TAS2XLS.DLL, TAS2EXCEL.DLL Documentation

NATIVE XLS

Add-On for TAS Professional

DLL's included: TAS2XLS.dll TAS2EXCEL.dll

www.aura-soft.com

Table of Contents

1 DLL Features

DLL features

TAS2XLS.dll and TAS2EXCEL.dll are made to provide a solution for managing XLS files from TAS programming language. Main features are creating new XLS file from scratch, reading and writing cell values and formulas. You can add, remove, or rename sheets, You can even copy rows, or insert row at specific position. Named cells are supported. Search for cell values is possible on row basis and column basis, also sheet search is possible too.

DLL difference

Features included in both DLL's are the same. TAS2XLS.dll is much faster and You don't have to have any other XLS software to manage XLS files. TAS2EXCEL.dll is using MS excel application in background to manage XLS files. TAS2EXCEL.dll is better for dealing with complicated files with protected cells etc.

Demo vs Licensed version

Demo dll's are fully functional for unlimited time period. Also demo dll's will put our Aurasoft company Aurasoft signature in first 6 cells in first column on each sheet. With demo dll's You can use 3 sheets, 5 columns and 20 rows.

System requirements

TAS2XLS.dll is tested on

Windows XP Professional SP2 English
TASProfessional 7.4 Build 2.1

TAS2EXCEL.dll is tested on

Windows XP Professional SP2 English
TAS Professional 7.4 Build 2.1
MS Excel 2003 English

License terms

You may give demo dll's to anyone without charge. You may not give, sold or rent licensed dll's. You may not modify dll's. You may not use demo dll's for commercial purpose. You may use licensed dll's for commercial purpose. You may not distribute licensed dll's with development systems.

2 DLL Specification

TAS2XLS.DLL and TAS2EXCEL Specification

function FileOpen (pod:**array** of Byte):integer;**stdcall**;
function CellWriteString256 (pod:**array** of Byte):integer;**stdcall**;
function CellWriteString32k (pod:**array** of Byte):integer;**stdcall**;
function CellWriteFormula32k (pod:**array** of Byte):integer;**stdcall**;
function CellWriteDateTime (pod:**array** of Byte):integer;**stdcall**;
function CellWriteDate (pod:**array** of Byte):integer;**stdcall**;
function CellWriteTime (pod:**array** of Byte):integer;**stdcall**;
function CellWriteInteger (pod:**array** of Byte):integer;**stdcall**;
function CellWriteDouble (pod:**array** of Byte):integer;**stdcall**;
function CellWriteBoolean (pod:**array** of Byte):integer;**stdcall**;
function CellReadString256 (**var** pod:**array** of Byte):integer;**stdcall**;
function CellReadString32k (**var** pod:**array** of Byte):integer;**stdcall**;
function CellReadFormula32k (**var** pod:**array** of Byte):integer;**stdcall**;
function CellReadDateTime (**var** pod:**array** of Byte):integer;**stdcall**;
function CellReadDate (**var** pod:**array** of Byte):integer;**stdcall**;
function CellReadTime (**var** pod:**array** of Byte):integer;**stdcall**;
function CellReadInteger (**var** pod:**array** of Byte):integer;**stdcall**;
function CellReadDouble (**var** pod:**array** of Byte):integer;**stdcall**;
function CellReadBoolean (**var** pod:**array** of Byte):integer;**stdcall**;
function FindSheet (**var** pod:**array** of Byte):integer;**stdcall**;
function FindColumn (**var** pod:**array** of Byte):integer;**stdcall**;
function FindRow (**var** pod:**array** of Byte):integer;**stdcall**;
function SheetAdd ():integer;**stdcall**;
function SheetDelete (pod:**array** of Byte):integer;**stdcall**;
function SheetCount (pod:**array** of Byte):integer;**stdcall**;
function SheetRename (pod:**array** of Byte):integer;**stdcall**;
function NamedCellPosition (**var** pod:**array** of Byte):integer;**stdcall**;
function CopyRow (pod:**array** of Byte):integer;**stdcall**;
function InsertRows (pod:**array** of Byte):integer;**stdcall**;
function Version (**var** pod:**array** of Byte):integer;**stdcall**;
function LanguageSet (**var** pod:**array** of Byte):integer;**stdcall**;
function FileSave ():integer;**stdcall**;
function FileClose ():integer;**stdcall**;
function FileNew (pod:**array** of Byte):integer;**stdcall**;

