

Purpose :

Show a window after an exploded/imploded focus rectangle has been displayed.

Declare Syntax :

Declare Sub cShowWindow Lib "t2win-16.dll" (ByVal hWnd As Integer, ByVal method As Integer, ByVal interval As Integer)

Call Syntax :

Call cShowWindow(hWnd%, method%, interval%)

Where :

hWnd%	is the handle of a form.
method%	0 : explode the form starting at center of the form.
	1 : implode the form starting at external.
interval%	0 : faster
	699 : lower

Comments :

The interval is a modulo of 700 and is calculated in millisecond.

Examples :

Call cShowWindow(Form1.hWnd, 0, 250)

See also :

TimeToScalar

Purpose :

TimeToScalar compute a scalar from all time parts.

Declare Syntax :

Declare Function cTimeToScalar Lib "t2win-16.dll" (ByVal nHour As Integer, ByVal nMin As Integer, ByVal nSec As Integer) As Long

Call Syntax :

Test& = cTimeToScalar(nHour%, nMin%, nSec%)

Where :

nHour%	is the Hour.
nMin%	is the Min.
nSec%	is the Sec.
Test&	is the returned computed scalar.

Comments :

The parameter Hour can be between 0 to 32767.

If the parameters are not correct, the returned value is -1.

Examples :

```
Dim Test            As Long

Test = cTimeToScalar(16, 50, 30)                    -> 60630
```

See also : cScalarToTime

CenterWindow

Purpose :

Center a window in the screen.

Declare Syntax :

Declare Sub cCenterWindow Lib "t2win-16.dll" (ByVal hWnd As Integer)

Call Syntax :

Call cCenterWindow(hWnd%)

Where :

hWnd%	is the handle of a form.

Comments :

Examples :

Call cCenterWindow(Form1.hWnd)

See also :

GetCtlRect, GetCtlRectTwips

Purpose :

GetCtlRect returns the Left, Top, Right, Bottom value of a control in Pixels.
GetCtlRectTwips returns the Left, Top, Right, Bottom value of a control in Twips.

Declare Syntax :

```
Declare Sub cGetCtlRect Lib "t2win-16.dll" (Obj As Object, RECT As Any)
Declare Sub cGetCtlRectTwips Lib "t2win-16.dll" (Obj As Object, RECT As Any)
```

Call Syntax :

```
Call cGetCtlRect(Ctl, Rect)
Call cGetCtlRectTwips(Ctl, Rect)
```

Where :

Ctl	is a VB standard control or VBX.
Rect	is a type'd variable tagRECT (see Constants and Types declaration)

Comments :

The returned 4 values are based on the container of the control.

Examples :

```
Dim Rect        As tagRECT

Call cGetCtlRect(Label1, Rect)
```

See also :

FilesInfoInDir

Purpose :

FilesInfoInDir retrieves each file in the specified directory and returns name, size, scalar date, scalar time, attribute.

Declare Syntax :

Declare Function cFilesInfoInDir Lib "t2win-16.dll" (ByVal nDir As String, FILEINFO As tagFILEINFO, ByVal FirstNext As Integer) As String

Call Syntax :

test$ = cFilesInfoInDir(nDir, FI, firstnext)

Where :

nDir	the directory to proceed with the file mask (*.* for all)
FI	the type'd variable tagFILEINFO
firstnext	TRUE for the first file FALSE for each next file
test$	the returned file

Comments :

If the nDir is invalid or if an error occurs when accessing a file, the returned filename is an empty string and all sub-variables in the type'd variable are -1.

Examples :

```
Dim i           As Integer
Dim Tmp         As String
Dim FI                  As tagFILEINFO

i = 0
Tmp = cFilesInfoInDir("c:\*.*", FI, True)

Debug.Print "The first 7 files in C:\ are : "

Do While (Len(Tmp) > 0)
    Debug.Print Tmp & ", " & FI.fSize & ", " & FI.fDate & ", " & FI.fTime & ", " & FI.fAttribute
    Tmp = cFilesInfoInDir("c:\*.*", FI, False)
    i = i + 1
    If (i >= 7) Then Exit Do
Loop
```

On my system:

```
The first 7 files in C:\ are :

SUHDLOG.DAT, 5166, 728480, 76033, 3
BOOTLOG.TXT, 22886, 728480, 78500, 2
MSDOS.---, 22, 728480, 75079, 2
DBLSPACE.001, 79036439, 728519, 48662, 7
SYSTEM.1ST, 230144, 728480, 76027, 7
WINA20.386, 9349, 727632, 21600, 0
AUTOEXEC.BAK, 968, 728456, 78015, 0
```

See also : cFilesInDirectory, cAllSubDirectories, cSubDirectory

RcsCountFileDir

Purpose :

RcsCountFileDir counts the total directories or files in a specified directory (with recursivity or not).

