IDBTOOLS

=

TABLE-handling routines for Visual Basic

ITabBlankLine	Remove contents of a given line in a table
ITabBlankLines	Remove contents of the given lines in a table
ITabDelete	Delete table and release memory
ITabDir	Create a table containing file and/or directory information.
ITabEnvList	Read environment settings into a new table
ITabFind	Search data , given mask, by column
ITabFindGE	Search for data, given mask, by column in pre-sorted table.
ITabGet	Get data as string type from table
ITabGetColWidth	Get the width of a given column in a table
ITabGetInt	Get data as integer type from table
ITabGetLine	Get data as string type from an array
ITabGetLong	Get data as long type from table
ITabGetNumColumns	Get the defined number of columns a table consists of
ITabGetNumLines	Get the defined number of lines/rows a table consists of
ITabGetReal	Get data as real type from table
ITabGetSize	Get the total amount of consumed memory for a given table
ITabInsertLine	Insert a line at a given line number in a table
ITabInsertLines	Insert lines from a given line number in a table
ITabNew	Create a new table.
ITabNewArray	Create a new table with only one column.
ITabPut	Write data (string) to a cell in a table
ITabPutInt	Write data (integer) to a cell in a table
ITabPutLine	Write data (string) to a line in a table/array
ITabPutLong	Write data (long) to a cell in a table
ITabPutReal	Write data (real) to a cell in a table
ITabRead	Read a table to memory from a file in a specific format
ITabReadFixedRecLenFile	Read a file with fixed record length to a table
ITabRemoveLine	Remove a line at a given line number in a table
ITabRemoveLines	Remove lines from a given line number in a table
ITabSmartSort	Sort a table by column using the SmartSort algorithm
ITabWrite	Write a table to a named disk file of a specific format

IDBVTSS.DLL:

ITabCopyFromVTSS	Read the contents of a Visual Tools SpreadSheet to a new table
ITabCopyToVTSS	Dump the contents of a table to a Visual Tools SpreadSheet

Include IDBTOOLS.BAS and IDBTABLE.BAS in your projects.

See also IDBTOOLS.WRI for description of more IdbTools routines

=

=

What is a TABLE ?

An IdbTools table is a dynamic array or matrix of variable length text strings.

It is possible to define arrays of strings in native Visual Basic too, but this is not very flexible, and the practical limits makes it completely useless compared to tables in IdbTools:

```
Static SmallMatrix(500, 10) As String ' Standard Visual Basic
For i% = 1 To 500
    For j% = 1 To 10
        SmallMatrix(i%, j%) = "TESTING"
    Next j%
Next i%
```

"**OUT OF STRING SPACE**" is the message from Visual Basic !
I´ve got 16 MB of RAM, but VB does not want to use it.

String Arrays in VB are not going to be mentioned any more, from this point. In the following, the term Array will only be used for describing a table with only one column.

```
' The good news are: the following works fine

BigTable& = ITabNew(5000, 10) ' IdbTools: 10 times bigger - no problem !
For i% = 1 To 5000
    For j% = 1 To 10
        ITabPut BigTable&, i%, j%, "TESTING"
    Next j%
Next i%

' You may put several megabytes of data into the IdbTools tables.
```

The powerful routines for table searching, sorting, file I/O and all the other routines found in IdbTools will open a new world for all Visual Basic programmers - giving the programming power you have dreamed of !

The rest of this document is describing the routines for table handling.

Sub ITabBlankLine

Erase contents of the given line.

Usage:
```
ITabBlankLine Handle&, atLine%
```

The line will still be there, but all columns will be empty. See also: ITabRemoveLine.

Sub ITabBlankLines

Erase contents of the specified lines.

Usage:
```
ITabBlankLines Handle&, atLine%, numLines%
```

The lines will still be there, but all columns will be empty. See also: ITabRemoveLines.

Sub ITabDelete

Delete a table from memory with effective memory release.