Simple TAS Professional example program code

3 Data Types

Data Types

DELPHI	TAS	SIZE[bytes]	RANGE
Byte	Byte	1	0..255
Word	/	2	0..65535
SmallInt	Integer	2	-32768..+32767
Integer	Record	3	-2147483648.. +2147483647
Double	Numeric	8	5*10^-324..1.7*10^308
Boolean	Logical	1	False..True
Char[]	Alphanumeric	1..4GB	0..255
/	Date	4	
/	Time	4	

Table1. Comparable data types in TAS and DELPHI

TAS	.XLS
Byte	Float
Integer	
Record	
Numeric	
Logical	Boolean
Alphanumeric	WideString
	Formula
Date	DateTime
Time	DateTime

Table2. Conversion between types from TAS to .XLS

4 Functions

4.1 function FileOpen (pod:array of Byte):integer;stdcall

function FileOpen (pod:**array** of Byte):integer;**stdcall**;

```
TInFileOpen = packed record
   TotalLength:Word;
   FilenameLength: Integer;
   Filename: array[0..255] of Char;
end;
```

Description:

This function opens .XLS file. If DLL license expired, ERROR is returned;

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define size1 type R
define dHndl, numInt type I
define filename type A size 256

//.DLL file
dHndl = LOAD_DLL('C:\TAS2XLS.dll')
// dHndl = LOAD_DLL('C:\TAS2EXCEL.dll')

//.XLS
filename = 'C:\test.xls'
size1=SIZE(filename,'a')

numInt = DLLFC(dHndl,'FileOpen','I', CREC(size1), TRIM(filename))
msg numInt
```

See also:
FileClose

4.2 function CellWriteString256 (pod:array of Byte):integer;stdcall

function CellWriteString256 (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteString256 = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  ValueLength: Integer;
  Value: array[0..255] of Char;
end;
```

Description:

This function writes a string into specified cell. Maximum string length is 256 characters.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define cellvalue type A size 256
define ValueLength type R

sheet = 1
column = 5
row = 2

//cellvalue=''
cellvalue='ABC'
ValueLength = SIZE(cellvalue,'a')

numInt = DLLFC(dHndl,'CellWriteString256','I', CREC(sheet), CREC(column),
           CREC(row), CREC(ValueLength), TRIM(cellvalue))
msg numInt
```

See also:
FileOpen, FileClose

4.3 function CellWriteString32k (pod:array of Byte):integer;stdcall

function CellWriteString32k (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteString32k = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  ValueLength: Integer;
  Value: array[0..32766] of Char;
end;
```

Description:

This function writes a string into specified cell. Maximum string length is 32767 characters.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define cellvalue32k type A size 32767
define ValueLength type R

sheet = 1
column = 5
row = 2

//cellvalue32k=''
cellvalue32k='ABC32K'
ValueLength = SIZE(cellvalue32k,'a')

numInt = DLLFC(dHndl,'CellWriteString32k','I', CREC(sheet), CREC(column),
           CREC(row), CREC(ValueLength), TRIM(cellvalue32k))
msg numInt
```

See also:
FileOpen, FileClose

4.4 function CellWriteFormula32k (pod:array of Byte):integer;stdcall;

function CellWriteFormula32k (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteFormula32k = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  FormulaLength: Integer;
  Formula: array[0..32766] of Char;
end;
```

Description:

This function writes formula into specified cell. Maximum formula length is 32767 characters. Formula cannot be empty string.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define cellvalue32k type A size 32767
define ValueLength type R

sheet = 1
column = 5
row = 2

//cellvalue32k = '=A1+B1'
```

```
cellvalue32k = 'A1+B1'
ValueLength = SIZE(cellvalue32k,'a')

numInt = DLLFC(dHndl,'CellWriteFormula32k','I', CREC(sheet), CREC(column),
         CREC(row), CREC(ValueLength), TRIM(cellvalue32k))
msg numInt
```

See also:
FileOpen, FileClose

4.5 function CellWriteDateTime (pod:array of Byte):integer;stdcall;

function CellWriteDateTime (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteDateTime = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  DateDay: Byte;
  DateMonth: Byte;
  DateYear: SmallInt;
  TimeSecHundredths: Byte;
  TimeSec: Byte;
  TimeMin: Byte;
  TimeHour: Byte;
end;
```