Declare Syntax :

Declare Function cRcsCountFileDir Lib "t2win-16.dll" (ByVal FileOrDir As Integer, ByVal FirstFileOrDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Integer

Call Syntax :

test% = cRcsCountFileDir(FileOrDir%, FirstFileOrDir$, MaskDir$, Recurse%)

Where :

FileOrDir%	FALSE for directories TRUE for files
FirstFileOrDir$	the starting directory (root or sub-dir) or file
MaskDir$	the mask for performing the search (If this is an empty string, "*.*" is used)
Recurse%	FALSE for no recursivity TRUE for recursivity
test%	the number of sub-dirs or files founden in the specified directory

Comments :

This function is a superset function of cCountDirectories and cCountFiles

For directory :

The internal '.' in each directory is not counted.
The root directory is not counted.

For file :

The mask is the standard search mask (*, ?, letters, ciphers).

See also : cCountDirectories, cCountFiles

FilesInDirOnDisk

Purpose :

FilesInDirOnDisk writes all files from a specified directory into a file on disk.

Declare Syntax :

Declare Function cFilesInDirOnDisk Lib "t2win-16.dll" (ByVal nFile As String, ByVal nFilename As String, ByVal nAttribute As Integer) As Integer

Call Syntax :

test% = cFilesInDirOnDisk(nFile$, nFilename$, nAttribute)

Where :

nFile$	the file on disk
nFilename	the directory to proceed with the file mask (if this is an empty string, '*.*' is used)
nAttribute	the attribute to search (see Constants and Types declaration)
test%	the number of files founded

Comments :

When nAttribute is a positive value, the search is based on an OR test. If one or more attributes of file is founded, the file is taken.
When nAttribute is a negative value, the search is based on an AND test. If all attributes of files are founded, the file is taken.

Examples :

```
Dim i As Integer

i = cFilesInDirOnDisk("c:\test.tmp", "*.*", A_ALL)
```

See also : cFilesInDirToArray, cFilesInDirectory, cFilesInfoInDir, cAllSubDirectories, cSubDirectory

FilesInDirToArray

Purpose :

FilesInDirToArray reads all files from a specified directory into an array.

Declare Syntax :

Declare Function cFilesInDirToArray Lib "t2win-16.dll" (ByVal nFilename As String, ByVal nAttribute As Integer, array() As Any) As Integer

Call Syntax :

test% = cFilesInDirToArray(nFilename$, nAttribute%, Array())

Where :

nFilename	the directory to proceed with the file mask (if this is an empty string, '*.*' is used)
nAttribute	the attribute to search (see Constants and Types declaration)
Array()	is the variable array string with one dimension.
test%	>=0 is the number of file in the specified directory, < 0 is an error occurs (error n° is the negative value of all DA_x values, see Constants and Types declaration).

Comments :

When nAttribute is a positive value, the search is based on an OR test.   If one or more attributes of file is founded, the file is taken.
When nAttribute is a negative value, the search is based on an AND test.   If all attributes of files are founded, the file is taken.