Usage:
```
ITabDelete Handle&
```

NB!
After a table is deleted the handle is invalid. Using a handle for a deleted table will cause an error.
Of course, the variable holding the handle value may be reused for new tables.

Example:
```
ITabDelete MyTable&
```

Function ITabDir

Create a table with a list of filenames and/or directory names, and optionally, more detailed information connected to these. The function returns a handle to the new table.

Usage:

```
Handle& = ITabDir(FileMask$, Type%)
```

FileMask$ can be a file / directory name, or a standard wildcard mask using the characters: "?" and "*"

Type%	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
1	File.Ext					
2	Filename	Ext				
3	Filename	Ext	Size			
4	Filename	Ext	Size	Date		
5	Filename	Ext	Size	Date	Time	
6	Filename	Ext	Size	Date	Time	Attr

7 As type 6, but includes **hidden** and **system** files in addition to normal files.
8 As type 7, but ***also*** includes **subdirectories** .
9 As type 6, but includes ***only*** **subdirectories**

Date format: "YYYYMMDD"
Time format: "HH:MM:SS"
Attr format: "ADHRS", the single letter will be found in the given position, (D=2, R=4 ...), when the attribute is active.
A: Archive (set when file is changed - used by back-up systems)
D: Directory name
H: Hidden file
R: Read-Only file
S: System file

The number of columns in the newly created table will be equal to Type% up to 6, and 6 for the rest.

Directory names will always be terminated by the character "\". Obs, beware: The directory name can include the ".EXT", in that case the character "\" will be found in column 2 (for types 8 and 9).

Remember to delete the table, using ITabDelete, when it is no longer needed (free system resources).

Example:

```
' Make a function for returning the size of a given file.
' Will return 0 if the file does not exist (not runtime error as FileLen):

Function FileLength& (ByVal FileName$)
   tempTab& = ITabDir(FileName$, 3)
   FileLength& = ITabGetLong(tempTab&, 1, 3)
   ITabDelete tempTab&
End Function
```

Function ITabEnvList

Create a new table containing all environment strings defined. The table will have two columns where the first column contains the variable name, and the second the environment setting.

Usage:

```
Handle& = ITabEnvList()
```

Remember to delete the table, using ITabDelete, when it is no longer needed (free system resources):

Example:

```
envTab& = ITabEnvList()

' the table may look like this:
'
'     1st column  2nd column
'     CONFIG      QEMM
'     COMSPEC     C:\DOS\COMMAND.COM
'     SHARE       ON
'     TMP         E:\TMP
'     APPEND      D:\PROG
'     LIST        E:\TMP
'     BLASTER     A220 I7 D1 H5 P330 T6
'     LIB         C:\MSVC\LIB;C:\MSVC\MFC\LIB;..\LIB
'     WINDIR      D:\WINDOWS
'     ...

row% = ITabFind(envTab&, "COMSPEC", 1, 1, IT_EXACT)   ' search
If row% Then
  CommandPath$ = ITabGet(envTab&,row%,2)        ' read col 2
Else
  CommandPath$ = ""     ' not found
EndIf
ITabDelete envTab&      ' clean up
```

Function ITabFind

Search in a table for data, given column to search in, given from-which-row to search from.
The function returns the row number of the first match-occurrence. If no matching-occurrence is found, the function returns a zero.

Usage:

```
Result% = ITabFind(Handle&, data$, row%, col%, type%)
```

Types:

IT_EXACT	The comparing is done exact.
IT_WILD	Search substring in any position of the column, as a wildcard search "*substring*". The "*" should not be included.