Description:

This function writes date and time into specified cell as .XLS native datetime value.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mydate type D
define mytime type T

sheet = 1
column = 5
row = 2

mydate = DATE();
mytime = TIME();
```

```
numInt= DLLFC(dHndl,'CellWriteDateTime','I', CREC(sheet), CREC(column),
        CREC(row), mydate, mytime)
msg numInt
```

See also:
FileOpen, FileClose

4.6 function CellWriteDate (pod:array of Byte):integer;stdcall;

function CellWriteDate (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteDate = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  DateDay: Byte;
  DateMonth: Byte;
  DateYear: SmallInt;
end;
```

Description:

This function writes date into specified cell as .XLS native datetime value.
Time portion is set to 0 hours, 0 minutes, 0 seconds, 0 milliseconds.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mydate type D

sheet = 1
column = 5
row = 2

mydate = DATE();

numInt= DLLFC(dHndl,'CellWriteDate','I', CREC(sheet), CREC(column), CREC(row),
        mydate)
msg numInt
```

See also:
FileOpen, FileClose

4.7 function CellWriteTime (pod:array of Byte):integer;stdcall;

function CellWriteTime (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteTime = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  TimeSecHundredths: Byte;
  TimeSec: Byte;
  TimeMin: Byte;
  TimeHour: Byte;
end;
```

Description:

This function writes time into specified cell as .XLS native datetime value. Milliseconds are set to 0. Date portion is set to 1899 year, 12 month, 30 day.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mytime type T

sheet = 1
column = 5
row = 2

mytime = TIME();

numInt= DLLFC(dHndl,'CellWriteTime','I', CREC(sheet), CREC(column), CREC(row),
          mytime)
msg numInt
```

See also:
FileOpen, FileClose

4.8 function CellWriteInteger (pod:array of Byte):integer;stdcall;

function CellWriteInteger (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteInteger = packed record
  TotalLength:Word;
  Sheet: Integer;
```

```
  Column: Integer;
  Row: Integer;
  Number: Integer;   //1,2 or 4 bytes
end;
```

Description:

This function writes integer value into specified cell. Supported types are:
1byte 0..255,
2byte -32768..+32767,
4byte -2147483648.. +2147483647,
with byte order from LSByte to MSByte.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mybyte type B
define myint type I
define myrecord type R

sheet = 1
column = 5
row = 2

mybyte = 200
numInt= DLLFC(dHndl,'CellWriteInteger','I', CREC(sheet), CREC(column), CREC(row),
          CBYT(mybyte))
myint  = 30000
numInt= DLLFC(dHndl,'CellWriteInteger','I', CREC(sheet), CREC(column),
          CREC(row+1), CINT(myint))
myrecord  = 2000000000
numInt= DLLFC(dHndl,'CellWriteInteger','I', CREC(sheet), CREC(column),
          CREC(row+2), CREC(myrecord))
msg numInt

msg mybyte
msg myint
msg myrecord
```

See also:
FileOpen, FileClose

4.9 function CellWriteDouble (pod:array of Byte):integer;stdcall;

function CellWriteDouble (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteDouble = packed record
  TotalLength:Word;
  Sheet: Integer;
```

```
  Column: Integer;
  Row: Integer;
  Number: Double;
end;
```

Description:

This function writes number with decimals into specified cell in standard IEEE 64bit floating point format.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mynumeric type N dec 4

sheet = 1
column = 5
row = 2

mynumeric = -118.625
numInt= DLLFC(dHndl,'CellWriteDouble','I', CREC(sheet), CREC(column), CREC(row),
          CFLT(mynumeric))
msg numInt
```

See also:
FileOpen, FileClose

4.10 function CellWriteBoolean (pod:array of Byte):integer;stdcall;

function CellWriteBoolean (pod:**array** of Byte):integer;**stdcall**;

```
TInCellWriteBoolean = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  Value: Boolean;
end;
```

Description:

This function writes boolean value into specified cell.

Sheet index starts with 1.
Column index starts with 1.

Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mybool type L

sheet = 1
column = 5
row = 2

mybool = True
numInt= DLLFC(dHndl,'CellWriteBoolean','I', CREC(sheet), CREC(column),
          CREC(row), CTOL(mybool))
msg numInt
```

