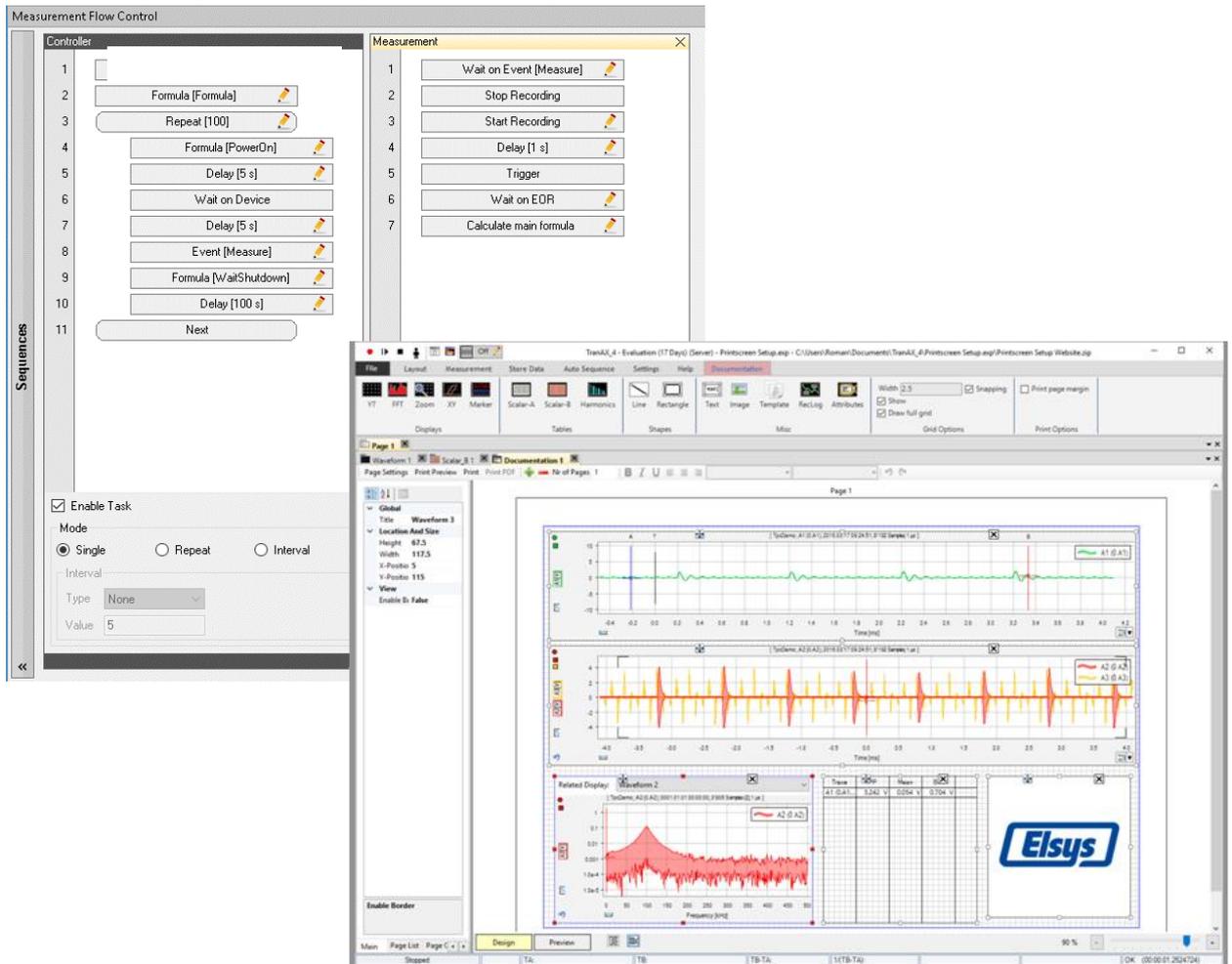


TranAX 4.1

Data Acquisition Application Software



User manual



Elsys AG
Mellingerstrasse 12
CH-5443 Niederrohrdorf

+41 56 496 01 55
www.elsys-instruments.com

Table of contents

1	Introduction and News	9
1.1	Preamble	9
1.2	Full version vs. Light Edition vs. Trial version	10
1.3	System Requirements	10
1.4	What is new in this version?	11
1.4.1	Measurement Flow Control (MFC)	11
1.4.2	Virtual Hardware Manager (VHM)	11
1.4.3	Zoom functions	11
1.4.4	Portable edition	11
1.4.5	Know how WIKI page	12
1.4.6	Other New Features	12
1.5	What was new in previous version?	13
1.5.1	Ribbon Toolbar	13
1.5.2	High Resolution Displays	13
1.5.3	Documentation Window	14
1.5.4	Digital Signals (Marker) Waveform	15
1.5.5	Formula Editor	15
1.5.6	Layout and Hardware Settings	16
1.5.7	Start Screen	16
2	Structure and components.....	17
2.1	Menu Bar	17
2.2	Ribbon Bar.....	17
2.3	Pages und Waveforms	18
2.4	Control Panel	18
2.5	Devices Manager.....	18
3	License handling	19
3.1	Offline activation	20
3.2	Online activation.....	21
3.3	Release license.....	21
3.4	Install software options	22
4	Ribbon Bar items	23
4.1	Ribbon Tab "Layout"	23
4.1.1	Displays.....	23
4.1.2	Documentation & Video.....	23
4.1.3	Tables.....	24
4.1.4	Controls	24

4.1.5	Misc. Controls	24
4.1.6	Formula Editor Controls.....	24
4.2	Ribbon Tab "Measurement"	25
4.2.1	Measurement	25
4.3	Ribbon Tab "Store Data "	26
4.3.1	Save and load data.....	26
4.3.2	Import	26
4.3.3	Export.....	26
4.3.4	Print & Snapshot.....	26
4.4	Ribbon Tab "Measurement Flow Control"	27
4.4.1	Measurement Flow Control.....	27
4.4.2	Edit	28
4.4.3	View	28
4.5	Ribbon Tab "Settings"	28
4.5.1	Settings	28
4.6	Ribbon Tab "Help".....	29
4.6.1	Info.....	29
4.7	Ribbon Tab "Waveform Display"	29
4.7.1	Block & Cursor Functions.....	29
4.7.2	Show / Hide.....	30
4.7.3	Areas / Rulers / Cursors	30
4.8	Ribbon Tab "Scalar Table A"	30
4.8.1	Scalar Function Settings.....	30
4.8.2	Number Formatting	30
4.9	Ribbon Tab "Formula Editor"	31
4.9.1	Calculate	31
4.9.2	Debug.....	31
4.9.3	Controls.....	32
4.9.4	Settings	32
4.9.5	Formula Editor Controls.....	32
5	First Steps.....	33
5.1	Startup Page.....	33
5.2	Device Manager	34
5.3	Simple Experiment: Template "Scope Areas"	35
5.3.1	Start Recording	36
5.3.2	Change Scalar Table	36
5.3.3	Analysis with Formulas	38
5.3.4	Save and load recorded signals	39

5.3.5	Save a section of a trace	40
6	Control Panel.....	41
6.1	Main settings.....	42
6.2	Icons	43
6.3	Control Panel tabs.....	43
7	Virtual Hardware Manager (VHM)	44
7.1	Short description of VHM	44
7.2	Connect to a VHM device	44
7.3	System requirement	44
8	Operation mode	45
8.1	Scope.....	45
8.2	Multi Block	45
8.3	Continuous.....	47
8.4	ECR mode	48
8.4.1	Basic Sequence	49
8.4.2	ECR single channel mode.....	50
8.4.3	ECR multi-channel mode	50
8.4.4	Dual mode	51
8.5	ECR Trigger option	51
8.5.1	Normal	52
8.5.2	Holdoff.....	52
8.5.3	Retrigger	53
9	Input Amplifier	54
9.1	Averaging	54
9.2	Amplifier options	54
9.3	Markers (Digital Inputs)	55
10	Marker	55
11	Trigger.....	56
11.1	Input multiplier	57
11.2	AND Link (logic AND operation).....	58
11.2.1	Example 1: AND-link (Slope and State)	59
11.3	Pre- and Post-Triggering (Trigger Delay).....	60
11.4	Trigger Modes	61
11.4.1	Slope	61
11.4.2	Window.....	61
11.4.3	Pulse > Time.....	61
11.4.4	Pulse < Time.....	61
11.4.5	Period > Time.....	62

11.4.6	Period < Time	62
11.4.7	Slew rate	62
11.4.8	State	63
11.5	Advanced Trigger-Modes (Overview)	63
11.5.1	Pulse inside t1 .. t2	64
11.5.2	Pulse outside t1 .. t2	65
11.5.3	Delay > t	66
11.5.4	Delay < t	67
11.5.5	Delay inside t1 .. t2	68
11.5.6	Delay outside t1 .. t2	69
11.5.7	Period inside t1 .. t2	70
11.5.8	Period outside t1 .. t2	71
11.6	Existing Trigger-Modes for Pulse / Period	72
11.6.1	Pulse > t	72
11.6.2	Pulse < t	72
11.6.3	Period > t	73
11.6.4	Period < t	73
12	Physical Unit	74
12.1	Scale Designer	74
13	Information Window	75
14	Cluster Configuration	76
15	Reference Pointers	77
15.1	Potential Sources for Reference Pointers	77
15.2	Reference-Pointers in the Formula Editor	78
15.3	Reference-Pointers in Autosequence	79
16	Auto Setup	80
17	Waveform Display	83
17.1	Organizing and arranging	83
17.2	Zooming	83
17.3	Moving traces	84
17.4	Set to full scale	84
18	Display options in Y/T Waveforms	85
18.1	Legend	85
18.2	Text Entries in Waveform Display	87
18.3	Grid	87
18.4	Background color	87
18.5	Areas	88
18.6	Y Axes	88

18.6.1	Number of Rulers Left and Right	88
18.6.2	Locking of Y Axes	89
18.6.3	Labelling of Y Axes	89
18.7	Visualization of Traces	89
18.8	Cursor Properties	90
18.9	Cursor on Sample points	91
18.10	On-Curve Measurements	92
18.11	Labels on Curves.....	93
18.12	Adding images and formatting of calculation results	94
18.13	Y-scale adjustments of curves	95
18.14	Snapshot.....	95
18.15	Analysis of Multi Block records (Block Jumping).....	96
18.16	Additional Waveform displays	97
19	XY Waveform	98
20	FFT Waveform	99
20.1	Vertical and Horizontal Scaling	99
20.2	Octave and 1/3 Octave scaling	100
21	Show Videos synchronized to recorded Traces	101
21.1	Video Waveform usage.....	102
22	Documentation Window.....	105
22.1	Place a design item	108
22.2	Page Settings.....	108
22.3	Split Waveform on multiple pages	109
22.4	Templates (Header and Footer).....	109
22.5	Reserved keywords in Text objects	110
22.6	Scalar values in Text objects	110
22.7	Printing (Legacy Mode).....	111
23	Saving records and traces	113
23.1	Save Range	113
23.2	Data reduction	113
23.3	Compression	114
23.4	Included Settings.....	114
23.5	Export.....	114
23.6	Saving pages and waveforms.....	114
23.7	Import Option	116
24	Signal Source Browser	117
24.1	HDF Viewer	118
24.2	Excel Importer.....	118

25	Scalar Functions	119
25.1	Scalar Functions Table A	120
25.1.1	Select a Trace for the Scalar Functions.....	120
25.1.2	Select Scalar Functions	121
25.1.3	Example 1: RMS	122
25.1.4	Example 2 Frequency.....	123
25.1.5	Example 3: Phase	125
25.2	Scalar Functions Table B	127
25.2.1	Add a Scalar Function to the table	128
25.2.2	Example 1: Apparent Power	130
25.2.3	Example 2: Power Factor	131
25.3	FFT Function (Table A/B).....	132
25.4	Additional Cursors.....	133
25.5	Auto-Refresh of the Scalar Function Table	134
25.6	Conditional Background Color	134
26	Harmonics Table	136
26.1	Enhanced Harmonics Table.....	139
27	Formula Editor	141
27.1	Using the Formula Editor	142
27.2	Place Cursors.....	144
27.3	String Variables	144
27.4	Assigning of Sub-Functions	144
27.5	Number format for scalar results	146
27.6	Error Messages.....	146
27.7	Groups of Functions, Overview.....	146
28	Measurement Flow Control (MFC)	147
28.1	MFC Functions.....	147
28.2	MFC Examples.....	150
29	Averaging over multiple recordings	151
29.1	Averaging in the Time Domain.....	151
29.2	Averaging in the Frequency Domain.....	152
30	Experiments und Experiment Sets	153
30.1	Data structure of an Experiment	153
30.2	Write protection for Experiment Sets.....	154
30.3	Password protection for Experiment Sets	154
31	Misc. Controls	155
31.1	Recording Log.....	155
31.2	Attributes	155

31.3	Error Log.....	156
32	Settings	157
32.1	Import/Export	157
32.2	User Interface	158
32.3	Performance	159
32.4	Other Settings	159
32.5	Trace Color Definitions (Menu "Extras")	160
33	SCOPE (Oscilloscope)	161
33.1	Channel settings.....	161
33.2	Buttons for recording commands.....	162
33.3	Time settings.....	162
33.4	Trigger conditions, trigger level	163
33.5	Digital Readout Boxes	164
33.6	Maximum curve display.....	164
34	Group of functions in Formula Editor	166
34.1	Array Functions	166
34.2	Auto Sequence Functions	170
34.3	Base Functions	174
34.4	Channels.....	182
34.5	Enumerable Functions	185
34.6	Exponential and Trigonometric	187
34.7	File Functions	190
34.8	Filter Functions	202
34.9	Layout Waveform	204
34.10	Measurement Flow Control	206
34.11	Misc. Functions.....	208
34.12	PLC Functions	220
34.13	Power Functions.....	227
34.14	Programming Functions	233
34.15	Report Generator	249
34.16	Signal Analysis	254
34.17	Signal Generations	269
34.18	Signal Processing	271
34.19	Spectrum (FFT)	279
35	Scalar Functions Description Table	284
35.1	Group Cursor.....	284
35.2	Group Horizontal	286
35.3	Group Misc.....	293

35.4	Group Periodic	298
35.5	Group Power	299
35.6	Group Vertical	303
36	Miscellaneous.....	310
36.1	ActiveX/COM- Interface	310
36.2	Sync.Clock Out.....	310
36.3	Command line parameter	311
36.4	Create shortcuts.....	313
36.5	Limitations.....	314
36.5.1	Digital inputs (markers)	314
36.5.2	Differential inputs	314
36.5.3	Maximum Sample rate	314
37	Trouble Shooting	315
37.1	TranAX Software version.....	315
37.2	TPC-Server Version.....	315
37.3	Driver and Firmware Version	316
37.3.1	Example Windows 7	317
37.3.2	TraNET FE.....	317
37.4	Error Messages.....	318
37.4.1	TranAX.....	318
37.4.2	TraNET Config Logfile.....	319
37.4.3	TranAX firewall and port settings	319

1 Introduction and News

1.1 Preamble

TranAX is a flexible and powerful tool for utilizing TraNET, TPCX and TPCE instruments for measurement, data acquisition tasks and signal analysis. This manual explains TranAX functions, operating modes and settings. It gives you an overview of the system and its extensive capabilities.