The IT_* parameters are defined as global constants in then file "IDBTABLE.BAS"

Example:

```
FileSubstStr "MYPROG.TXT", "OLDLIB", "NEW LIB"
' This call is supposed to open the file "MYPROG.TXT", replace all occurences
' of the string "OLDLIB" with "NEW LIB" and write the file back to disk.
' The code for this task can be written like this:

Sub FileSubstStr (ByVal FileName$, ByVal FromStr$, ByVal ToStr$)
   table& = ITabRead(FileName$, IT_TEXTFILE)
   row% = 0
   Do
     row% = ITabFind(table&, FromStr$, row% + 1, 1, IT_WILD)
     If row% = 0 Then Exit Do
     ITabPutLine table&,row%,SubstAll(FromStr$,ToStr$,ITabGetLine(table&,row%))
   Loop
   ok% = ITabWrite(table&, FileName$, IT_TEXTFILE)
   ITabDelete table&
End Sub

' The line ITabPutLine table&,row%,SubstAll(FromStr$,ToStr$,ITabGetLine(table&,row%))
' may look unreadable, but this illustrates the power of routines returning strings
' that can be used directly as an argument to an other routine and so on.
' The line could have been split into 3 lines like this:
' temp1$ = ITabGetLine(table&, row%)
' temp2$ = SubstAll(FromStr$, ToStr$, temp1$)
' ITabPutLine table&, row%, temp2$
```

Function ITabFindGE

Search in a sorted table, given column, for "data*". The data comparing is exact. Folded/not folded letters are evaluated differently. It is essential that the table is pre-sorted**.** The function returns the row number of the first match-occurrence which is greater or equal(GE). If no matching-occurrence is found, the function returns a zero.

Usage:
```
Result% = ITabFindGE(Handle&, findStr$, col%)
```

Example:
A very fast way to look up data from a huge ascii file can be done this way:
An ascii file consists of 20,000 lines where each line is 80 + 2 positions long. (Cr/LF=2). In a VB loop the 8 first characters of each line in the ascii file is read into an array. The file is assumed to be sorted.

To get hold of data from the ascii file:
Search in the table and get match based on the 8 characters. If match, the function *ITabfindGE* returns the row number. Knowing the fact that each line is 82 bytes long, the exact bytes position within the ascii file is [(matching row number-1) * 82].

```
Dim Found, BytePos As integer
Dim DataLine As string
Found = ITabFindGE(MyTable&, "1234PROD", 1) 'Search in the table
BytePos = (Found-1) * 82
'Knowing the absolute byte position the Basic operators are used:
DataLine=String(82," ") 'Define the variable to be read in
Open  "Data.Txt"  #1    'Open the ascii file
Seek  #1, , BytePos     'Set file pointer to exact position in the file
Get   #1, , DataLine    'Get the data line from the file
Close #1                'Close the ascii file
```

Function ITabGet

Read data from a cell in a table.

Usage:
```
Result$ = ITabGet(Handle&, Row%, Col%)
```

Example:

```
' Display search path in a ListBox:
eTab& = ITabEnvList()
row%  = ITabFind(eTab&, "PATH", 1, 1, IT_EXACT)
path$ = ITabGet(eTab&, row%, 2) ' e.g. "C:\DOS;C:\WINDOWS;D:\UTILS;E:\PROG"
i%=1
Do
   p$ = PickWord(path$, i%, Asc(";"))
   If Len(p$) = 0 Then End Loop
   List1.AddItem p$
   i% = i% + 1
Loop
ITabDelete eTab&
```

Function ITabGetInt

Read data from a cell in a table and return an Integer.

Usage:
```
Result% = ITabGetInt(Handle&, Row%, Col%)
```

Function ITabGetLine

Read data from a line in a table. This is practical when reading rows from tables with only one column.

Usage:
```
Result$ = ITabGetLine(Handle&, Row%)
```

Function ITabGetLong

Read data from a cell in a table and return a Long Integer.

Usage:
```
Result& = ITabGetLong(Handle&, Row%, Col%)
```

Function ITabGetReal

Read data from a cell in a table and return a Real (double precision floating point) number.