See also:
FileOpen, FileClose

4.11 function CellReadString256 (var pod:array of Byte):integer;stdcall;

function CellReadString256 (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadString256 = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  ValueLength: Integer;            //OUT
  Value: array[0..255] of Byte;    //OUT
end;
```

Description:

This function reads computed cell value and returns it as string. Maximum string length is 256 characters.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
```

```
define cellvalue type A size 256
define aValueLength type A size 4
define ValueLength type R

sheet = 1
column = 5
row = 2

cellvalue = '@'
aValueLength = '@@@@'

numInt = DLLFC(dHndl,'CellReadString256','I', CREC(sheet), CREC(column),
            CREC(row), aValueLength, cellvalue)

XFER FROM aValueLength TO ValueLength NCHR 4

cellvalue = MID(cellvalue,1,ValueLength)
msg cellvalue
msg (SIZE(cellvalue,'a'))
```

See also:
FileOpen, FileClose

4.12 function CellReadString32k (var pod:array of Byte):integer;stdcall;

function CellReadString32k (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadString32k = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  ValueLength: Integer;              //OUT
  Value: array[0..32766] of Byte;   //OUT
end;
```

Description:

This function reads computed cell value and returns it as string. Maximum string length is 32767 characters.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
```

```
define cellvalue type A size 32767
define aValueLength type A size 4
define ValueLength type R

sheet = 1
column = 5
row = 2

cellvalue = '@'
aValueLength = '@@@@'

numInt = DLLFC(dHndl,'CellReadString32k','I', CREC(sheet), CREC(column),
            CREC(row), aValueLength, cellvalue)

XFER FROM aValueLength TO ValueLength NCHR 4

cellvalue = MID(cellvalue,1,ValueLength)
msg cellvalue
msg (SIZE(cellvalue,'a'))
```

See also:
FileOpen, FileClose

4.13 function CellReadFormula32k (var pod:array of Byte):integer;stdcall;

function CellReadFormula32k (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadFormula32k = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  FormulaLength: Integer;          //OUT
  Value: array[0..32766] of Byte;  //OUT
end;
```

Description:

This function reads cell formula and returns it as string. Maximum formula length is 32767 characters. Returned formula does not contain leading '='. Empty cell results with an error.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define cellvalue type A size 32767
define aValueLength type A size 4
define ValueLength type R

sheet = 1
column = 5
row = 2

cellvalue = '@'
aValueLength = '@@@@'

numInt = DLLFC(dHndl,'CellReadFormula32k','I', CREC(sheet), CREC(column),
            CREC(row), aValueLength, cellvalue)

XFER FROM aValueLength TO ValueLength NCHR 4

cellvalue = MID(cellvalue,1,ValueLength)
msg cellvalue
msg (SIZE(cellvalue,'a'))
```

See also:
FileOpen, FileClose

4.14 function CellReadDateTime (var pod:array of Byte):integer;stdcall;

function CellReadDateTime (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadDateTime = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  DateDay: Byte;          //OUT
  DateMonth: Byte;        //OUT
  DateYear: SmallInt;     //OUT
  TimeSecHundredths: Byte;  //OUT
  TimeSec: Byte;          //OUT
  TimeMin: Byte;          //OUT
  TimeHour: Byte;         //OUT
end;
```

Description:

This function reads cell value as native datetime value. Datetime value is converted to TAS format and returned separately date and time. For empty .XLS cell, date is 1899-12-30 and time is 00:00:00:000AM.

Sheet index starts with 1.
Column index starts with 1.

Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mydate type D
define mytime type T

sheet = 1
column = 5
row = 2

numInt= DLLFC(dHndl,'CellReadDateTime','I', CREC(sheet), CREC(column),
            CREC(row), mydate, mytime)

msg DTOS(mydate)
msg mytime
```