DAQ channels can be configured quick and easy for measuring. All settings can be saved and reloaded. This gives the possibilities to prepare layouts and hardware settings for sharing with other users.

Features

- Fast configuration or settings, **no programming knowledges needed**
- Data visualization in multi-waveform displays
- Unlimited number of channels and waveforms
- X-Y waveform display
- FFT Analysis with different scaling and windowing function
- Multiple cursors, labels and measuring tools per waveform
- Intuitive and fast zoom for detailed analysis in large or long-time measurements
- More than **40 scalar functions** to measure based on any significant waveform parameter on time- or FFT-curves
- Powerful formula editor for more than **100 mathematic functions**, syntax highlighting, for-loops, array calculations, string manipulations, etc.
- Macros for controlling the measuring procedure
- WYSIWYG documentation window
- Data export to **HDF5**, ASCII and customer specific data formats
- Data import ASCII and TPC (TransAS 2) files
- Audio files (*.wav and *.mp3) import and export
- **ActiveX/COM interface** for direct accessing from customer specific software to TranAX

TranAX consists of individual windows. As soon as you move a window, docking guides will pop up and let you drag & drop the moving window on the symbols. Arranging and organizing your workspace hasn't been easier.

In TranAX and thus in this user manual, the term "**Experiment**" is often used. An Experiment actually must be seen **as a project**. All the setting, such as amplifier range, sample rate, channel name, the arrangement of windows, formulas, auto sequences, etc. are stored for a particular measurement project or "Experiment". Signals are usually stored within this, by the Experiment, managed environment.

When new to TranAX, please have a look at the [First Steps](#).

1.2 Full version vs. Light Edition vs. Trial version

In general, there is one software version of TranAX 4.1. After installation, TranAX can be used for 30 days as a full version with all supported features and functions. Afterwards, the functionality will switch back to Light Edition (LE) with limited functionality. To get the full version, the Software has to be activated, either during the Trial period or afterwards.

The Trial version, which can be downloaded at www.elsys-instruments.com, defines the 30 days of full version before switching to the Light Edition

Benefits of the full version:

- Measurement Flow Control (MFC)
- Virtual Hardware Manager (VHM)
- Unlimited number of pages and waveforms
- Fully Formula Editor functionality

1.3 System Requirements

TranAX is running on any common Microsoft Windows based computer system. To get the maximum experience efficiency and workflow, the following requirements have to be fulfilled

- Microsoft Windows operating system, Windows 7 or newer, 32bit or 64bit version
- Installation requires approx. 250MB, make sure there is 1GB free space on the hard disk or SSD drive
- 4GB RAM, please note only a windows system with 64bit supports more RAM
- Screen resolution 1366x768, higher resolution recommended.

1.4 What is new in this version?

TranAX 4.1 has many new features and improvements compared to the previous versions.

1.4.1 Measurement Flow Control (MFC)

The MFC replaces the well-known Autosequences from the previous versions but with many new functionalities. The MFC is a powerful tool for measurement automation. Define once your measurement flow and never lose any important data again.

Compared to the Autosequences the MFC will bring the following enhancements:

- Parallel run of multiple tasks
- Single, repetition or on interval task execution
- Graphical configuration with Drag & Drop elements
- Simple flow programming with if/else statements
- In-Task formula execution
- Executing of tasks triggered by events like "End of Recording", "Trigger" or predefined time period or measurement length
- Graphical visualization of the measurement flow
- Faster execution time
- The MFC can also be used for the automation of 3rd party hardware or software.

1.4.2 Virtual Hardware Manager (VHM)

The VHM is a software layer which is insert between the measurement application like TranAX or BallAX and the physical hardware (TPCServer). Therefor it allows to setup and configure virtual instruments based on Elsys DAQ hardware. When no hardware is available, it gets simulated by the VHM. This allows to prepare any measurement settings in the application without physical hardware attached.

1.4.3 Zoom functions

Zooming in the waveform displays is now improved, well known short cuts like Ctrl and scroll wheel up and down for zoom in and out, Shift and scroll wheel up and down for left and right is now supported.

The two short cuts Shift and Y allows zooming with mouse cursor just in Y axis, Shift and X just for the X axis.

1.4.4 Portable edition

There is a well know TranAX installer and also a portable edition available. In case of technical limitation for the installation of software, the portable edition can be used without administrator privileges on the system.

1.4.5 Know how WIKI page

Elsys has now also a WIKI know how database online. We frequently upload examples and more detailed know how to our products and software. Also, several Examples for Formula Editor can be found here

<http://wiki.elsys-instruments.com/index.php>



1.4.6 Other New Features

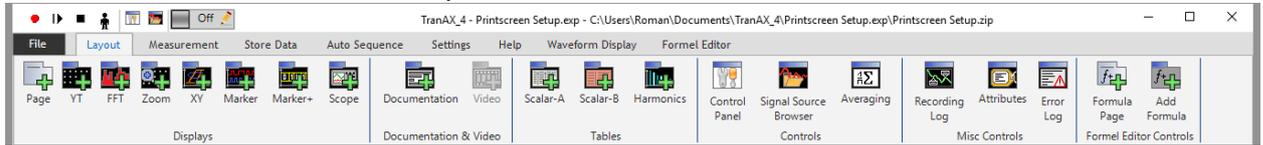
- Speed improvements: Faster Scalar Tables, parallel processing support in formulas for multi core systems.
- New formula functions for faster and straightforward code
- New function for easy calculation with date and time variables

1.5 What was new in previous version?

TranAX 4.0 sets new standards in terms of ease of use, operation and functionality.

1.5.1 Ribbon Toolbar

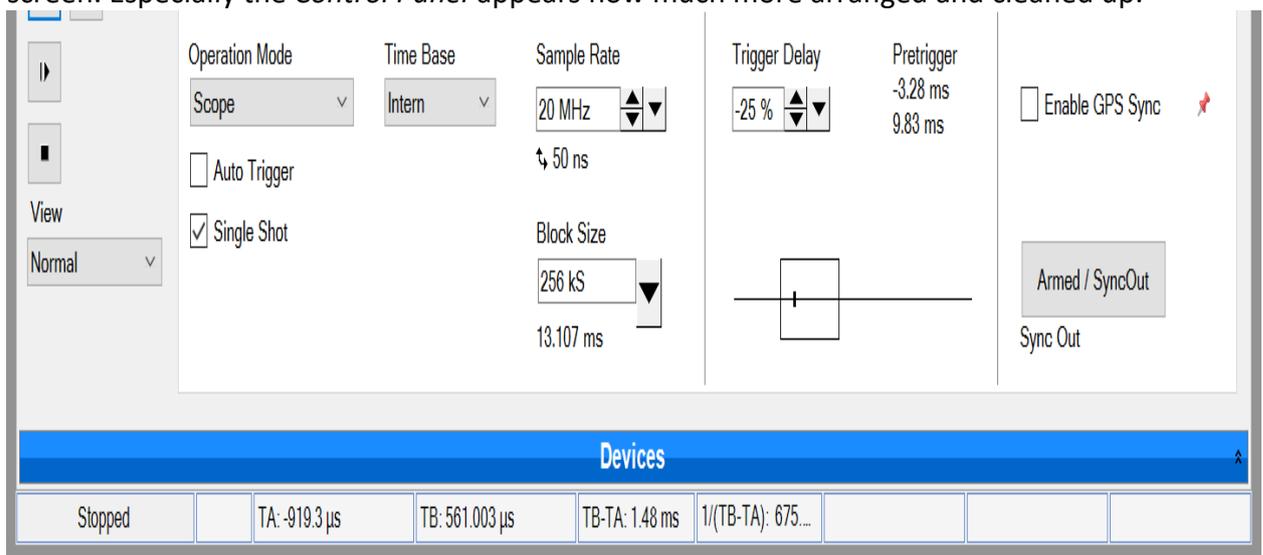
The new Ribbon Toolbar helps handling the software more intuitive. Larger icons with hints are rearranged for a better overview. Depending on user's activity, the toolbar is changing and indicates the relevant functionality.



1.5.2 High Resolution Displays

Screen resolution on new Notebooks and LCD screens are growing continuously. When the resolution goes up and the screen size remains the same, all text elements and menu entries get smaller as the element sizes are defined in pixel. Windows® enables in fact to scale up all text elements (set DPI settings to 120% or higher). But larger text needs more space in comparison to the rest of the elements. Many programs on the market do not handle the extra space needed for the text elements correctly.

In TranAX 4 all elements were revised for displaying different text sizes correctly on every screen. Especially the *Control Panel* appears now much more arranged and cleaned up.

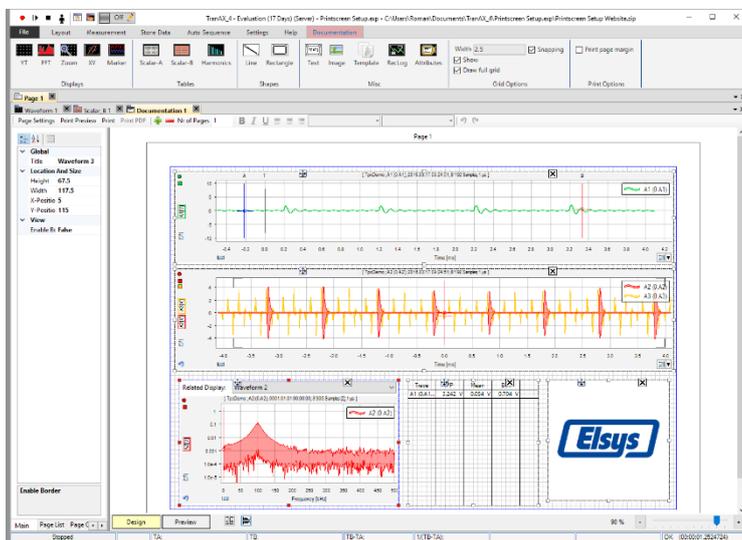


1.5.3 Documentation Window

Printing out a measurement report is still an important need for many customers. In TranAX 3 it was quite tricky to print out the measurement data in a predefined format, because it was dependent on the screen size and the resolution.

Therefore, a WYSIWYG (What You See Is What You Get) Documentation Window was implemented in TranAX 4. It allows to place freely *Waveform Curves*, *Scalar Tables*, *Pictures* and *Text Elements* like on a sheet of paper. In a nutshell some of the new possibilities:

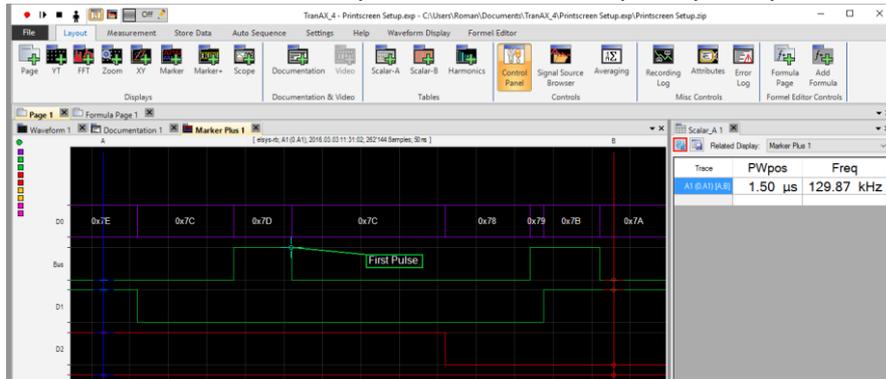
- Design a measurement report in just a few clicks.
- Split long-time measurements on several pages, if needed.
- Use report templates for having consistent looks and CI compliance of your reports.
- No more need for different layout settings for measurement data visualization on the screen and report generation for printing.
- Report results come out from the *Formula Editor* directly in a text field.
- Use the MFC Function "Print" for print out the report after each measurement, or print it in a PDF file.