Usage:
```
Result#  =  ITabGetReal (Handle&, Row%, Col%)
```

Function ITabGetColWidth

The function returns the width, as an integer, of a given column

Usage:
```
Result% = ITabGetColWidth(Handle&, Col%)
```

Example:
```
' What is the longest line in a file ?
aTab& = ITabRead("C:\AUTOEXEC.BAT", IT_TEXTFILE)
longestLine% = ITabGetColWidth(aTab&)
ITabDelete(aTab&)
```

Function ITabGetNumColumns

The function returns the number of columns the table consists of.

Usage:
```
Result% = ITabGetNumColumns(Handle&)
```

Example:
```
' Find out how many columns there are in a TAB-delimited file:
aTab& = ITabRead("DATAFILE.CSV", IT_CSVFILE + 9)
numCols% = ITabGetNumColumns(aTab&)
ITabDelete(aTab&)
```

Function ITabGetNumLines

This function returns the current number of lines the table.

Usage:
```
Result% = ITabGetNumLines(Handle&)
```

Note: The number of lines in a table is not static.
Several routines are capable of changing the number of lines i a table:

ITabInsertLine
ITabInsertLines
ITabRemoveLine
ITabRemoveLines

Calling ITabGetNumLines is normaly the logical thing to do after the following calls:

ITabRead
ITabReadFixedRecLenFile
ITabEnvList
ITabDir
ITabCopyFromVTSS

Example:
```
table& = ITabRead ("\AUTOEXEC.BAT", IT_TEXTFILE)
lines% = ITabGetNumLines(table&)

' Note: is lines%=0, there are two possibilities:
' A) the file exists and has 0 lines
' B) the file does not exist (suppose we had the wrong drive?)
' If the difference is significant, you should check for the existence of
' the file before you attempt to read it.

' The following function determines whether a file exist or not:
Function FileExist% (ByVal FileName$)
   tempTab& = ITabDir(FileName$, 1)
   If ITabGetNumLines(tempTab&) Then
      FileExist% = True
   Else
      FileExist% = False
   End If
   ITabDelete tempTab&
End Function
```

Function ITabGetSize

The function returns the size of a given table in bytes. Once the table is dimensioned by the ITabNew operator, the table

does not occupy memory space of any consideration. When data is loaded into the table an increase in memory consumption can be observed. The memory consumption is dynamic and depends on the amount of loaded data.

Usage:

```
Result& = ITabGetSize(Handle&)
```

Sub ITabInsertLine

Insert a blank line/row in a table at a given line number, atLine%. The new inserted line will contain blank cells in all columns. The lines at and below the insert point will be pushed down one position. The number of lines in the table will be affected/changed, see *ITabGetNumLines*(+1).

Usage:
```
ITabInsertLine Handle&, atLine%
```

Sub ITabInsertLines

Insert blank lines/rows in a table at a given line number, atLine%. The new inserted lines will contain blank cells in all columns. The lines at and below the insert point will be pushed down as many positions as the number of inserted lines, numLines%. The number of lines in the table will be affected/changed, see *ITabGetNumLines*(+n)

Usage:
```
ITabInsertLines Handle&, atLine%, numLines%
```

Function ITabNew

Create and dimension a new table. Returns a handle which will identify the table.

Usage:
```
Handle& = ITabNew(rows%, columns%)
```

Remember to delete the table, using ITabDelete, when it is no longer needed (free system resources).

Example:
```
' Define a table consisting of 100 rows and 10 columns pr row.
Mytab&= ITabNew(100, 10)
```

Function ITabNewArray

Create a table with one column . The function returns a handle which will identify the table.

Usage:
```
Handle& = ITabNewArray(ByVal lines%)
```

Remember to delete the table, using ITabDelete, when it is no longer needed (free system resources).

Example:
```
' Define a table consisting of 100 lines (one column).
Mytab&= ITabNewArray(100)
```

Sub ITabPut

Put string data into a cell in the table.

Usage:
```
ITabPut Handle&, Row%, Col%, DataString$
```

Sub ITabPutInt

Put numeric (integer) data into a cell in the table.