See also:
FileOpen, FileClose

4.15 function CellReadDate (var pod:array of Byte):integer;stdcall;

function CellReadDate (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadDate = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  DateDay: Byte;       //OUT
  DateMonth: Byte;     //OUT
  DateYear: SmallInt;  //OUT
end;
```

Description:

This function reads cell value as native datetime value. Datetime value is converted to TAS date format. For empty cell, date is 1899-12-30.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mydate type D

sheet = 1
column = 5
row = 2

numInt= DLLFC(dHndl,'CellReadDate','I', CREC(sheet), CREC(column), CREC(row),
            mydate)

msg DTOS(mydate)
```

See also:
FileOpen, FileClose

4.16 function CellReadTime (var pod:array of Byte):integer;stdcall;

function CellReadTime (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadTime = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  TimeSecHundredths: Byte;  //OUT
  TimeSec: Byte;            //OUT
  TimeMin: Byte;            //OUT
  TimeHour: Byte;           //OUT
end;
```

Description:

This function reads cell value as native datetime value. Datetime value is converted to TAS time format. For empty cell time is 00:00:00:000AM.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mytime type T

sheet = 1
column = 5
row = 2
```

```
numInt= DLLFC(dHndl,'CellReadTime','I', CREC(sheet), CREC(column), CREC(row),
          mytime)

msg mytime
```

See also:
FileOpen, FileClose

4.17 function CellReadInteger (var pod:array of Byte):integer;stdcall;

function CellReadInteger (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadInteger = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  Number: Integer;  //OUT 1,2 or 4 bytes
end;
```

Description:

This function reads integer number and depending on input parameter length returns
1byte 0..255,
2byte -32768 .. +32767 or
4 byte -2147483648 .. +2147483647 integer.
For empty cell error is returned. Number with decimals are rounded.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define alphaint1 type a size 1
define alphaint2 type a size 2
define alphaint4 type a size 4
define rint type R
define iint type I
define bint type B

sheet = 1
column = 5
row = 2

alphaint1='@'
numInt= DLLFC(dHndl,'CellReadInteger','I', CREC(sheet), CREC(column), CREC(row),
          alphaint1)
XFER FROM alphaint1 TO bint NCHR 1
msg bint
alphaint2='@@'
numInt= DLLFC(dHndl,'CellReadInteger','I', CREC(sheet), CREC(column),
```

```
            CREC(row+1), alphaint2)
  XFER FROM alphaint2 TO iint NCHR 2
  msg iint
  alphaint4='@@@@'
  numInt= DLLFC(dHndl,'CellReadInteger','I', CREC(sheet), CREC(column),
            CREC(row+2), alphaint4)
  XFER FROM alphaint4 TO rint NCHR 4
  msg rint
```

See also:
FileOpen, FileClose

4.18 function CellReadDouble (var pod:array of Byte):integer;stdcall;

function CellReadDouble (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadDouble = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  Number: Double;   //OUT
end;
```

Description:

This function reads number with decimals from specified cell in standard IEEE 64bit floating point format.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
  define sheet, row, column type R
  define mynumeric type N dec 4

  sheet = 1
  column = 5
  row = 2

  numInt = DLLFC(dHndl,'CellReadDouble','I', CREC(sheet), CREC(column), CREC(row),
            mynumeric)

  msg mynumeric
```

See also:
FileOpen, FileClose

4.19 function CellReadBoolean (var pod:array of Byte):integer;stdcall;

function CellReadBoolean (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInCellReadBoolean = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;
  Value: Boolean;   //OUT
end;
```

Description:

This function reads boolean value from specified cell.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define mybool type L

sheet = 1
column = 5
row = 2

numInt = DLLFC(dHndl,'CellReadBoolean','I', CREC(sheet), CREC(column),
             CREC(row), myBool)

msg mybool
```

See also:
FileOpen, FileClose

4.20 function FindSheet (var pod:array of Byte):integer;stdcall;

function FindSheet (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInFindSheet = packed record
  TotalLength:Word;
  Sheet: Integer;   //OUT
  SheetNameLength: Integer;
  SheetName: array[0..255] of Char;
end;
```

Description:

This function finds sheet by name and returns index of sheet. First sheet index is 1. Sheet name cannot be empty. If sheet not found zero is returned.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet type R
define asheet type A size 4
define SheetNameLength type R
define SheetName type A size 256

asheet='@@@@'
SheetName = 'Sheet3'
SheetNameLength = SIZE(SheetName,'a')

numInt = DLLFC(dHndl,'FindSheet','I', asheet, CREC(SheetNameLength),
            TRIM(SheetName))

XFER FROM asheet TO sheet NCHR 4

msg sheet
```

