TranAX 4

Report-Generator

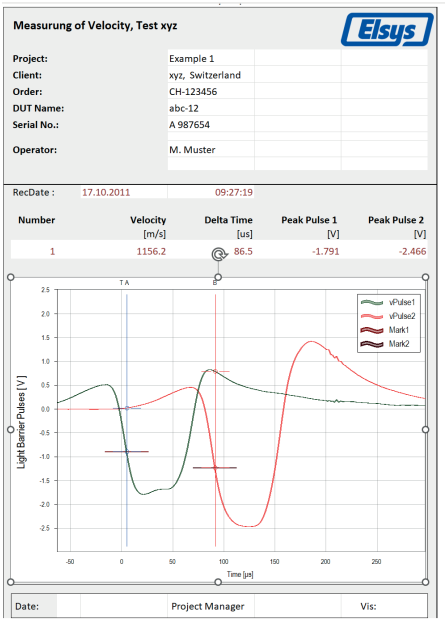
Elsys

User Manual

1. TranAX Report Generator

The report generator is based on special functions in the formula editor and allows you to create a measurement report based on Excel file.

These functions are only available if the report generator option is activated.



1.1 Introduction

The basis of a report is an existing Excel file which is used as a template. Formatting such as font sizes, cell colors or cell borders are defined in this template in Excel. Using the formula editor, this template can be filled with measurement curves or scalar values.

he Excel file can be accessed in two different ways: If Excel is on the system, the Excel file can be accessed via the COM interface. This has the advantage that the report is opened immediately and the inserted data is immediately visible. The second option accesses the file directly. No installed Excel is required for this.

Either Excel files in .xlsx or .xltx format can be used as template files. It is recommended to use the xltx files as templates, as these cannot be changed so quickly by mistake.

1.2 Report Functions

1.2.1 Initialization and file functions

XlInitializeCOM() XlInitializeXML()	One of the two commands must be executed before other report generator functions can be called. XlInitializeCOM() establishes a connection to Excel via the COM interface. (Excel must be installed on the device) XlInitializeXML() establishes a connection directly via the file. No Excel is required.

XlCreateEmptyFile()	Creates an empty Excel file and returns a random name with the extension ".xlsx" as the return value. This name must be used to be able to work on the file. By default, this file is assigned the path "data" of the current experiment.
	
	This function is recommended if calculated values are to be saved quickly in an Excel file. It is not based on a previously created template, but can be subsequently processed with Excel.
	
	If this function is called, XlOpenFile () can be omitted.
	
	Example:
	`fn  = XlCreateEmptyFile()`
	`fn2 = "MyRreport_#.xlsx" ; any own name`
	`XlSetCellValue(fn, "A10", 7.123)`
	`XlSave(fn, fn2) ; fn2 will be saved in «.\data»`

XlOpenFile (fname)	Opens an existing Excel file with the name "fname".
	The return value is again the file name supplemented with the absolute file path. This must be used in subsequent Excel functions.
	
	Example:
	`XlInitializeXML()`
	`fn = ".\Excel_Templates\MyTemplate.xlsx"`
	`filename = XlOpenFile(fn)`
	`XlSetCellValue(filename, "A10", 7)`

XlSave (fname, fnameReport)	Saves the open Excel file “fname” including all changes.
	If the template is not to be overwritten, a different name (including path) must be selected for the second parameter “fnameReport”.
	As long as no CloseFile(fname) command is executed afterwards, the Excel file remains open. Additional entries could therefore be made immediately.
	
	Example:
	`XlInitializeCOM()`
	`fn = ".\Excel_Templates\MyTemplate.xlsx"`
	`fnam = XlOpenFile(fn)`
	`fnRep = .\Excel_Reports\Report_#.xlsx"`
	`; Block with value insertions in EXCEL Sheet.`
	`XlSave (fn, fnRep) ; Saves the EXCEL file (incl. all insertions) in the Report File.`

1.2.2 Excel-Sheet Functions

XlCreateSheet(fname [,name [,templateName]])	Creates a new sheet within an Excel file. The parameter “fname” of type string is the currently used name of the file. The first optional parameter “name” of type String defines the name of the new sheet. Without this parameter, the new sheet is given the name “Empty” plus an ascending number, e.g. “Empty 5” The second optional parameter “templateName” of type String can be used to copy an existing sheet within the Excel file and create it with a new name.

XlSelectSheet (fname, sheet)	In the Excel file “fname”, select the sheet with the name “sheet”. If this command is missing, the first sheet is selected by default.
	
	Example:
	`fn = ".\Excel_Templates\MyTemplate.xlsx"`
	`filename = XlOpenFile(fn)`
	`XlSelectSheet(filename, "table2")`

XlRemoveSheet(fname, sheetname)	Removes the sheet “sheetname” from the file.
XlRenameSheet(fname, sheetname, newSheetname)	Renames an Excel sheet.

1.2.3 Cell Functions

XlGetCellValue (fname, cell)	Returns the value in the “cell” cell. As a rule, “cell” is a single cell (“B3”). A string is expected as “cell”. Depending on what is entered in the cell, a string or a number is returned.
	
	A cell range can also be defined (e.g. “A1:A4” or “A1:D1”). The cell range may only be defined in the same column or row (not e.g. “A1:C3”). A corresponding array or list must also have been declared beforehand. The array or list is then filled with the string or number values of the cells.
	Arrays must be declared as string arrays or double arrays, depending on whether strings or numbers are expected in the cells.
	