1.5.4 Digital Signals (Marker) Waveform

For analyzing digital signals – called *Marker* in TranAX – a new marker window is available. In this window, marker signals are handled as individual digital signals like in the normal waveform for the analogue inputs. Some features are:

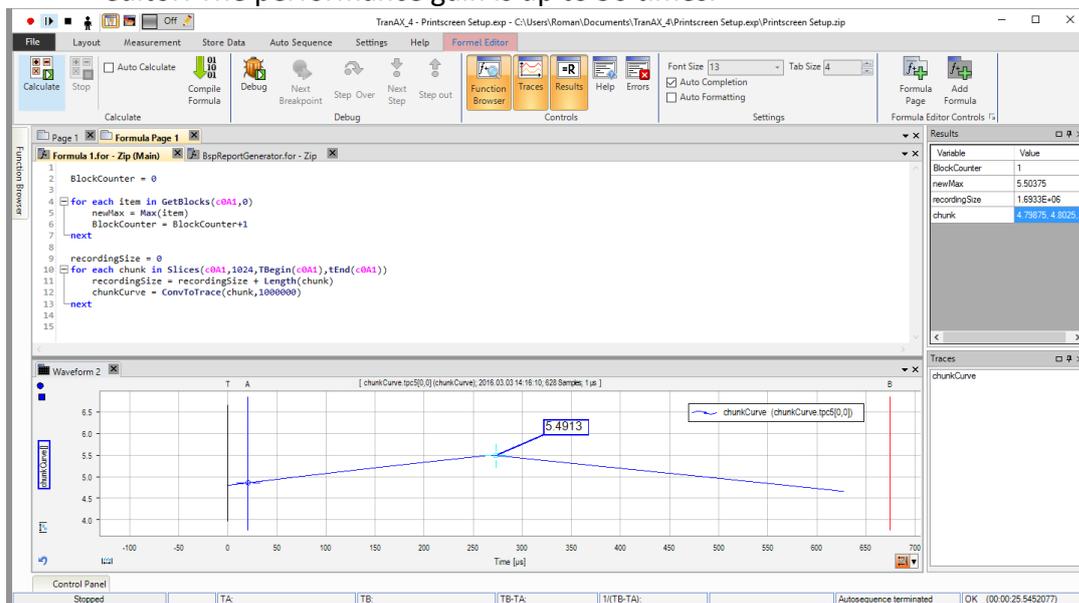
- Arrangement of the markers can be changed independently of the corresponding related analogue channel.
- Different markers can be grouped as bus.
- Numeric interpretation of buses with different display formats and bit ordering
- Scalar function for pulse width and frequency analysis



1.5.5 Formula Editor

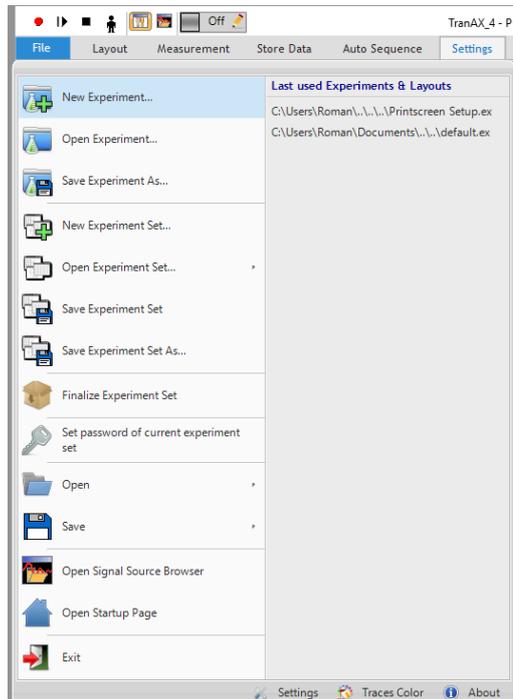
Several new features were added to the *Formula Editor*:

- The *Formula Editor* is now displayed in a tab like a Waveform Window, the same for the result table. This allows placing the results more flexible and closer to other parts in the program.
- New element “for each” allows to iterate over blocks or slices in a much more comfort way. Analysis of large measurement is now much easier.
- Predefined scalar calculations like *Pulse-Width* or *Peak-Peak* from the scalar table are now available as function in the *Formula Editor* too.
- Complex and time-consuming calculation can be accelerated by compiling the formula. This step will convert the formula in a DLL and thus it is no longer interpreted by the editor. The performance gain is up to 50 times.



1.5.6 Layout and Hardware Settings

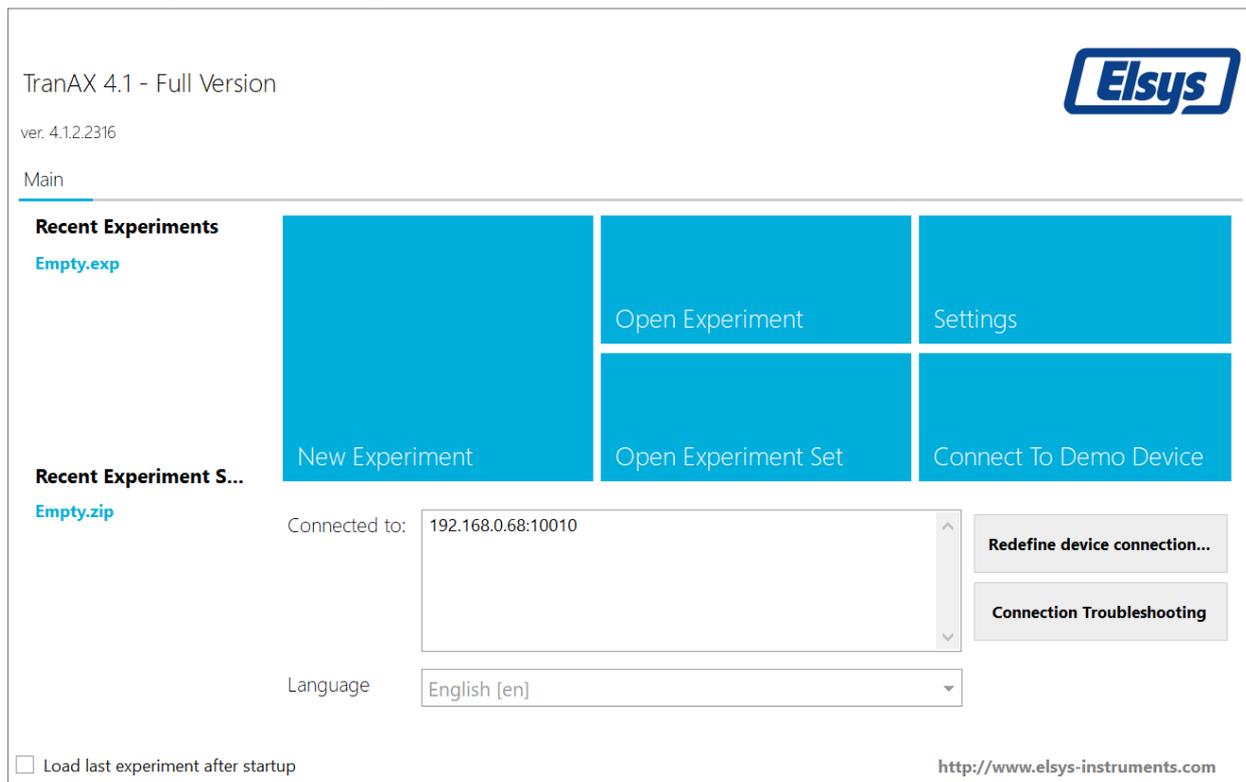
The way to handle settings data has changed lightly in TranAX 4. The role of the *Experiment* is still the same. It is the top-level entity and groups all settings and measurement data in one folder.



An *Experiment* can hold several so-called *Experiment Sets* which are equivalent to the “All Settings”-function in TranAX 3. While, in TranAX 3, all different kind of settings were stored in different files, in TranAX 4 all settings get archived and loaded in and from a ZIP-file. This allows keeping together all needed files for restoring a working set. *Experiment Sets* can be write-protected or even password-protected to avoid unwanted changes.

1.5.7 Start Screen

The new start screen shows a choice of tasks for recent experiments as well as the opportunity, to load the last experiment. This helps to keep the settings clean and organized instead of loading automatically the last setting.



2 Structure and components

TranAX can divide the following main components:

2.1 Menu Bar

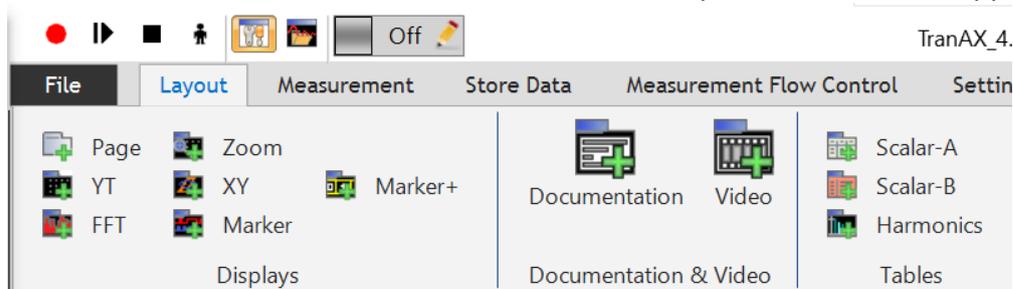
Quick access to the main elements is here possible: Start, manual Trigger and Stop measurement. Key elements such as the [Control-Panel](#) and the [Signal Source Browser](#) can be opened directly.

	Start Recording (F6)
	Manual Trigger (F7)
	Stop Recording (F8)
	Status-Display of the Recording. Additional animated figure, to the status bar in the lower left corner
	Open the Control-Panel
	Open the Signal Source Browser , Access to all Signals, also loaded from files.
	Experiment Sets can be saved read-only, so that they cannot be changed during measurements.

The switch for write protection provides the ability to back up settings, so that these are opened unchanged at the next start of TranAX. This avoids unwanted changes in the settings, especially for batch testing and production usage.

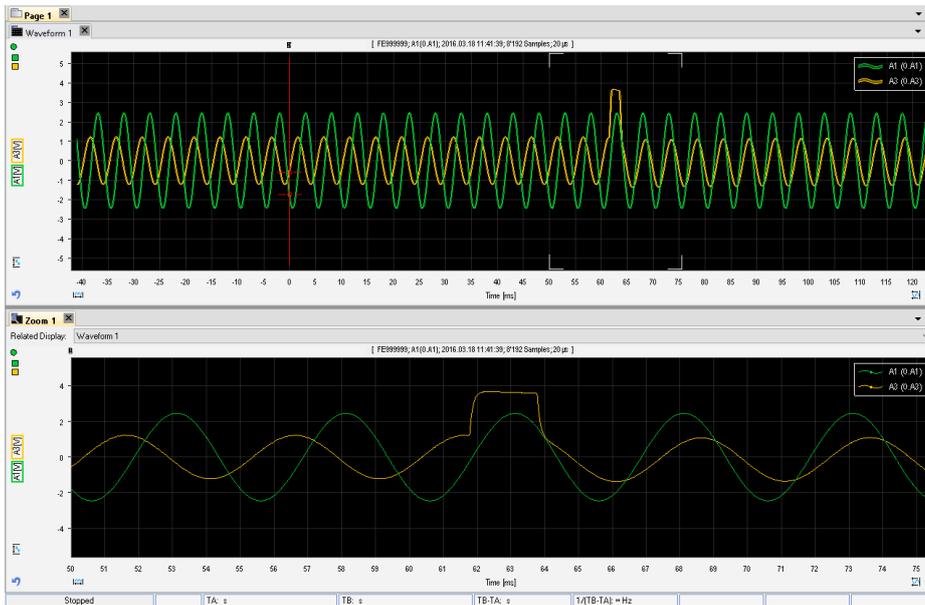
2.2 Ribbon Bar

In the Ribbon Bar, all available functions and settings are listed. According to the selected Waveform or Page, the corresponding available Ribbon tabs are available. The functional usage of these Ribbon Bar is similar to the Microsoft Office products and other applications.



2.3 Pages und Waveforms

On the so-called pages Waveforms (YT, FFT, zoom, etc.) can be placed. The measurement data and curves are displayed on these Waveforms. The Ribbon Tab "Layout", group "Displays" lists all available Waveforms.



2.4 Control Panel

The Control Panel is used to set up and configure the Measurement settings. The combination of the individual measurement modules groups called clusters can be done here. Please note that changes in the Control Panel settings usually only affect the next measurement.

Ch.	Name	Mode	Coupl.	Range	Offset	Input Range	Avg	Filter	Trigger Mode	Link	Level	Hyst.
Device: FE999999 (192.168.0.69:10010) 162.90 GB												
A1	A1	SE	AC 1M	10	50	-5..5 V	14 bit	Off	Off	-	-	-
A2	A2	SE	AC 1M	10	50	-5..5 V	14 bit	Off	Off	-	-	-
A3	A3	SE	AC 1M	10	50	-5..5 V	14 bit	Off	Off	-	-	-
A4	A4	SE	AC 1M	10	50	-5..5 V	14 bit	Off	Off	-	-	-
B1	B1	SE	AC 1M	10	50	-5..5 V	14 bit	Off	Off	-	-	-
B2	B2	SE	AC 1M	10	50	-5..5 V	14 bit	Off	Off	-	-	-
B3	B3	SE	AC 1M	10	50	-5..5 V	14 bit	Off	Off	-	-	-
B4	B4	SE	AC 1M	10	50	-5..5 V	14 bit	Off	Off	-	-	-

Main Input Amplifier Marker Trigger Physical Unit Channel-Description
 Operation Mode: Scope Time Base: Intern Sample Rate: 50 kHz Trigger Delay: -25% Pretrigger: -41 ms
 Auto Trigger Single Shot Block Size: 8 ks
 Enable GPS Sync Armed / SyncOut Sync Out

Stopped TA: 86.785 ms TB: 13.039 ms TB-TA: -73.746 ms 1/(TB-TA): -1

2.5 Devices Manager

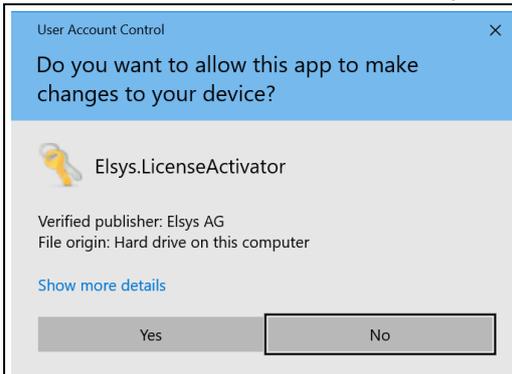
The Devices Manager is used to connect to both, single and groups of multiple TraNET devices. These devices can be handled and configured afterwards in the Control Panel.