See also:
FileOpen, FileClose

4.21 function FindColumn (var pod:array of Byte):integer;stdcall;

function FindColumn (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInFindColumn = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;  //OUT
  Row: Integer;
  ColumnNameLength: Integer;
  ColumnName: array[0..255] of Char;
end;
```

Description:

This function finds column by header name in header row and returns index of column. Column name can be empty. For cells that contain formula, computed value is tested. For cells that contain number formatting, the nonformatted value is tested. If column not found zero is returned.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define ColumnName type A size 256
define acol type a size 4
define ColNameLength type R

sheet = 1
row = 1
ColumnName = 'Third'
acol = '@@@@'
ColNameLength = SIZE(ColumnName,'a')

numInt = DLLFC(dHndl,'FindColumn','I', CREC(sheet), acol, CREC(row),
              CREC(ColNameLength), TRIM(ColumnName))

XFER FROM acol TO column NCHR 4

msg column
```

See also:
FileOpen, FileClose

4.22 function FindRow (var pod:array of Byte):integer;stdcall;

function FindRow (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInFindRow = packed record
  TotalLength:Word;
  Sheet: Integer;
  Column: Integer;
  Row: Integer;     //OUT
  RowNameLength: Integer;
  RowName: array[0..255] of Char;
end;
```

Description:

This function finds row by header name in header column and returns index of row. Row name cannot be empty. For cells that contain formula, computed value is tested. For cells that contain number formatting, the nonformatted value is tested. If row not found zero is returned.

Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK

1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define RowName type A size 256
define arow type a size 4
define RowNameLength type R

sheet = 1
column = 2
RowName = 'MyRow'
arow  = '@@@@'
RowNameLength = SIZE(RowName,'a')

numInt = DLLFC(dHndl,'FindRow','I', CREC(sheet), CREC(column), arow,
            CREC(RowNameLength ), TRIM(RowName))

XFER FROM arow TO row NCHR 4
msg row
```

See also:
FileOpen, FileClose

4.23 function SheetAdd ():integer;stdcall;

function SheetAdd ():integer;**stdcall**;

Description:

This function adds empty sheet in workbook. New sheet is added as last sheet.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
numInt = DLLFC(dHndl,'SheetAdd','I')

msg numInt
```

See also:
FileOpen, FileClose

4.24 function SheetDelete (pod:array of Byte):integer;stdcall;

function SheetDelete (pod:**array** of Byte):integer;**stdcall**;

```
TInSheetDelete = packed record
  TotalLength:Word;
  Sheet: Integer;
```

```
end;
```

Description:

This function deletes specified sheet from workbook. Sheet will be deleted if it is not only sheet in workbook. On this kind of delete try, sheet will not be deleted and no error message.

Sheet index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet type R

sheet = 1
numInt = DLLFC(dHndl,'SheetDelete','I', CREC(sheet))
msg numInt
```

See also:
FileOpen, FileClose

4.25 function SheetCount (pod:array of Byte):integer;stdcall;

function SheetCount (pod:**array** of Byte):integer;**stdcall**;

```
TInSheetCount = packed record
  TotalLength: Word;
  SheetCount: Integer;
end;
```

Description:

This function adjusts number of sheets in workbook. Workbook must contain at least one sheet. Number of sheets can be increased and decreased.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheetCount type R

sheetCount = 5
numInt = DLLFC(dHndl,'SheetCount','I', CREC(sheetCount))
msg numInt
```

See also:

FileOpen, FileClose

4.26 function SheetRename (pod:array of Byte):integer;stdcall;

function SheetRename (pod:**array** of Byte):integer;**stdcall**;

```
TInSheetRename = packed record
  TotalLength:Word;
  Sheet: Integer;
  SheetNameLength: Integer;
  SheetName: array[0..255] of Char;
end;
```

Description:

This function renames sheet specified. Sheet index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, SheetNameLength type R
define SheetName type A size 256

sheet = 3
SheetName = 'Third'
SheetNameLength = SIZE(SheetName,'a')
numInt = DLLFC(dHndl,'SheetRename','I', CREC(sheet),
            CREC(SheetNameLength),TRIM(SheetName))
msg numInt
```