	If a cell is empty, an empty string or “n. def.” is returned. If a list has been declared, its elements then contain strings or numbers
	
	Example:
	`fn = ".\Excel_Templates\Template_1.xlsx"`
	`fnam = XlOpenFile(fn)`
	
	`v1 = XlGetCellValue(fnam, "A1") ; single value`
	
	`valArr = Array(0 to 3) as String`
	`valArr = XlGetCellValue(fnam, "A2:A5")`
	
	`valArrD = Array(0 to 3) as Double`
	`valArrD = XlGetCellValue(fnam, "B2:E2")`
	
	`valArrL = List()`
	`valArrL = XlGetCellValue(fnam, "A2:E2")`

XlGetCellReferences (fname, searchKey [, all])	Keywords must have been entered in the Excel sheet (template) beforehand. If "keyword" is not found, the function returns "n. def.".
	It is advisable to select keywords that are otherwise rarely used (e.g. "%V1"). No distinction is made between lower and upper case letters.
	If the optional parameter all = True is specified, an array or a list is returned. This contains all cell names for which the keyword searchKey was found.
	
	Example:
	`fn = ".\Excel_Templates\MyTemplate.xlsx"`
	`fnam = XlOpenFile(fn)`
	`cellRef = XlGetCellReferences(fnam, "%x")`

XlChangeColumn (cell, number)	Adds the specified number "number" to the defined cell "cell". The column is influenced by "cell".
	"cell" must be a string (e.g. "B5") and "number" a number.
	
	Example:
	`cell = "B2"`
	`newCell = XlChangeColumn(cell, 2) ; result is «D2»`

XlChangeRow (cell, number)	Adds the specified number "number" to the defined cell "cell". The row is influenced by "cell".
	"cell" must be a string (e.g. "B5") and "number" a number.
	
	Example:
	`cell = "B2"`
	`newCell = XlChangeRow(cell, 2) ; result is «B4»`

XlSetCellValue (fname, cell | searchKey, value)	In the Excel sheet “fname”, enter the value “value” in the cell “cell”. Instead of a cell (e.g. “B2”), a key “searchKey” can also be specified (this means that XlGetCellReferences can be omitted).
	“fname” and “cell” or “searchKey” must be strings.
	“value” can be a number, string, array, list or curve.
	If “value” is an array, list or curve, the values are inserted into the column of the specified cell “cell” by default.
	
	If you want to insert the values in a row, the cell must be defined as follows: “A7:row”.
	
	Curves (as well as arrays or lists) must not be too long for transfer to Excel. The maximum length (number of samplings or elements) is:
	Transfer to a column: 1‘048‘577 - Line of the first value (e.g. <=1‘048‘571, if cell = “x6”).
	Transfer to a row: 16‘385 - Column of the first value (e.g. <=16‘382, if cell = “Cx:row”).
	Curves that are too long (large block lengths) can be shortened beforehand with Slice(..), Resampling(..) or Skip(..).
	
	Example:
	`filename = "excel\Template.xlsx"`
	`filename = XlOpenFile(filename)`
	`colArr = Array(1,2,3,4)`
	`XlSetCellValue(filename, "A1", colArr)`
	`XlSetCellValue(filename, "A1:row", colArr)`

XlEnd (fname, cell, direction) *(TranAX 4.3 and newer required)*	Searches for the last non-empty cell from the cell “cell” and in the direction of “direction”. ▪ 0 = up ▪ 1 = down ▪ 2 = left ▪ 3 = right

1.2.4 Image Functions

XlInsertImage (fname, winname,
cell [, width [, height]])

Inserts a screenshot of the “winname” window at the specified position cell in the Excel file “fname”.
“winname” must be a string. This can be the name of a single curve display, a scalar table, an entire page or an image file (including path details).
“cell” is usually a single cell (e.g. “B2”). It determines the top left-hand corner of the image to be inserted.
A range can also be specified for “cell” (e.g. “A4:F30”). The cell range determines the location and size of the inserted image.

The width “width” and height “height” of the inserted image can be defined optionally. If only the width is specified, the height is determined proportionally (same aspect ratio as the original image).
“width” and “height” must be specified in the number of pixels. If a cell range (not just a single cell) is specified, “width” and “height” are ignored.

If “White background” is selected under “Extras / Settings / User interface / Snapshot”, the transferred image always has a corresponding background color. If you are working with a black background on the screen, the curve colors should be darker.

Example:

```
fn = ".\Excel_Templates\MyTemplate.xlsx"
fnam = XlOpenFile(fn)
XlInsertImage(fnam, "Waveform 1", "A4:E20")
XlInsertImage(fnam, "Skalar_A 1", "A21", 100, 300)
```

1.2.5 Header and Footer Functions

XlSetHeader (fname, alignment | searchKey, value)
 	
XlSetFooter (fname, alignment | searchKey, value)
	Inserts the value "value" in the header or footer line of the Excel file "fname".
	The second parameter determines where "value" is to be inserted. This can be a keyword XlLeft, XlCenter, XlRight (for left, centered, right) or a search word. A search word must first be inserted in the header or footer of the Excel template.
	
	Example:
	`Fn = "excel\MyTemplate.xlsx"`
	`Fn = XlOpenFile(Fn)`
	`Skey="%Hd-1" ; Should be set in Template`
	`XlSetHeader(Fn, XlLeft, "Report")`
	`XlSetHeader(Fn, Skey, "Unit Test 1")`
	`XlSetFooter(Fn, XlCenter, "Result Table")`

XlLeft XlCenter XlRight	Keywords for the alignment parameter of the XlSetHeader() and XlSetFooter() functions.