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

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

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

Examples :

```
ReDim AD(-999 To 1000)          As tagVARSTRING

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

Debug.Print cFilesInDirToArray("c:\*.*", A_ALL, AD())

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

See also : cFilesInDirOnDisk, cFilesInDirectory, cFilesInfoInDir, cAllSubDirectories, cSubDirectory

RcsFilesSize

Purpose :

RcsFilesSize returns the logical size of files by file mask in a specified directory (with recursivity or not).
RcsFilesSizeOnDisk returns the physical size of files by file mask in a specified directory (with recursivity or not).
RcsFilesSlack returns in one call, the slack from files by file mask in a specified directory (with recursivity or not), the logical size and the physical size.

Declare Syntax :

Declare Function cRcsFilesSize Lib "t2win-16.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Long
Declare Function cRcsFilesSizeOnDisk Lib "t2win-16.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer) As Long
Declare Function cRcsFilesSlack Lib "t2win-16.dll" (ByVal FirstDir As String, ByVal MaskDir As String, ByVal Recurse As Integer, Size1 As Long, Size2 As Long) As Integer

Call Syntax :

test& = cRcsFilesSize(FirstDir$, MaskDir$, Recurse%)
test& = cRcsFilesSizeOnDisk(FirstDir$, MaskDir$, Recurse%)
test% = cRcsFilesSlack(FirstDir$, MaskDir$, Recurse%, Size1, Size2)

Where :

FirstDir$	the starting directory (root or sub-dir).
MaskDir$	the mask for performing the search (If this is an empty string, "*.*" is used)
Recurse%	FALSE for no recursivity
	TRUE for recursivity
test&	is the size of all files founden with the file mask.
test%	is the slack for all files fouden with the file mask.
Size1	is the logical size of all files fouden with the file mask.
Size2	is the physical size of all files fouden with the file mask.

Comments :

If the mask is invalid or if the file not exists or if an error occurs when accessing the file, the return value is 0
The slack of a file or files by file mask is the % of all spaces not used on all last clusters.

Examples :

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

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

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

See also : cFileSize, cGetDiskClusterSize

ReadMnuLanguage

Purpose :

SaveMnuLanguage creates or updates a file which contains the text (menu) for supporting a language.
ReadMnuLanguage reads a file which contains the text (menu) for supporting a language.

Declare Syntax :

```
Declare Function cReadMnuLanguage Lib "t2win-16.dll" (hCtlFirstMenu As Control, ByVal FileLanguage As String) As Integer
Declare Function cSaveMnuLanguage Lib "t2win-16.dll" (hCtlFirstMenu As Control, ByVal FileLanguage As String) As Integer
```

Call Syntax :

```
test% = cSaveMnuLanguage(hCtlFirstMenu, FileLanguage)
test% = cReadMnuLanguage(hCtlFirstMenu, FileLanguage)
```

Where :

hCtlFirstMenu	is the first menu control on the form.
FileLanguage$	is the file name to perform the language management.
test%	TRUE if all is ok FALSE is an error has occured

Comments :

These functions are very, VERY simple to use and your application can support multi-language very fast.

If a problem occurs when accessing the menus or if the form has no menu or if the filename is an EMPTY string, the returned value is FALSE. These fonctions doesn't test the validity of the file name.

FileLanguage is the name of the file to use to store or retrieve the Property. After the first saving, you translate the file (with NOTEPAD, b.e.) into an another language and save it to an other name. You can use the extension als follows .T?? with ?? is FR (for FRench), UK (for United Kingdom, GE (for GErmany), IT (for ITaly), SP (for SPain), ...

Examples :

```
test% = cSaveMnuLanguage(mnu_File, "D:\TIME2WIN\DEMO\TIME2WIN.TUK")
        translate it to French and save it in the file "D:\TIME2WIN\DEMO\TIME2WIN.TFR"
test% = cReadMnuLanguage(mnu_File, "D:\TIME2WIN\DEMO\TIME2WIN.TFR")
```

See also : cReadCtlLanguage, cSaveCtlLanguage

SpellMoney

Purpose :

SpellMoney spells money value with hundredth.

Declare Syntax :

Declare Function cSpellMoney Lib "t2win-16.dll" (ByVal Value As Double, ByVal Units As String, ByVal Cents As String) As String

Call Syntax :

Test$ = cSpellMoney(Value#, Units$, Cents$)

Where :

Value#	is the money value to spell.
Units$	is the text string for units part.
Cents$	is the text string for cents part.
Test$	is the returned spelled money value.

Comments :

Examples :

Test$ = cSpellMoney("98765.43", "dollars", "cents")

SpellMoney of '4.12' is 'Four dollars and Twelve cents'
SpellMoney of '16' is 'Sixteen dollars'
SpellMoney of '25' is 'Twenty-Five dollars'
SpellMoney of '34' is 'Thirty-Four dollars'
SpellMoney of '43' is 'Forty-Three dollars'
SpellMoney of '61' is 'Sixty-One dollars'
SpellMoney of '98765.43' is 'Ninety-Eight Thousand Seven Hundred Sixty-Five dollars and Forty-Three cents'
SpellMoney of '123456789.75' is 'One Hundred Twenty-Three Million Four Hundred Fifty-Six Thousand Seven Hundred Eighty-Nine dollars and Seventy-Five cents'

See also :

Fraction

Purpose :

Fraction returns a value into the form of a fraction.

Declare Syntax :