Name	Description	Address	Website	Status
<input checked="" type="checkbox"/> TraNETKalib	Kalibrationsplatz	TraNETKalib.local:10010	Website	
<input checked="" type="checkbox"/> MMTranet	TraNet 404 2x4S/80/16	192.168.0.55:10010	Website	
<input checked="" type="checkbox"/> Tpc0	TraNet	192.168.0.63:10010	Website	
<input checked="" type="checkbox"/> elsys-ib-i7	elsys-ib-i7	elsys-ib-i7.local:10010	Website	
<input checked="" type="checkbox"/> FE999999	TraNET FE 204DP-2x4S/80/16, AdvTrg, ECR, Mar	192.168.0.69:10010	Website	Connected
<input checked="" type="checkbox"/> MMElsys	2x4Ch@20MHz@64MS	192.168.0.86:10010	Website	

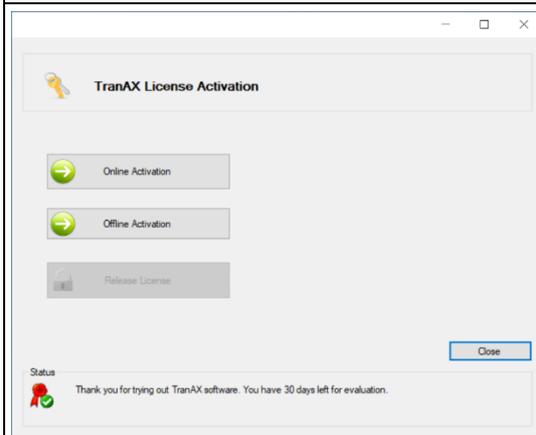
Connect Search IP-Address 127 . 0 . 0 . 1 : 10010 Connect Manually

3 License handling

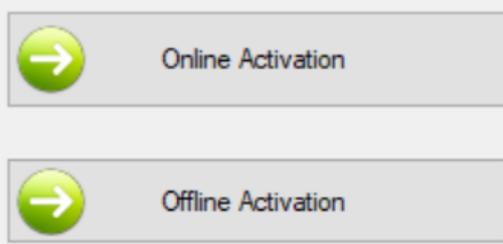
Since TranAX 4.0, a software activation will be required to use the full version of TranAX. Without, the software switches after a trial period of 30 days to the LE (Light Edition) mode.



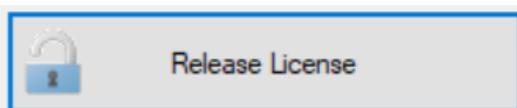
After the first start of TranAX, the Window "TranAX License Activation" appears. If the activation will be done afterwards, administration rights must be granted for this application.



TranAX runs now as a full version with all available features and options, such as "Com / ActiveX", "All Import Options" and "Excel Report" for 30 days. After this time TranAX must be activated, otherwise TranAX will run with reduced functionality as so called TranAX LE (Light Edition).



To use all the implemented functions and features, the software has to be activated. This can be done online and, in case of no internet connection or restrictions, also offline.



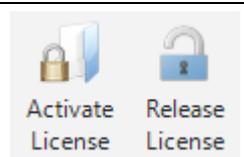
If the activated and currently used license should be ported to another computer (new computer, change the location of the DAQ device to a new department, etc.), the license can be released on the actual system and later activated on the new computer.



TranAX requires an Internet connection only for the first online activation. **For further usage of TranAX, no active Internet connection is required!**



TranAX license can be activated or released at any time in the Ribbon Bar, tab "Settings".



3.1 Offline activation

	Click the button "Offline Activation".
<p>Offline Activation</p> <p>Activation Key: <input type="text"/></p>	Enter your license code from the manual or the USB stick to the field "Activation Key".
<p>Computer ID: <input type="text" value="FF2D55EE2858233021C"/></p>	To generate a valid "Computer Key" the "Computer ID" must be sent to Elsys Instruments for the activation. Please send an Email to info@elsys.ch .
<p>Offline Activation</p> <p>Activation Key: <input type="text"/></p> <p>Computer ID: <input type="text" value="FF2D55EE2858233021C"/></p> <p>Computer Key: <input type="text"/></p> <p><small>Please click onto the QR-Code on the right side which will be copied to the Clipboard. Send the QR-Code to Elsys to receive the 'Computer Key' for Offline Activation. Note: The QR-Code only contains the 'Computer ID' which is necessary for Offline Activation.</small></p> 	It is also possible to just click on the QR code, this will copy the graphics to the Windows clipboard, and can be paste it into an Email.
<p>Computer Key: <input type="text"/></p>	If it is a valid request for a key, the matching computer key will be sent back per Email and has to be entered into the filed "Computer Key".
<p>Status</p>  <p>Your license is invalid</p>	In case of a type error or another reason for a non-valid Activation Key, there is an information in the status visible.
<p>Status</p>  <p>Your license needs activation</p>	If the Activation Key is entered correctly, the Status looks like this.
<p>Activate</p>	Click the button Activate and TranAX will be activated to the full version. Please note there is no need for an internet connection anymore.

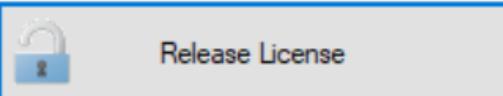


The QR-Code only contains the "Computer ID" which is necessary for Offline Activation.

3.2 Online activation

	<p>Click the button "Online Activation".</p>
<p>Online Activation</p> <p>Activation Key: <input type="text"/></p>	<p>Enter your license code from the manual or from the USB stick to the filed "Activation Key" the and press the button "Activate" TranAX will be activated.</p>
<p>Status</p>  <p>Your license is invalid</p>	<p>In case of a type error or another reason for a non-valid Activation Key, there is an information in the status visible.</p>
<p>Status</p>  <p>Your license needs activation</p>	<p>If the Activation Key is entered correctly, the Status looks like this.</p>
	<p>Click the button Activate and TranAX will be activated to the full version. Please note there is no need for an internet connection anymore.</p>

3.3 Release license

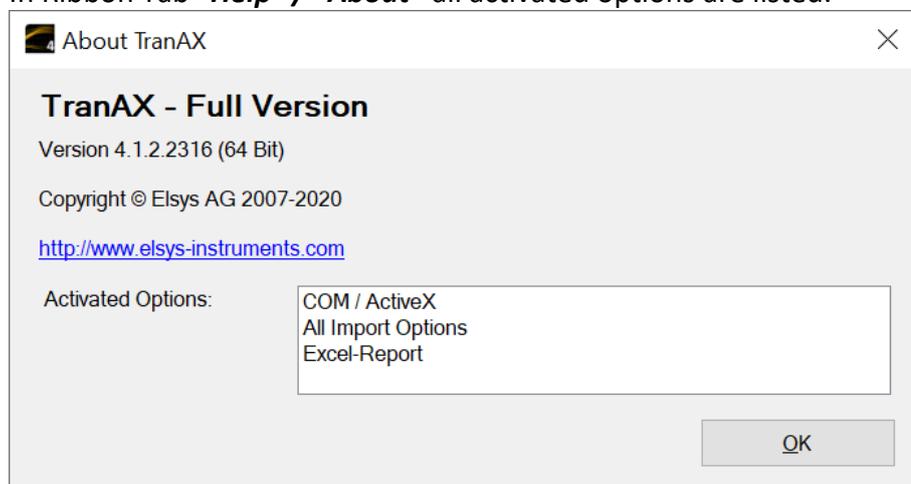
  <p>Release License</p>	<p>If the activated and currently used license should be ported to another computer (new computer, change the location of the DAQ device to a new department, etc.), the license can be released on the actual system and later activated on the new computer.</p>
---	--

3.4 Install software options

Before an option can be used, it needs to be activated. The options are bounded to the license key and are handled through the Elsys license server. In TranAX, several software options are available:

Name	Description
Import	<p>After the Import-Option is installed, the following file types via Ribbon Tab "Store Data" / "Import" can be imported:</p> <ul style="list-style-type: none"> • *.TPC (from TransAS 2) • *.ASD (ASCII from TransAS 2) • *.MP3, *.WAV (Audio-Files) • *.SGY (Segy, geological data format) <p>In addition, with the help of a so-called "Wizard", all kinds of ASCII (Text) files can be imported.</p> <p>At importation the external files are copied into *.TPC5-Format and can as such be used in TranAX then.</p>
ActiveX	Activates the ActiveX interface so that TranAX can be controlled from another program.
Report-Generator	This option enables the creation of Excel reports. After it is installed, the corresponding functions can be accessed with the formula editor

In Ribbon Tab "**Help**" / "**About**" all activated options are listed.



If you like to get an additional option later, please contact your local representative.

4 Ribbon Bar items

The Ribbon Bar consists of individual elements, Tabs and Groups. These are described on the following pages

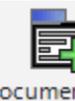
4.1 Ribbon Tab "Layout"

In this Ribbon Tab all the elements are listed that can be placed in TranAX.

4.1.1 Displays

	Add a new Page for several new Waveforms
	Add a new Waveform Display
	Add a new FFT Waveform Display to show the frequency spectrum of signals
	Add a new Zoom Waveform Display to another related display for a zoomed view. This Waveform just shows the selected area from the related Display.
	Add a new XY Waveform Display to show the XY view of signals
	Add a new Marker Waveform Display to show the markers (digital signals) in a separate display
	Enhanced Marker Waveform. This gives the possibilities for grouping digital signals and Markers for analyzing as Bus signals.
	Add a new SCOPE Window

4.1.2 Documentation & Video

	Add a new Documentation window. This can be used to create test reports. Recordings, Waveforms and Tables will be updated at real-time. The printed document locks exactly like the document on the screen.
	Add a new Video window. Video movies can be played in TranAX for- and backwards or simply displayed as a still image. Movies, recorded simultaneously with the captured traces, can be played back simultaneously.

4.1.3 Tables

 Scalar-A	Add a new Scalar Functions Table A . Scalar tables A are used to calculate and read with the same scalar function over multiple traces and channels.
 Scalar-B	Add A new Scalar Functions Table B . This table is suitable for calculation of curve parameters for each channel individually.
 Harmonics	New Harmonics Table determines the fundamental and the harmonics of a periodic signal.

4.1.4 Controls

 Control Panel	Open the Control Panel , in this the recording parameters and settings from each single channel can be configured.
 Signal Source Browser	Open the Signal Source Browser , Access to all Signals: Loaded from files, references, single recording blocks.
 Averaging	Open the Averaging Window, Summation Averaging over 2 - up to multiple records (usable only in Scope Mode). Reduction of noise on periodic signals.

4.1.5 Misc. Controls

 Recording Log	Recording Log : Add event comments to your measurement and get an overview of all occurred trigger events. Add own comments for continuous- or ECR Mode. Entries may also be made afterwards.
 Attributes	Attributes : Add comments and supplementary information to your records. E.g. Test number, conditions, name of participants.
 Error Log	Error Log lists all relevant program operations and errors occurred

4.1.6 Formula Editor Controls

 Formula Page	Opens a new Formula Page .
 Add Formula	Add a new Formula window to the Formula Page to analyze and calculate your acquisitions.

4.2 Ribbon Tab "Measurement"

4.2.1 Measurement

 Start	Start Recording (F6)
 Trigger	Manual Trigger (F7)
 Stop	Stop Recording (F8)
 External	Start a Recording via external TTL-Signal input. For wiring, please see the Hardware User Manual.
 Auto Setup...	Open the Auto Setup dialog. Automatic setup of measuring range and view of Waveform, as a function of the signal-amplitude.
<input type="checkbox"/> Warn unsaved measurements	Warning messages for unsaved recordings can be enabled or disabled.

4.3 Ribbon Tab "Store Data "

4.3.1 Save and load data

	Save Traces as TPC5 file format, equal to HDF5 standard.
Save Tpc5	
	Save Spectrum, TPS5 Format, data needs to be calculated in Spectrum waveform before
Save Tps5	
	Save an entire Page , the sources of the displayed curves are substituted references so that they then show the corresponding curves in the file.
Save Page	
	Load an entire Page
Load Page	

4.3.2 Import

	Opens the file import dialog. Several file formats are supported: *.asd, *.tpc (TransAS2), *.mp3, *.wav, *.sgy
Import Traces	
	Opens the ASCII Wizard for import ASCII files from 3 rd party applications
Ascii Wizard	

4.3.3 Export

	File Export , save Traces (ASCII, Krenz, Segy, DIAdem, TransAS2, Wave). Implementation of additional file formats on request.
Export Traces	
	Export Scalar Table to an ASCII-File (*.txt). Creates a text file from the actual selected Scalar Table. Columns are Tab separated.
Export Scalar Table	

4.3.4 Print & Snapshot

	Print preview , opens the print preview dialog, allows for configuring print layouts.
Print Preview	
	Note: This function is obsolete, please use the Document Window instead. Documentation Window offers more possibilities in terms of visualization and printing!
	Print a page according to the configuration in Print preview.
Print	

Note: This function is obsolete, please use the [Document Window](#) instead. Documentation Window offers more possibilities in terms of visualization and printing!