See also:
FileOpen, FileClose

4.27 function NamedCellPosition (var pod:array of Byte):integer;stdcall;

function NamedCellPosition (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInNamedCellPosition = packed record
  TotalLength:Word;
  Sheet: Integer;   //OUT
  Column: Integer; //OUT
  Row: Integer;     //OUT
  CellNameLength: Integer;
  CellName: array[0..255] of Char;
end;
```

Description:

This function gets cell position by cell name. Cell name scope is entire workbook. Sheet index starts with 1.
Column index starts with 1.
Row index starts with 1.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, column type R
define asheet, acol, arow type A size 4
define CellName type A size 256
define CellNameLength type R

CellName = 'MyCellName'
CellNameLength = SIZE(CellName,'a')
asheet = '@@@@'
acol   = '@@@@'
arow   = '@@@@'
numInt = DLLFC(dHndl,'NamedCellPosition','I', asheet, acol, arow,
               CREC(CellNameLength), TRIM(CellName))

XFER FROM asheet TO sheet NCHR 4
XFER FROM acol TO column NCHR 4
XFER FROM arow TO row NCHR 4

msg 'Sheet: ' + sheet + ' Column: ' + column + ' Row: ' + row
```

See also:
FileOpen, FileClose

4.28 function CopyRow (pod:array of Byte):integer;stdcall;

function CopyRow (pod:**array** of Byte):integer;**stdcall**;

```
TInCopyRow = packed record
  TotalLength:Word;
  Sheet: Integer;
  Row: Integer;
  NumberOfRows: Integer;
end;
```

Description:

This function copies single row specified number of times. Rows copied will overwrite existing rows. Rows are copied after the original row. Format of row being copied is copied, formulas are copied and other cell values are copied too.
Sheet index starts with 1.
Row index starts with 1.

NumberOfRows can be 1 or greater.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, numberOfRows type R

sheet = 1
row=5
numberOfRows=7

numInt = DLLFC(dHndl,'CopyRow', 'I', CREC(sheet), CREC(row),
              CREC(numberOfRows))

msg numInt
```

See also:
FileOpen, FileClose

4.29 function InsertRows (pod:array of Byte):integer;stdcall;

function InsertRows (pod:**array** of Byte):integer;**stdcall**;

```
TInInsertRows = packed record
  TotalLength:Word;
  Sheet: Integer;
  Row: Integer;
  NumberOfRows: Integer;
end;
```

Description:

This function inserts multiple rows at specified row position. Rows are added before specified row. Existing rows at specified row position and greater are moved down.
Sheet index starts with 1.
Row index starts with 1.
NumberOfRows can be 1 or greater.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define sheet, row, numberOfRows type R

sheet = 1
row=5
```

```
numberOfRows=7

numInt = DLLFC(dHndl,'InsertRows', 'I', CREC(sheet), CREC(row),
            CREC(numberOfRows))

msg numInt
```

See also:
FileOpen, FileClose

4.30 function Version (var pod:array of Byte):integer;stdcall;

function Version (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInVersion = packed record
  TotalLength:Word;
  Major: Integer;          //OUT
  Minor: Integer;          //OUT
  Release: Integer;         //OUT
  Build: Integer;          //OUT
end;
```

Description:

This function reads DLL version and returns it as 4 numbers.
Minimum values are [0, 0, 0, 0].
Maximum values are [32767, 32767, 32767, 32767].

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define aValueMajor type A size 4
define aValueMinor type A size 4
define aValueRelease type A size 4
define aValueBuild type A size 4
define ValueMajor type R
define ValueMinor type R
define ValueRelease type R
define ValueBuild type R

aValueMajor = '@@@@'
aValueMinor = '@@@@'
aValueRelease = '@@@@'
aValueBuild = '@@@@'

numInt = DLLFC(dHndl,'Version','I', aValueMajor, aValueMinor, aValueRelease,
            aValueBuild)
```

```
XFER FROM aValueMajor TO ValueMajor NCHR 4
XFER FROM aValueMinor TO ValueMinor NCHR 4
XFER FROM aValueRelease TO ValueRelease NCHR 4
XFER FROM aValueBuild TO ValueBuild NCHR 4

msg ValueMajor
msg ValueMinor
msg ValueRelease
msg ValueBuild
```