Usage:
```
ITabPutInt  Handle&, Row%, Col%, IntegerNumber%
```

Sub ITabPutLine

Put data into a line in the table (as string). This is a practical call for writing lines to tables with only one column.

Usage:
```
ITabPutLine Handle&, Row%, DataString$
```

Sub ITabPutLong

Put numeric data into a cell in the table (as Long).

Usage:
```
ITabPutLong Handle&, Row%, Col%, LongNumber&
```

Sub ITabPutReal

Put numeric data into a cell in the table (as Double).

Usage:
```
ITabPutReal Handle&, Row%, Col%, DoubleRealNumber#
```

Function ITabRead

Read a file into a new table. The table is dimensioned depending on the contents of the disk file.

Usage:
```
Handle& = ITabRead(FileName$, FileType%)
```

Remember to delete the table, using ITabDelete, when it is no longer needed (free system resources):

If a table file, IT_TABFILE, is read, the file that was written from a table as source, the table will gain the same dimension as the source table had.

If an ordinary textfile, IT_TEXTFILE, is read, the table will become an array.

***Filetype*:**

IT_TABFILE	Read an earlier written table.
IT_TEXTFILE	An ordinary text file
IT_CSVFILE + Delim.	Read a file where the columns are delimited by a given character
IT_CSV0FILE + Delim	As above, but the first line in the file is written to line zero in the table (typically for column headers)

Delim is the ascii value of then delimitter character:`Tab=9,(Asc";"),(Asc",")`

Additional parameters:

+ IT_ASCII	Translate from DOS characters to Windows characters
+ STRIP_T	Remove trailing blanks

The IT_* parameters are defined as global constants in the file "IDBTABLE.BAS"

Example:
```
Mytab&= ITabRead("Written.Tab", IT_TABFILE)
```
Read an earlier written table file named "Written.Tab" to memory. How the table is dimensioned is determined by the dimension of the read table/file "Written.Tab"

```
Mytab& = ITabRead(Text.fil,IT_TEXTFILE[+ STRIP_T][ + IT_ASCII])
```
Read a textfile named "Text.fil" to memory and pr line remove trailing blanks, optional, and translate from DOS to Windows characters-set, optional. The table becomes an array.

Function ITabReadFixedRecLenFile

Read a file with fixed record length to a new table. The function returns the table handle.

Usage:
```
Handle& = ITabReadFixedRecLenFile(FileName$, fmt$)
```

Remember to delete the table, using ITabDelete, when it is no longer needed (free system resources):

A linefeed between each record is assumed. The "fmt$" parameter tells the system what to be picked from the record and placed to which column in the table. Capitalised letters (A-Z) are used for giving the position and length (repeated) in the record, The letter used. also indicates which column the data is to be put into. "A" is column 1, "B" = 2 and "C" = 3..."Y" = 25 and "Z" = 26. The width is given by repeating the letter. "A" means, pick one character and put it in column 1. "BBBB" means pick four characters and put it in column 2. "ZZZ" means 3 characters to column 26. The sequence can mixed, order, according to the datafile, as long as "A" comes before "B". Further on empty columns can be reserved in the table by skipping letters in the sequence. Leading and trailing blanks will be stripped before data is put to the table.

Example:

The first line shows the "fmt$" and the 10 to follow the datalines within a datafile:

```
"AAAAAAAAAAAAABBBBBBBB  DD CC EE  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
File Name    Size      Date      D e s c r i p t i o n
=================================================================
TOLL20.ZIP   35652      04-28-93  Tool Button Custom Control For VBasic
VB2_TB.ZIP   162895    01-14-93  The Ultimate VBASIC v2.0 Add-On, Supe
VB4EX.ZIP    116486    03-14-92  Example Of How To Use DLL's With VisB
VBE1NG.ZIP   118521    04-04-93  Visual Basic Engine For Making DataBa
VGX3.ZIP      194111    01-06-93  VGA Graphics File Lib For QB/BASIC Pr
VXBASE.ZIP   212606    03-25-92  XBase Windows Visual BASIC Functions
VXBDOC.ZIP   132274    03-19-92  XBase Windows Visual BASIC Docs [2/2]
WBB12.ZIP    266520    09-10-92  BasicBasic For Windows v1.2

The table defined with 6 columns and 10 lines:
1.            2.        3. 4. 5.  6.column:
File Name     Size      e  Da     D e s c r i p t i o n
============ ========  == == ==  ====================================
TOLL20.ZIP    35652     28 04 93  Tool Button Custom Control For VBasic
VB2_TB.ZIP    162895    14 01 93  The Ultimate VBASIC v2.0 Add-On, Supe
e.t.c.
```

Sub ITabRemoveLine

Remove a line/row in a table from a given line number, atLine%. The lines below the given linenumber will be scrolled up one line. The number of lines in the table will be affected/changed, *ITabGetNumLines*(-1*)*.

Usage:
```
ITabRemoveLine Handle&, atLine%
```

Sub ITabRemoveLines

Remove lines/rows in a table from a given line number, atLine%. The lines below the given linenumber, atLine%, + the

number of deleted lines, num%, will scroll up. The number of lines in the table will be affected/changed, ITabGetNumLines% (-n)

Usage:
```
ITabRemoveLines Handle&, atLine%, num%
```

Sub ITabSmartSort

Sort a table by the contents of the given column. This rutine sorts both text and numbers logically. The sorting is not case sensitive. If there are duplicates in the column beeing sorted, the original order will be kept. This makes it possible to sort on several columns just by repeating this call (sort least significant column first).

Usage:
```
ItabSmartSort Handle&, Col%
```

If the column number is negative, the sorting will be descending on column `(-Col%)`.

Example
Given then table, "TestTab", consisting of one column and data as follows:

Number 1 of 100
Number 10 of 100
Number 100 of 100
Number 2 of 100
Number 20 of 100
Number 20 of 50
(The result of an ordinary sort in Excel)

```
Call ITabSmartsort(TestTable&, 1)
```

The result of "smart"sorting :

Number 1 of 100
Number 2 of 100
Number 10 of 100
Number 20 of 50
Number 20 of 100
Number 100 of 100

Function ITabWrite

Write a table to a disk file. Return value is a zero.

Usage:
```
Result% = ITabWrite(Handle&, FileName$, FileType%)
```

FileType:
IT_TABFILE The internal format for reading/writing
IT_TEXTFILE Ordinary textfile which can be read to a table consisting of one column

Additional options:
IT_ASCII Translate from Windows to the DOS character set.

The IT_* parameters are defined as global constants in the file "IDBTABLE.BAS"

The following functions are defined in IDBVTSS.DLL

The functions are only useful if you have got the Formula One VBX/DLL from Visual Tools.

Function ITabCopyFromVTSS

Read the contents of a Visual-Tools spreadsheet to a new table. This function makes it possible to read Excel 4.0 spreadsheet and *.vts files, internal format of Visual Tools, indirectly via the worksheet.
The number of lines in the table will be as many as it are datafilled lines in the worksheet. Blank trailing lines in the worksheet are disregarded.

Usage:
```
Handle& = ITabCopyFromVTSS(SShandle&)
```

Remember to delete the table, using ITabDelete, when it is no longer needed (free system resources):

Example:
```
MyTab& = ITabCopyFromVTSS(Sheet1.SS)
```

Function ITabCopyToVTSS

Dump/write the contents of a table to a Visual Tools spreadsheet.
This function makes it possible to write Excel 4.0 spreadsheet and *.vts files, internal format of Visual Tools, indirectly via the spreadsheet.

Usage:
```
ITabCopyToVTSS Handle&, SShandle&
```

Example:
```
ITabCopyToVTSS MyTab&, Sheet1.SS
```