Snapshot, copy the actual Waveform to the Clipboard. By clicking the mouse on the arrow in the icon, the screen content of all windows in the actual page (not only trace or scalar windows) can be copied to the clipboard.

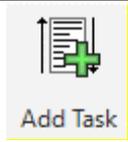
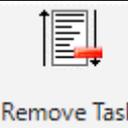
Via menu Ribbon Tab "Settings", Button "Settings", in section "User Interface / Snapshot", various parameters can be set.

4.4 Ribbon Tab "Measurement Flow Control"

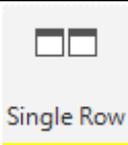
4.4.1 Measurement Flow Control

 Execute	<p>Starts the defined and enabled tasks. You will have to define the tasks according your application. Button F4 has the same definition, starts the MFC as the Auto Sequence in the previous versions.</p>
 Stop	<p>Stops the Measurement Flow and its defined tasks. Button F8 has the same definition.</p>
 Edit...	<p>Opens the Measurement Flow Control window.</p>
<input checked="" type="checkbox"/> Autostart on Experiment Set loading	<p>If enabled, the Measurement Flow Tasks starts at Start-up of TranAX or after loading the Experiment set.</p>
<input checked="" type="checkbox"/> Show current calculating sequence	<p>Visual feedback of the current status of the Tasks. Needs some additional performance, can be used during development of Task, afterward disabled to increase the performance and speed.</p>
<input checked="" type="checkbox"/> Remote Measurement Flow Control	<p>This allows to run Tasks on a TraNET directly without a running TranAX application. Please note that not all functions are supported. This can be used for example autonomous recording and saving of data in a TraNET FE device</p>
 Download External Sequence	<p>Download the Measurement Flow from a TraNET FE device TraNET FE -> TranAX</p>
 Upload External Sequence	<p>Upload the Measurement Flow to a TraNET FE device TranAX -> TraNET FE</p>

4.4.2 Edit

 <p>Add Task</p>	<p>Adds a new Task to the Measurement Flow. Each Task can run independent from each other or connected through events. Also, Formulas will be handled independent in each Task. To Transfer results from one Task to another, the known functions WriteforNext and ReadofPrevious will be used.</p>
 <p>Remove Task</p>	<p>Removes the selected Task. Please note that "Task 1" is always present and cannot be deleted. Renaming is possible.</p>
 <p>Remove Sequence</p>	<p>Removes the selected items in a Task, alternatively, the button delete can be pressed</p>
 <p>Clear Sequences</p>	<p>Clears an entire Task. The same as select all items in a Task and pressing the button delete</p>

4.4.3 View

 <p>Single Row</p>	<p>One row of Tasks, each Task will be next to another, from left to right</p>
 <p>Two Rows</p>	<p>Two rows of Task, numeration will from top to bottom and form left to right.</p>

4.5 Ribbon Tab "Settings"

4.5.1 Settings

 <p>Settings</p>	<p>Opens the settings dialog for configuration and performance optimization of TranAX.</p>
 <p>Trace Colors</p>	<p>Opens the "Trace Color Definitions" dialog. This defines a specific color for each hardware channel. In a given Experiment, the channels have the same corresponding colors for every waveform.</p>
 <p>Custom Toolbar</p>	<p>Opens a dialog to configure a customized Toolbar. On a separated Ribbon Tab, the customized buttons will be available. Can be used for starting 3rd party applications.</p>
 <p>Activate License</p>	<p>Opens the license dialog for online or offline activate of TranAX.</p>

 Release License	In case TranAX will be used on several computers, this button can be used to release an active TranAX license for usage on another computer. Internet connection is recommended; else you will have to contact your local distribution partner.
 Get Update	Downloads the latest released version of TranAX
 Bug Report	Opens the Bug Report dialog. This tool can be used to report malfunctions and bugs to Elsys Instruments.

4.6 Ribbon Tab "Help"

4.6.1 Info

 Content	Opens the user manual as a PDF document. In case multiple documents are installed, an overview to select opens first.
 Shortcuts	List of short cuts for TranAX
 About	Shows a dialog with TranAX version and installed options.

4.7 Ribbon Tab "Waveform Display"

The Ribbon Tab "Waveform Display" will appear, when a Waveform is selected.

Waveform Display

4.7.1 Block & Cursor Functions

 Previous	Previous Block , time window moves to previous block. Mostly used for Multi block- and ECR-recordings.
 Next	Next Block , time window moves to next block. Mostly used for Multi block- and ECR-recordings.
 Lock on screen	Lock Cursors on display. Cursors are locked to display, also during zooming and moving of curves, i.e. are not locked to the Traces!
 Lock Timewindow	Lock Time Window. Time window marker on the main waveform, are locked to display, also during zooming and moving of curves, i.e. are not locked to Traces!

 Fit Timewindow	Fit Time Window to Block. Set time window marker at the border of a block.
 Fit Block to screen	Start and stop of a recorded block will be fit into the time borders of the waveform.
 Move cursor simultaneously	Move Cursors A and B simultaneously. Cursors (normally A and B) will be moved together.

4.7.2 Show / Hide

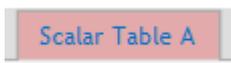
<input checked="" type="checkbox"/> Cursors	Cursors visible / hidden
<input type="checkbox"/> Horizontal Cursors	Horizontal Cursors visible / hidden
<input checked="" type="checkbox"/> Legend	Legend visible / hidden
<input checked="" type="checkbox"/> Grid	Waveform grids visible / hidden
<input type="checkbox"/> Block Numbers	Block numbers visible / hidden
<input checked="" type="checkbox"/> Trigger Lines	Trigger lines visible / hidden

4.7.3 Areas / Rulers / Cursors

Areas	<input type="text" value="1"/>	Number of Areas
Rulers Left	<input type="text" value="1"/>	Number of Y-axis on the left side
Rulers Right	<input type="text" value="0"/>	Number of Y-axis on the right side
Cursors	<input type="text" value="0"/>	Number of additional Cursors (C-Z)

4.8 Ribbon Tab "Scalar Table A"

The Ribbon Tab "Scalar Table A" will appear, when a Scalar Table A is selected.



4.8.1 Scalar Function Settings

Function	<input type="text" value="Cursor Amplitude"/>	Selection of the scalar function
Waveform	<input type="text" value="A2 (0.A2)"/>	Selection of the reference trace, e.g. for scalar function "Delta".

4.8.2 Number Formatting

Formatting	<input type="text" value="None"/>	Number formatting: None, Engineering, Fixed-Point, etc.
No. Digits	<input type="text" value="3"/>	Number of digits

4.9 Ribbon Tab "Formula Editor"

This Ribbon Tab appears, when a Formula page is selected.



4.9.1 Calculate

 Calculate	<p>With the Calculate button, calculations can be started manually. Calculation can also be initiated by the command "Calculate" in an auto sequence. Normally calculation can also be started by the key F10.</p>
 Stop	<p>By pressing the Stop button, a running calculation will be cancelled.</p>
<input type="checkbox"/> Auto Calculate	<p>When the option auto calculate is checked, all calculations are performed immediately after recording of a signal.</p>
Runtime <input type="text" value="2.0"/>	<p>Select the Runtime environment. For existing a perfectly running formulas, we recommend to keep it as configured (also if on version 1.0). For new Formula Editor applications, please select the latest one.</p>
 Compile Formula	<p>Creates a DLL file out of the formula code. This can, depending on the code increase the calculations more than factor 50. This is also usefully for code protection; no plain text of the source code will be visible.</p>

4.9.2 Debug

 Debug	<p>The Button "Start Debug" starts to calculate the formula until the first break point. The calculation will stop at this point and the just calculated values (traces and results) can be analyzed.</p>
 Next Breakpoint	<p>Clicking the red bullet button with the blue arrow "Next Breakpoint" will execute the rest of the code until the next breakpoint or until the end.</p>
 Step Over	<p>"Step Over": As with the button "Next Step" calculating a program line will be performed, whereby however sub functions are bypassed.</p>
 Next Step	<p>Click the arrow down icon "Next Step" to go to next line in the formula.</p>
 Step out	<p>The button with the arrow up symbol "Step Out" will finish the calculation of a loop function (for, loop etc.) and will stop at the next line after the loop has been completed.</p>

4.9.3 Controls

 Function Browser	Displays the "Function Browser". All available functions, instructions and channels can be selected in this window.
 Traces	Displays the window "Traces". The calculated signal curves are listed in this window. These curves may be placed via Drag & Drop into a waveform window.
 Results	Displays the window "Results". The calculated scalar values (numbers, no signal curves) are listed in this window.
 Help	Displays the window "Help". This window displays a brief description for each selected function in the "Function Browser".
 Errors	Additional window with debug outputs and error messages from calculated formulas.

4.9.4 Settings

Font Size <input type="text" value="13"/>	The font size in the window "Formula" can be set individually and will be stored with the layout settings.
Tab Size <input type="text" value="4"/>	Number of spaces that will be inserted while pressing the Tab key.
<input checked="" type="checkbox"/> Auto Completion	When this checkbox is selected, suggestions for auto completion will appear as you type the formula. Using the arrow keys up and down, the appropriate proposal can be selected and adopted by pressing the "TAB" or space key. By pressing "ESC", auto competitions will be cancelled.
<input type="checkbox"/> Auto Formatting	Rearranges the written code according "Tab Size". Nested for loops will be reformatted correctly.

4.9.5 Formula Editor Controls

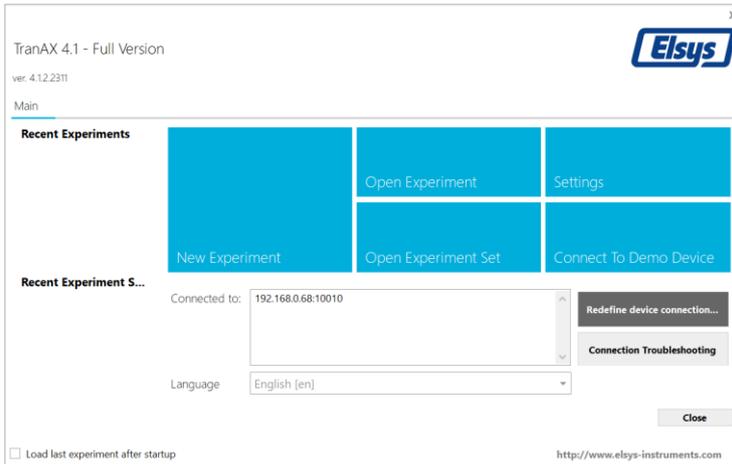
 Formula Page	Opens a new Formula Page .
 Add Formula	Add a new Formula window to the Formula Page to analyze and calculate your acquisitions.

5 First Steps

This chapter is an introduction to the almost unlimited record and analysis features of TranAX. For applications where these special features are not required, we direct you to the so called [SCOPE](#)-application.

5.1 Startup Page

After the first Start-up of TranAX, the start page appears. On its left side is a list with recently opened Experiments and Experiment sets. The buttons on the right side give the options to create or open a new Experiment or Experiment Sets. The button "**Connect To Demo Device**" is a useful option to simulate some Hardware channels of no real Hardware is attached. The startup page appears on every startup of TranAX.

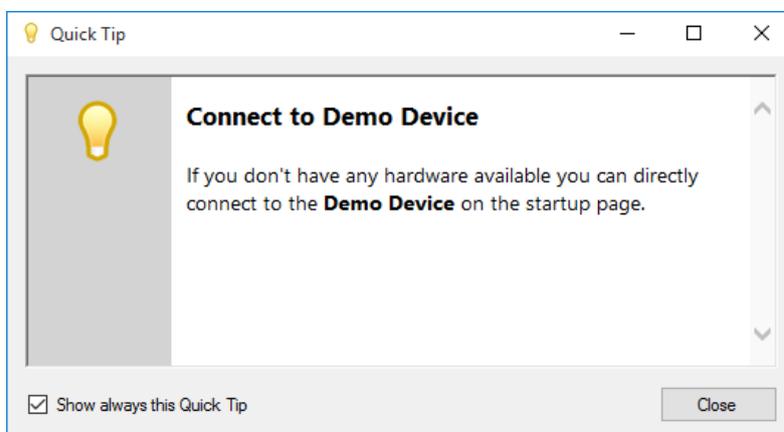


Alternatively, TranAX can also be used in legacy mode; the last Experiment and Settings will be reloaded after start of TranAX. Please uncheck the checkbox "Load last experiment after startup". Click "File / Open Startup Page" to open this dialog anytime again.



If there are no real hardware channels installed click "**Connect To Demo Device**", this will start a DAQ system with simulated traces and signals. To use these traces, click on the Control Panel on Devices and connect to the local device.

During start-up of TranAX, a Quick Tip dialog will be opened, which gives useful information and possibilities for using TranAX. Quick Tips can be enabled or disabled any time.



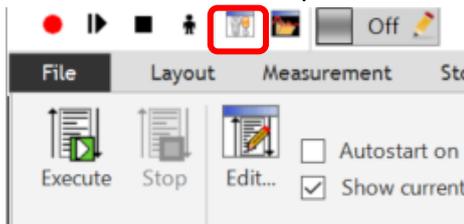
As a next step, you can either create a new experiment or select and open an existing one. There are already predefined templates available. These templates will be installed with the official TranAX installer setup package.