```
Declare Function cFraction Lib "t2win-16.dll" (ByVal nValue As Double, nNumerator As Double, nDenominator As
Double) As Double
```

Call Syntax :

```
Test# = cFraction(Value#, Numerator#, Denominator#)
```

Where :

Value#	is the value to proceed.
Numerator#	is the returned numerator.
Denominator#	is the returned denominator.
Test#	is the returned value (Numerator# / Denominator#).

Comments :

Examples :

```
Dim Value               As Double
Dim Numerator           As Double
Dim Denominator As Double
Dim CalculatedValue     As Double

Value = 0.75
CalculatedValue = cFraction(Value, Numerator, Denominator)
        -> Numerator = 3
        -> Denominator = 4
        -> CalculatedValue = 0.75

Value = 3.14159265
CalculatedValue = cFraction(Value, Numerator, Denominator)
        -> Numerator = 3017882801
        -> Denominator = 960621932
        -> CalculatedValue = 3,14159265
```

See also :

RndInit, RndD, RndI, RndL, RndS

Purpose :

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

Declare Syntax :

```
Declare Sub cRndInit Lib "t2win-16.dll" (ByVal nRnd As Long)
Declare Function cRndD Lib "t2win-16.dll" () As Double
Declare Function cRndI Lib "t2win-16.dll" () As Integer
Declare Function cRndL Lib "t2win-16.dll" () As Long
Declare Function cRndS Lib "t2win-16.dll" () As Single
Declare Function cRnd Lib "t2win-16.dll" () As Double
```

Call Syntax :

```
Call cRndInit(nRnd&)
Test% = cRndI()
Test& = cRndL()
Test! = cRndS()
Test# = cRndD()
Test# = cRnd()
```

Where :

nRnd	< 0 : initialization with the current date and time. > 0 : initialization with the passed value.
Test?	the returned random number

Comments :

Examples :

```
Call cRndInit(-1)

debug.print cRndI()          -> 316
debug.print cRndL()          -> 45980750
debug.print cRndS()          -> 1,330308E+38
debug.print cRndD()          -> 1,87044922807943E+304
debug.print cRnd()           -> 1,87044922807943E+304
```

See also :

StringSAR

Purpose :

StringSAR searchs and replaces a string by an another in the specified string.

Declare Syntax :

Declare Function cStringSAR Lib "t2win-16.dll" (ByVal Txt As String, ByVal Search As String, ByVal Replace As String, ByVal Sensitivity As Integer) As String

Call Syntax :

Test$ = cStringSAR(Txt$, Search$, Replace$, Sensitivity%)

Where :

Txt$	the string to proceed.
Search$	the string to be searched.
Replace$	the replacement string.
Sensitivity%	TRUE if the search must be case-sensitive, FALSE if the search is case-insensitive.
Test$	the returned string with replacement.

Comments :

If the search string is an EMPTY string, the returned string is the passed string.

If an error occurs when creating buffer, the returned string is the passed string.

The length of the replace string can be > or < of the search string.
The replace string can be an EMPTY string. In this case, the search string is removed from the file.

Examples :

```
Dim Txt                  As String
Dim Search               As String
Dim Replace              As String
Dim Test         As String

Txt = "TIME TO WIN, TIME TO WIN IS A DLL"

Search = "TIME TO WIN"
Replace = "TIME2WIN"
Test = cStringSAR(Txt, Search, Replace, False)

debug.print Test         -> "TIME2WIN, TIME2WIN IS A DLL"

Search = "TIME to WIN"
Replace = "TIME2WIN"
Test = cStringSAR(Txt, Search, Replace, True)

debug.print Test         -> "TIME TO WIN, TIME TO WIN IS A DLL"