4.31 function LanguageSet (var pod:array of Byte):integer;stdcall;

function LanguageSet (**var** pod:**array** of Byte):integer;**stdcall**;

```
TInLanguageSet = packed record
  TotalLength:Word;
  Language: Integer;
end;
```

Description:

This function sets language used for message display.
0 Croatian
1 English

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
numInt = DLLFC(dHndl,'LanguageSet','I', CREC(1))
msg numInt
```

4.32 function FileSave ():integer;stdcall;

function FileSave ():integer;**stdcall**;

Description:

This function saves .XLS file.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
numInt = DLLFC(dHndl,'FileSave','I')
msg numInt
```

See also:
FileOpen, FileClose
function FileClose ():integer;**stdcall**;

Description:

This function closes .XLS file without saving.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
numInt = DLLFC(dHndl,'FileClose','I')
msg numInt
```

See also:
FileOpen, FileClose

4.33 function FileClose ():integer;stdcall;

Description:

This function closes .XLS file without saving.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
numInt = DLLFC(dHndl,'FileClose','I')
msg numInt
```

See also:
FileOpen, FileSave

4.34 function FileNew (pod:array of Byte):integer;stdcall;

function FileNew (pod:**array** of Byte):integer;**stdcall**;

```
TInFileNew = packed record
   TotalLength:Word;
   FilenameLength: Integer;
   Filename: array[0..255] of Char;
end;
```

Description:

This function creates new .XLS file. If file with the same name exists, it will be overwritten.

Returns:

0 OK
1 ERROR

TAS EXAMPLE:

```
define size1 type R
define dHndl, numInt type I
define filename type A size 256

//.DLL file
dHndl = LOAD_DLL('C:\TAS2XLS.dll')
// dHndl = LOAD_DLL('C:\TAS2EXCEL.dll')

//.XLS
filename = 'C:\test.xls'
size1=SIZE(filename,'a')

numInt = DLLFC(dHndl,'FileNew','I', CREC(size1), TRIM(filename))
msg numInt
```

See also:
FileOpen, FileSave, FileClose

4.35 TAS Professional EXAMPLE

TAS Professional EXAMPLE


```
//Example
#WinForm Example
```

```
(* ---------------------------------------------------------------------------
   Events for: Example Object Type: TTASForm
   --------------------------------------------------------------------------- *)

  Example.OnOpenFiles:
    Ret

  Example.OnStart:
    Ret

  Example.OnClose:
    Ret True

 (* --------------------------------------------------------------------------- *)
  Go.click:

  define dHndl,numInt type I
  define filename type A size 256
  define size2 type R
  define sheet, row, column type R
  define myBool type L
  define mynumeric type N dec 4

  dHndl = LOAD_DLL('C:\TAS2XLS.dll')

  numInt = DLLFC(dHndl,'LanguageSet','I', CREC(1))

  filename = 'C:\test.xls'
  size2=SIZE(filename,'a')
  numInt = DLLFC(dHndl,'FileNew','I', CREC(size2), TRIM(filename))

  size2=SIZE(filename,'a')
  numInt= DLLFC(dHndl,'FileOpen','I',CREC(size2),TRIM(filename))

  if numInt=0
    sheet = 1
    column =  3
    row = 5
    mynumeric = 100
    numInt = DLLFC(dHndl,'CellWriteDouble','I', CREC(sheet), CREC(column), CREC(row),
             CFLT(mynumeric))

    sheet = 1
    column =  3
    row = 6
    mynumeric = 1000
    numInt = DLLFC(dHndl,'CellWriteDouble','I', CREC(sheet), CREC(column), CREC(row),
             CFLT(mynumeric))

    sheet = 1
    column =  3
    row = 7
```

```
    mynumeric = 500.625
    numInt = DLLFC(dHndl,'CellWriteDouble','I', CREC(sheet), CREC(column), CREC(row),
             CFLT(mynumeric))

    numInt= DLLFC(dHndl,'FileSave','I')
    numInt= DLLFC(dHndl,'FileClose','I')
  endif

  myBool = REMOVE_DLL(dHndl)
  msg 'Finished'
  ret
(* ------------------------------------------------------------------------- *)
```

Index

- T -

- V -

- W -