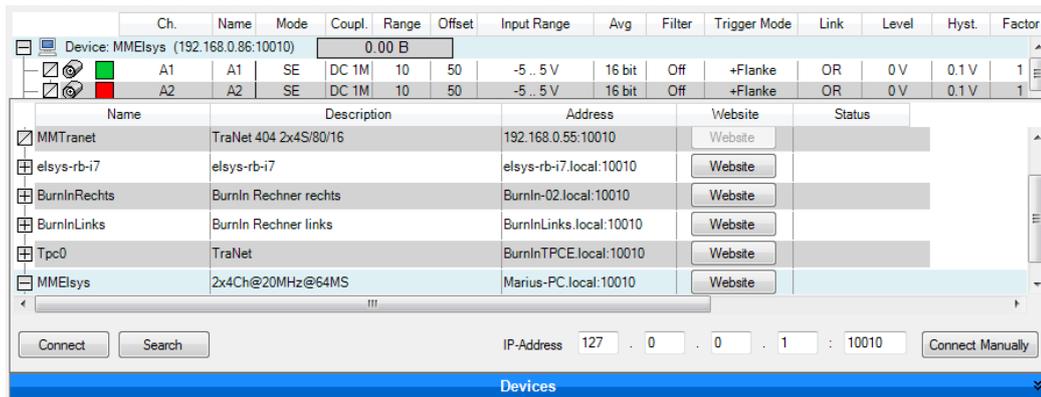
To open an existing TranAX 3 experiment, the open file dialog has to be changed the file extension from *.exp to *.lay or *.all.

5.2 Device Manager

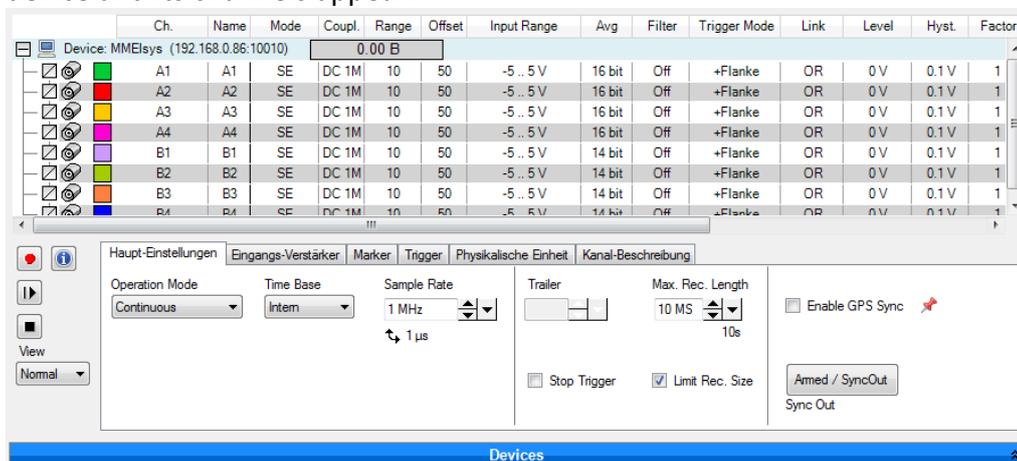
To use the installed TPCX / TPCE modules, or external devices such as TraNET FE, TranAX has to be connected to this device. Since the entire communication is based on TCP / IP, devices are addressed with its IP address and the corresponding port. After opening an Experiment, the Control Panel can be opened:



In Control Panel, all connected devices are listed. To connect to a device, click at the bottom on the expander "Devices". The Device Manager appears in an overlapping window. All available devices in the network are listed in this Devices list.



Selected with the mouse the device and click the button "Connect". In the Control Panel the device and its channels appear:

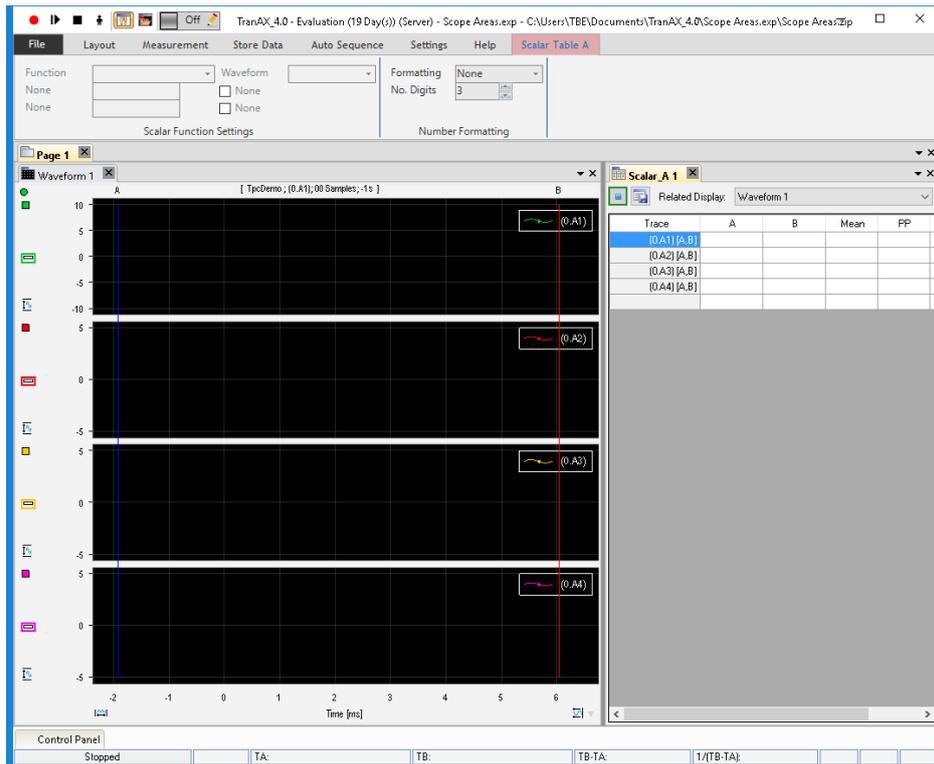


5.3 Simple Experiment: Template "Scope Areas"

Click on the Startup Page on "New Experiment" and load the experiment Template "Scope Areas". The name and the directory for the new experiment is automatically suggested and can be changed before creating. Experiments can be renamed and moved anytime, as long they are not opened in TranAX



Templates for TranAX are filed in the directory "C:\ProgramData\Elsys\TranAX_4.0\ExperimentTemplates". These templates can be used for own experiments. New templates (Experiment Sets) can be copied into this directory and later used again.



Next, you have to connect a device or installed DAQ module in "Control Panel" , button "Devices".



If no hardware is installed, you can also connect directly to the "Demo device" on the "Startup Page". If this button is highlighted in orange, the simulated hardware is ready and connected.

Demo Device Connected

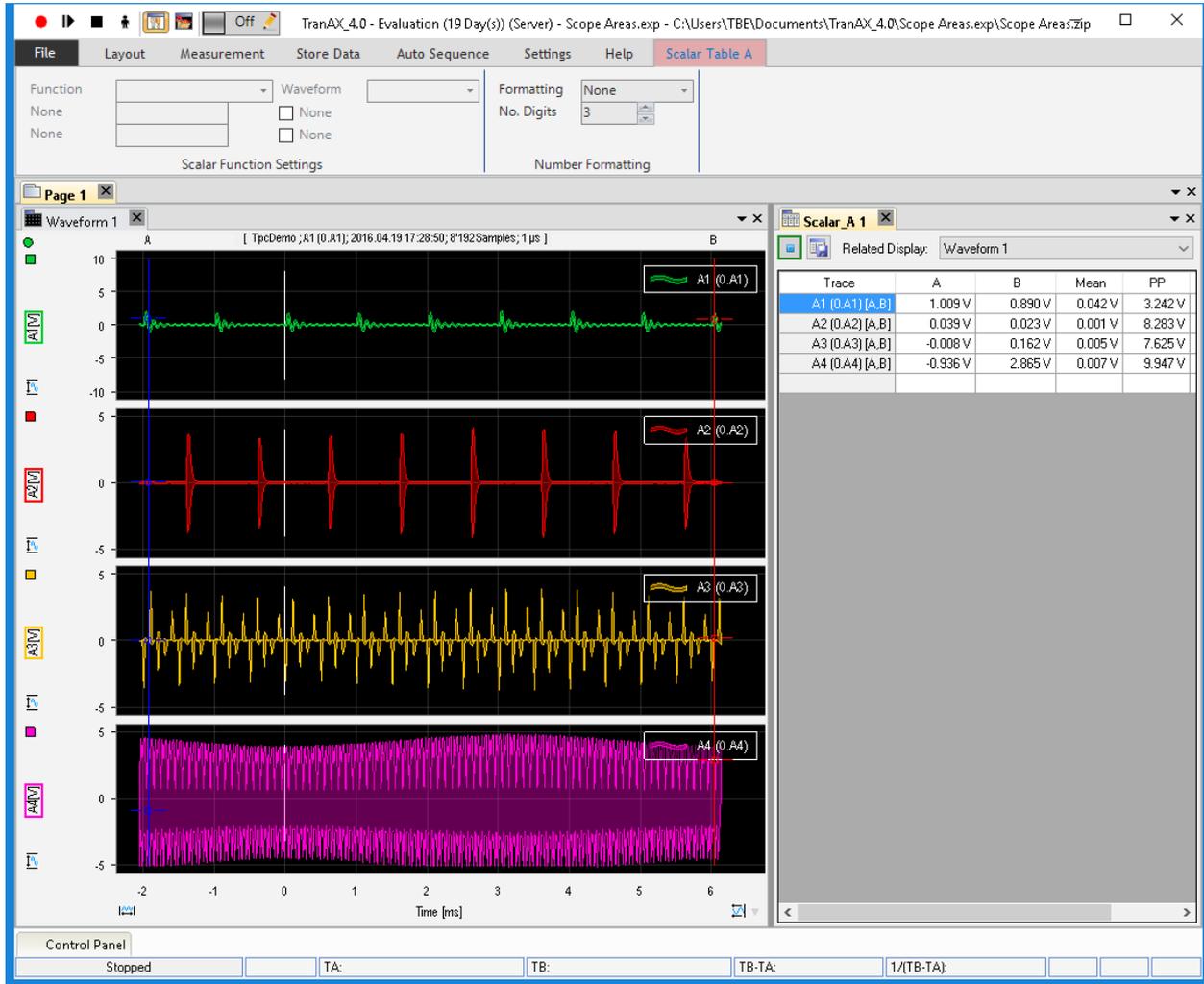


The Demo Server will not be listed as a Devices! If you would like to connect manually, please note that the port will be 10030.

Connected to: 127.0.0.1:10030 (Demo Device)

5.3.1 Start Recording

To start a recording, click the **start button**  or press F6. The Waveform window shows the recorded traces, the Scalar Table on the right side calculates the values according the position of Cursor A and B.



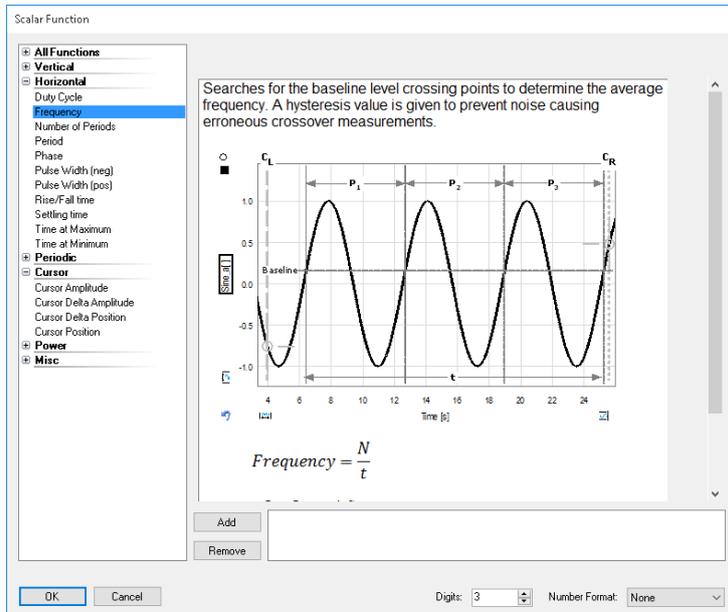
To stop a recording, press the Stop Button  or F8. More options and possibilities regarding recording settings and settings of each single channel can be found in the Control Panel.

5.3.2 Change Scalar Table

If other values should be calculated the scalar table, you can either insert a new column or modify the settings of an existing one. Double-click on the top field of the column whose properties can be adjusted.

Instead of values at position of Cursor A, the frequency should be calculated. Double-click the row A and the scalar function dialog opens.

Trace	A	B	Mean	PP
A1 [0.A1] [A,B]	1.009 V	0.890 V	0.042 V	3.242 V
A2 [0.A2] [A,B]	0.039 V	0.023 V	0.001 V	8.283 V



Select "Frequency" from the horizontal functions and click the button "OK" on the bottom left side.

Afterward, the measured and calculated Frequencies of the signal will be listed in the Scalar Table A.