Search = " TO "
Replace = "2"
Test = cStringSAR(Txt, Search, Replace, True)

debug.print Test         -> "TIME2WIN, TIME2WIN IS A DLL"
```

See also :

TruncatePath

Purpose :

TruncatePath truncates a long path with filename.

Declare Syntax :

Declare Function cTruncatePath Lib "t2win-16.dll" (ByVal nFilename As String, ByVal NewLength As Integer) As String

Call Syntax :

Test$ = cTruncatePath(nFilename, NewLength%)

Where :

nFilename	is the path.
NewLength%	is the new length of the path.
Test$	is the returned truncated path.

Comments :

If 'nFilename' is an invalid file, the returned path is always an EMPTY string.
If 'NewLength' is below to 25, the returned path is always an EMPTY string.
If the length of 'nFilename' is below 25, the 'nFilename' is returned.

Examples :

```
Dim Tmp             As String
Dim Test        As String
Dim NewLength       As Integer

NewLength = 25

Tmp = "time2win.bas"
debug.print cTruncatePath(Tmp, NewLength)       ' -> time2win.bas

Tmp = "windows\system\time2win.bas"
debug.print cTruncatePath(Tmp, NewLength)       ' -> windows......time2win.bas

Tmp = "c:\windows\system\time2win.bas"
debug.print cTruncatePath(Tmp, NewLength)       ' -> c:\windows...time2win.bas

Tmp = "c:\windows\system\vb\time2win\time2win.bas"
debug.print cTruncatePath(Tmp, NewLength)       ' -> c:\windows...time2win.bas

Tmp = "c:\windows\system\vb\source\time2win\time2win.bas"
debug.print cTruncatePath(Tmp, NewLength)       ' -> c:\windows...time2win.bas
```

See also :

Notice for VB 4.0

Normally, all routines except routines with variant usage must work with Visual Basic 4.0

The variant problem is due to the fact that Microsoft uses the OLEAPI in replacement of VBAPI.

You must use the T2WIN-16.DLL for Visual Basic 4.0 (16-Bit).

CountI, CountL, CountS, CountD

Purpose :

CountI counts a specific value in an Integer array.
CountL counts a specific value in a Long array.
CountS counts a specific value in a Single array.
CountD counts a specific value in a Double array.

Declare Syntax :

```
Declare Function cCountI Lib "t2win-16.dll" (array() As Integer, ByVal Value As Integer) As Long
Declare Function cCountL Lib "t2win-16.dll" (array() As Long, ByVal Value As Long) As Long
Declare Function cCountS Lib "t2win-16.dll" (array() As Single, ByVal Value As Single) As Long
Declare Function cCountD Lib "t2win-16.dll" (array() As Double, ByVal Value As Double) As Long
```

Call Syntax :

```
cnt& = cCountI(array(), Value%)
cnt& = cCountL(array(), Value&)
cnt& = cCountS(array(), Value#)
cnt& = cCountD(array(), Value!)
```

Where :

array()	is the array (Integer, Long, Single, Double).
Value?	is the value to count (Integer, Long, Single, Double).
cnt&	is the returned counted value.

Comments :

See Also : Array routines

SearchI, SearchL, SearchS, SearchD

Purpose :

SearchI Searchs a specific value in an Integer array.
SearchL Searchs a specific value in a Long array.
SearchS Searchs a specific value in a Single array.
SearchD Searchs a specific value in a Double array.

Declare Syntax :

```
Declare Function cSearchI Lib "t2win-16.dll" (array() As Integer, ByVal Value As Integer) As Long
Declare Function cSearchL Lib "t2win-16.dll" (array() As Long, ByVal Value As Long) As Long
Declare Function cSearchS Lib "t2win-16.dll" (array() As Single, ByVal Value As Single) As Long
Declare Function cSearchD Lib "t2win-16.dll" (array() As Double, ByVal Value As Double) As Long
```

Call Syntax :

```
cnt& = cSearchI(array(), Value%)
cnt& = cSearchL(array(), Value&)
cnt& = cSearchS(array(), Value#)
cnt& = cSearchD(array(), Value!)
```

Where :

array()	is the array (Integer, Long, Single, Double).
Value?	is the value to search (Integer, Long, Single, Double).
cnt&	> 0 : the position of the searched value;
	= -1 : the searched value is not found.

Comments :

See Also : Array routines