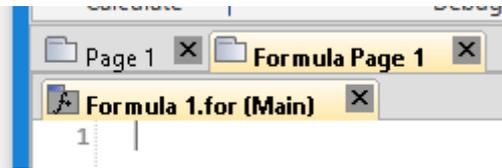
Trace	Freq	B	Mean	PP
A1 [0.A1] [A,B]	8.027 kHz	0.890 V	0.042 V	3.242 V
A2 [0.A2] [A,B]	7.921 kHz	0.023 V	0.001 V	8.283 V
A3 [0.A3] [A,B]	6.016 kHz	0.162 V	0.005 V	7.625 V
A4 [0.A4] [A,B]	60.000 kHz	2.865 V	0.007 V	9.947 V



The analyzed area can be adjusted by moving the Cursors (mostly A and B).

5.3.3 Analysis with Formulas

Complex analysis of the curves can be made by using the Formula Editor. In the following example, a section of a curve will be cut out, filtered and displayed again in the waveform. In addition, a scalar value "Max" will be calculated in the formula.



In Ribbon Tab "Layout", group "Formula Editor Controls" click the button "Formula Page" and a new formula window opens.

Formula for this example:

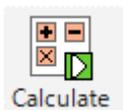
```
; Don't create files for
; calculated traces
; Will be faster!
UseMemory(True)
```

```
; Copy channel A1 to the variable "traceOrig"
traceOrig = c0A1
```

```
; --- slice out section between cursors ---
tA = GetCrs("Waveform 1", "A")
tB = GetCrs("Waveform 1", "B")
traceSlice = Slice(traceOrig, tA, tB)
```

```
; --- filter sliced trace ---
; Lowpass filter:
; Bessel, 6th order, cutoff frequency at 10kHz
traceFilter = LowPass(traceSlice, Bessel, 6, 10E3)
```

```
; --- calculate some scalar values---
maxOrig = Max(traceOrig)
maxSlice = Max(traceSlice)
maxFilter = Max(traceFilter)
```

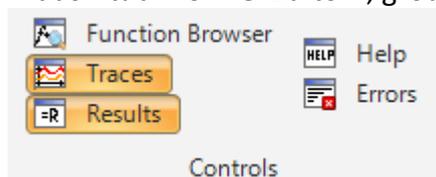


To calculate this formula, click the button "Calculate" in the Ribbon tab "Formel Editor", group "Calculate" or press the button F10.



By pressing the button F10, formals will be calculated even the formula page or the formula window is not activated or visible.

To use and see the calculated values, activate the items "**Traces**" and "**Results**" in the Ribbon tab "**Formel Editor**", group "**Controls**".



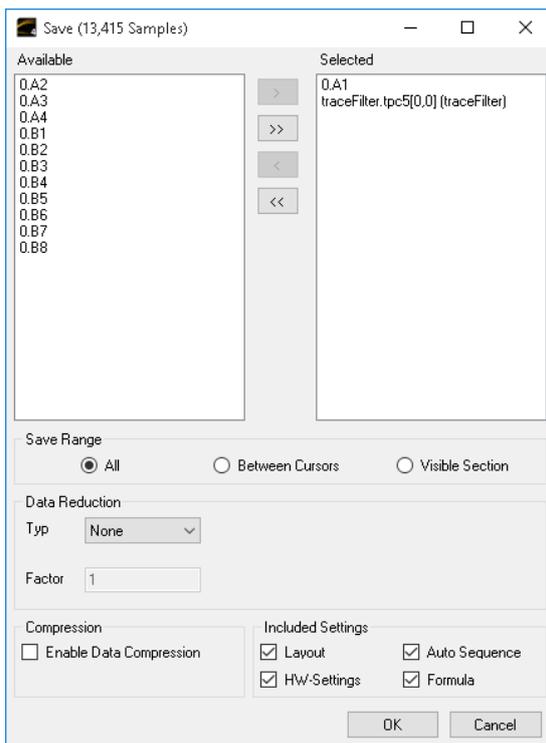
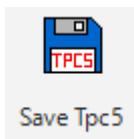
Variable	Value
A	-0.000697102
tB	0.00452533
maxOrig	2.11766
maxSlice	2.11766
maxFilter	1.32561

Traces
traceOrig
traceSlice
traceFilter

"Traces" lists all calculated curves; **"Results"** all calculates scalar values. The curves in "Traces" can be dragged and dropped into a Waveform window for visualization and post analysis.

5.3.4 Save and load recorded signals

After the recording has stopped and the calculations are done, traces can be saved for later analysis or archiving. In the Ribbon Tab "Store Data" group "Save and load data", the button "Save Tpc5" can be clicked.



On the left side, all available traces and curves are listed. On the right side are the traces which will be saved into a file. Double click a curve on the left side or click the icon  to add it to the right-side list. Calculated traces have to be dragged and dropped into a waveform window to be listed on the left side too.

Click the button "OK" to open the file save dialog and for saving these traces.



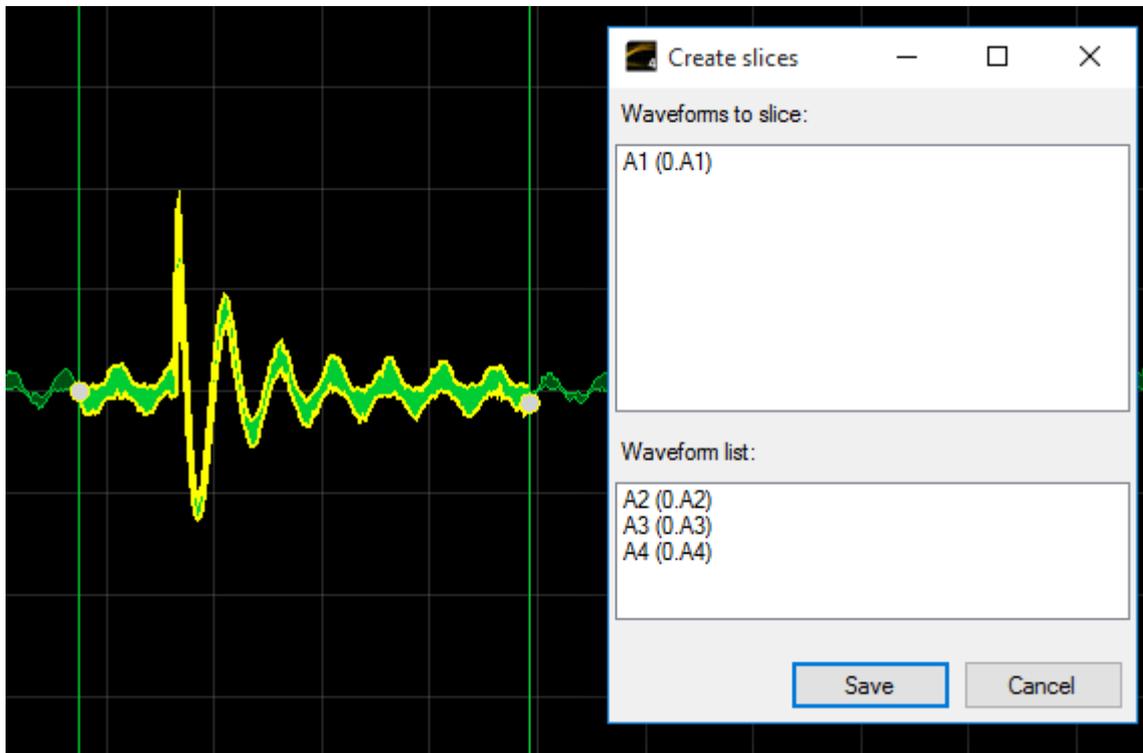
More information regarding options for saving files, please see the Section [Save Traces](#).



Saved traces can be opened, loaded and analyzed again. The single curves from a TPC5 file can be opened with the [Signal Source Browser](#) .

5.3.5 Save a section of a trace

Right click on a trace to open the "Create slices" dialog. There will appear a new green cursor which can be moved to the left or the right side. The yellow marked section can be saved into a TPC5 file.



6 Control Panel

The control panel is used for setting up data **acquisition parameters** such as sample rate, input voltage range, etc. This window is presented **in two sections**:

Ch.	Name	Mode	Coupl.	Range	Offset	Input Range	Avg	Filter	Trigger Mode	Link	Level	Hyst.	Facto
A1	A1	SE	AC 1M	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	1
A2	A2	SE	DC 1M	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	1
A3	A3	SE	DC 1M	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	1
A4	A4	SE	DC 1M	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	1
B1	B1	SE	DC 1M	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	1
B2	B2	SE	DC 1M	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	1
B3	B3	SE	DC 1M	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	1
B4	B4	SE	DC 1M	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	1

The **upper section** contains a table, which lists the current setup for all the channels. The channels are recognized automatically at program start. The table allows the user to select one or more channels in order to modify the setup. By pressing and holding the left mouse button and moving it over the desired channels you can easily select several channels at once.

This method will not work if the mouse cursor is placed over the **first two columns**. These two columns are reserved for moving channels to the waveform display by drag & drop



The upper part of the Control Panel can be saved to a text file. Select the channels as described above and press <Ctrl>+s to open the Save File dialog.

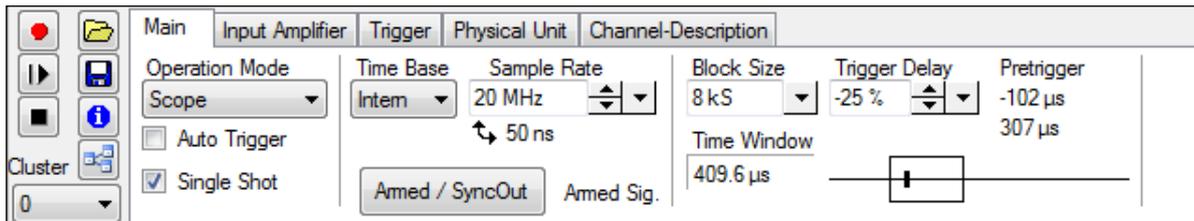
The setup for the selected channels is presented in the **lower display section** of the control panel. The channel parameters can be modified by selecting the relevant tabs.

6.1 Main settings

With a TranAX data acquisition system (transient recorder) it's possible to measure fast signals (transients), but also slow, sporadic and periodical signal.

The basic **hardware contains 4 or 8 channels**. Depending on the Computer you are using, it's possible to add up to **64 channels** per system.

Signals will be recorded parallel, for every channel its own data array will be used, independent of the memory size of the computer. Trigger events can be set individual for each channel; it's also possible to combine these events logical.



The "**Main**" tab contains the following configurable time base parameters:

- Operation Mode:
 - [Scope \(with auto trigger and/or single shot\)](#)
 - [Multi Block](#)
 - [Continuous](#)
 - [ECR \(Event Controlled Recording\)](#)
- Sample clock source: internal or external
- The sample rate (internal clock) or expected clock frequency (at external time base, used for some time related data analyses functions)
- The measurement length (block size with [pre- and post-trigger](#))
- Trigger delay (defines relation of [pre- and post-trigger](#))

By the Button "[Armed / SyncOut](#)" the corresponding output on the Digital Connector can be switched as Armed Out or as Clock Pulse Output with settable frequency. This frequency is independent of the set Time Base Rate. This signal may be used as synchronization of external devices (e.g. High-speed Cameras).

At older devices this button may not be present. Such devices have to be updated at factory.

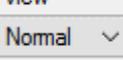
The **sample rate** can be set and displayed using either **frequency** or **time period**. The input field will accept the following short form (*u* for micro, *m* for milli, *k* for kilo, *M* for mega) and for units (*H* or *Hz* for Hertz, *s* for seconds). Example:

Input	Interpreted as	Input	Interpreted as
10k	10 kHz	15u	15 μ s
2.5M	2.5 MHz	500Hz	500 Hz
2.5m	2.5 ms	3s	3 sec

The data acquisition time (Time Window) depends on block length and sample rate.

6.2 Icons

The command buttons to the left of the setting tabs have the following functions:

	Start a measurement (F6).
	Alternatively, the data acquisition can be started by an external TTL-signal and by enabling external start by a click on the corresponding Button "External " in the Ribbon tab Measurement, group "Measurement".
	Manual trigger (F7)
	Stop measurement manually (F8)
	Display Hardware Configuration (information window)
	Change between normal view and Cluster view. In Cluster view, each single board can be assigned to a cluster, each cluster can be configured individually regarding recording mode etc.

6.3 Control Panel tabs

To change the settings, use the dropdown and option lists. Text fields can either be set manually by typing your desired value into it or by the following two buttons:

	In/decrease the value by a given step
	Choose values from the appearing list.
	Switch the parameters which you want to edit.

Some settings will be displayed in a small **illustration** next to the settings. You then also can change the settings by **moving the markers** in the illustrations, e.g. Trigger Delay

7 Virtual Hardware Manager (VHM)

7.1 Short description of VHM

Elsys's Virtual Hardware Manager (simple called VHM) gives whole new possibilities of the usage and utilizing TraNET, TPCX and TPCE instruments for measurement, data acquisition tasks and signal analysis.

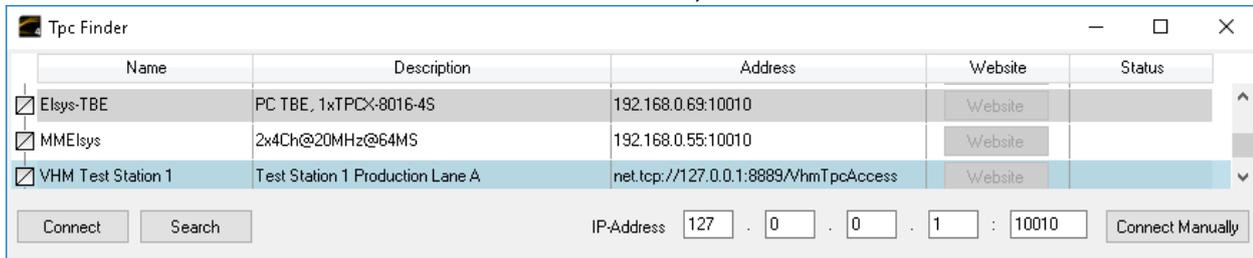
The binding limitation between software configuration and hardware setup can be ignored by using VHM. Preparation of experiments and settings without existing hardware and also combination and conversion of different devices and different number of channels can now be done very simple.

Features

- Server application (VHM Server), handles devices and DAQ modules
- Client application (VHM Client), simple configuration of virtual servers on different computers in the same network range
- Create virtual devices
- Create virtual modules, merge 2x4 channels to 1x8 channels
- Simple drag and drop existing DAQ modules into the virtual devices
- Lock configuration and password protection of Server application

7.2 Connect to a VHM device

Virtual Hardware Manager (VHM) devices are listed in the device manager like any other physical TraNET Device click select instead of a TraNET device, the VHM device and click connect.



7.3 System requirement

For using Virtual Hardware Manager Software and its benefits and features, the following software versions are minimum required:

Software	Version
Virtual Hardware Manager	4.1.0.1795
TranAX	4.1.0.1795



For more detailed information, please read the extra manual for Elsys's Virtual Hardware Manager (VHM). Please don't hesitate to contact your local distribution partner or Elsys instruments directly: info@elsys-instruments.com

8 Operation mode

8.1 Scope

The Scope mode is the default mode. No data will be actually stored to the hard disc. The measured data is only present in the channel memory. After [starting the measurement](#) the actual block will be shown when a [trigger](#) event occurred.

In this tab you can set the time base, the sample rate, block size and [trigger delay](#). The illustration below the trigger delay entry field shows a representation of the trigger point in relation to the complete measurement. The trigger delay can also be set graphically by moving the box in the illustration.

<input checked="" type="checkbox"/> Auto Trigger	If no trigger event occurs TranAX will trigger automatically.
<input checked="" type="checkbox"/> Single Shot	Only one shot will be displayed and the measurement will not be continued.



If more than one [cluster](#) is configured, the **Single Shot** checkbox is always on, auto recording start is not possible. You may use an [Auto Sequence](#) with "start recording" command in a loop instead.

8.2 Multi Block

The Multi Block mode will store signals sequentially in segments of the channel memory. Therefore, each trigger event will initiate a new block of data.

In this tab you can set the time base, the sample rate, block size and [trigger delay](#). The trigger delay can also be set graphically by moving the box in the illustration. In addition to the scope mode the number of blocks respectively the number of measurements can be defined. The maximum possible number of blocks depends on the block length and the total capacity of the on-board memory.

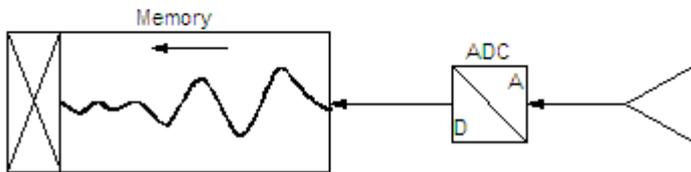
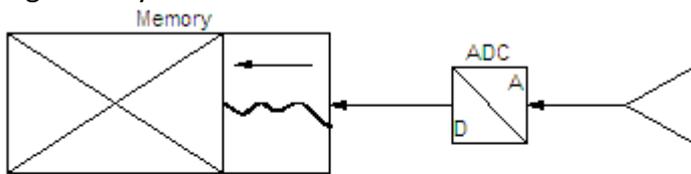


The Multi Block Mode is specially designed for burst-mode applications with a **fast trigger rate** and **minimized dead-time** between Blocks.

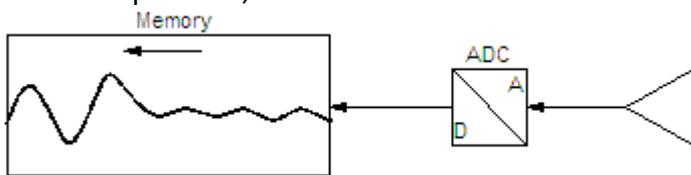
In case it is a requirement to have dead-time free burst mode acquisitions, the ECR Mode is the ideal data acquisition mode.

In Multi block mode, the TPCX/TPCE modules take over full control of measurement process and data storage (channel memory). The measurement will be **allocated to a block size** which matches the available memory. The only involvement of the computer in the measurement process is to give the start command and then wait until the hardware has completed the task. The PC then reads the data. For this reason, block mode is the simplest mode and doesn't require complex settings for quick and satisfactory results.

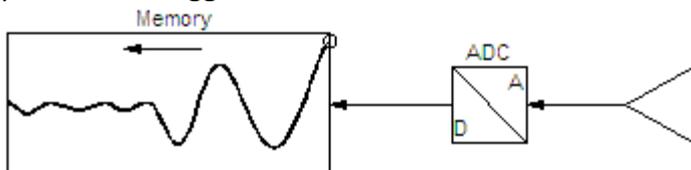
The following illustrations describe the measurement principles in the block mode. Immediately after **measurement start**, the TPCX/TPCE module begins to fill the on-board memory with values digitized by the **ADC**.



From this point on, the **oldest data** will be **overwritten** by new data.



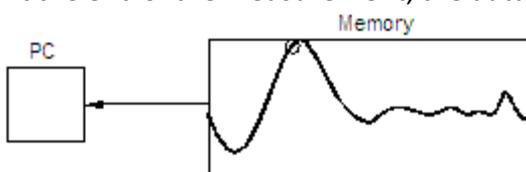
If a **trigger event** occurs, the current sampling stops and all the data is present one block length prior to the trigger event.



If required to capture a signal before and/or **after the trigger** occurs, then the data acquisition runs on a predefined number of samples (see [Pre- and Post-Trigger \(Trigger Delay\)](#)).



At the end of the measurement, the data in the on-board memory is transferred to the computer.



8.3 Continuous

In continuous mode the instrument works as **disk recorder**. No block size or pre/post trigger settings can be made.

The start is normally initiated by the button . There are three possibilities to terminate the data acquisition.

<p>Trailer</p> <p>1 kS  </p> <p>1ms</p> <p><input checked="" type="checkbox"/> Stop Trigger</p>	<p>The acquisition will stop when a trigger event occurs. Additionally, a trailer length must be set. The measurement will then run after the trigger event for a predefined period. The trigger settings are set in the Trigger tab.</p>
<p>Max. Rec. Length</p> <p>10 MS  </p> <p>10s</p> <p><input checked="" type="checkbox"/> Limit Rec. Size</p>	<p>The acquisition will stop after a pre-defined time limit. The Maximum Record Length depends on the free hard disc capacity. If the disc capacity is exceeded, no more data will be recorded!</p>
<p></p>	<p>Stop the measurement manually.</p>

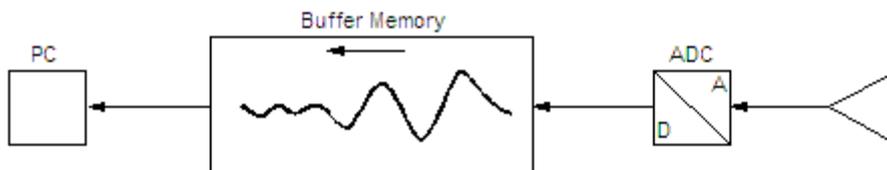


If both options are enabled, then the recording will stop when either one of the events occurs.



If another [Cluster](#) detects a trigger, then this trigger would also initiate the stop trigger of the continuous recording!

In continuous mode the **measured data** is continuously written to the computer where it will be stored, for example, to the **hard disk**. The measurement can be stopped either by a signal trigger event or manually via computer. The **on-board memory** is used as one large **buffer** in this mode.



The **measurement length** is limited by the hard **disk capacity** and the sample rate is limited by the transfer speed between the module and the computer. Depending on the computer specification, the total sample rate can be up to a few tens of mega-samples per second. This maximum rate is achieved when no other applications are running at the same time. The measurement is protected from fluctuations in sample rate or loss of data due to computer loading by buffering the data through the large on-board memory (up to 64 MSamples/channel). This measurement mode is intended for data captures **over longer periods**.



The recording stops automatically if the hard disk gets full.

8.4 ECR mode



The ECR mode is a software option.

The ECR mode allows targeted acquisition of **cyclic or sporadically arising events**. This implies that the registration of measuring data only occurs if certain **signal conditions** (trigger, time window, repetitions, etc.) are fulfilled. Thus, many unwanted and unneeded signal data will not be stored.

Nevertheless, it can be guaranteed that no dead times arise and therefore no events will be lost. This even applies if many channels at maximum sample rate have to be supervised over a long period of time. Since each channel possesses its own signal buffer (up to 64M samples), only the average number of events per second may not exceed a certain value. This value depends on the adjustable block length per event and furthermore it is defined by the maximum possible transfer rate to the hard disk (**approx. 20M samples per second**, depending upon CPU/Disk systems). The trigger conditions can be individually set for each channel, whereby even more complex signal criteria (e.g. pulse width/height, slew rate, window-IN/OUT) can be defined.



Compared to the block mode, with ECR mode it is guaranteed to have **no dead times** between adjacent blocks. Note that, if in block mode a trigger event occurs at the end of the block, the event might not be recorded.

In the ECR-mode it is guaranteed that there is no dead-time between adjacent blocks. The overlapping data-area depends on the event-rate and it can be controlled within certain limits with the Holdoff function. In Block Mode on the other hand, the blocks are strictly sequential data acquisitions with a gap between blocks.

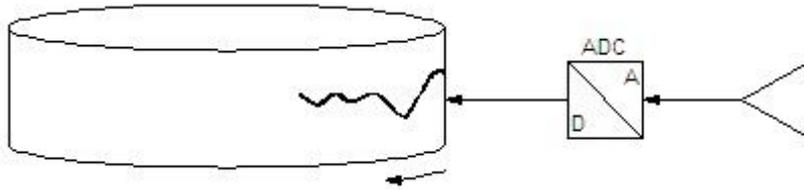
If the operation mode is set to ECR mode, an additional [ECR tab](#) will be opened. In the ECR mode the block size is determined explicitly by [pre- and post-trigger settings](#). As with the multi block mode you also can set the maximum number of blocks that will be recorded. Furthermore, there is a [Retrigger \(RT\)](#) marker in the illustration below the settings or a [Holdoff \(HO\)](#) marker shown, depending on the settings made in the [ECR tab](#). There are two different ECR modes, the single and multi-channel mode. Both modes support a [Dual mode option](#).



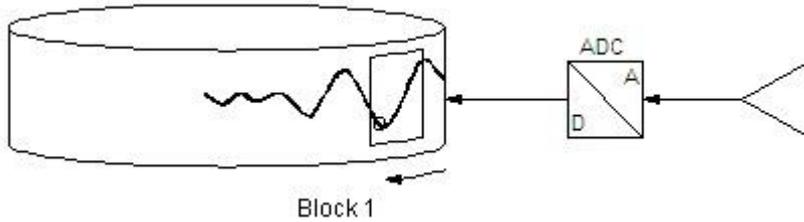
If the trigger conditions are set very uncritically, then in ECR Mode the CPU could easily be overloaded by fast periodically signals. The CPU might seem to be blocked.

8.4.1 Basic Sequence

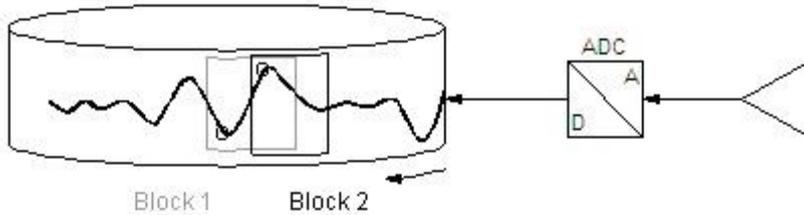
The ECR mode runs as follows: The digitalized signal will be stored to the on-board memory which acts as a **ring buffer**.



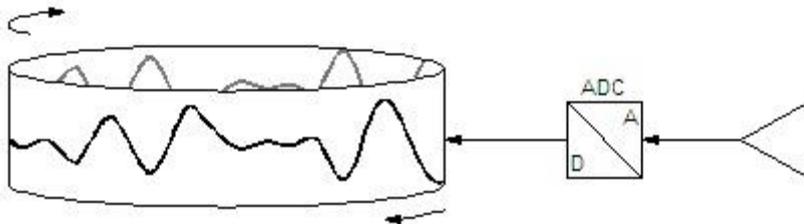
As soon as the **trigger** is released, a block of samples will be read from the ring buffer and will be saved to the **hard disk**.



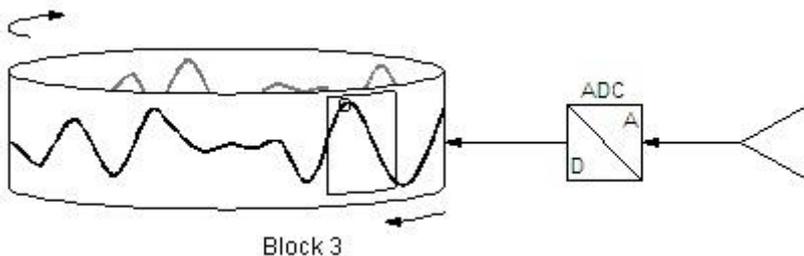
If a **new trigger event** within the actual block occurs, a new overlapping block will be saved.



If the ring buffer is full, the **oldest measurement data** will be **overwritten** with new incoming data. Usually, the overwritten data would be transferred to the hard disk before this happens. If too many events occur in a period of time, the **ring buffer may overflow**. TranAX will display a message according to the status.

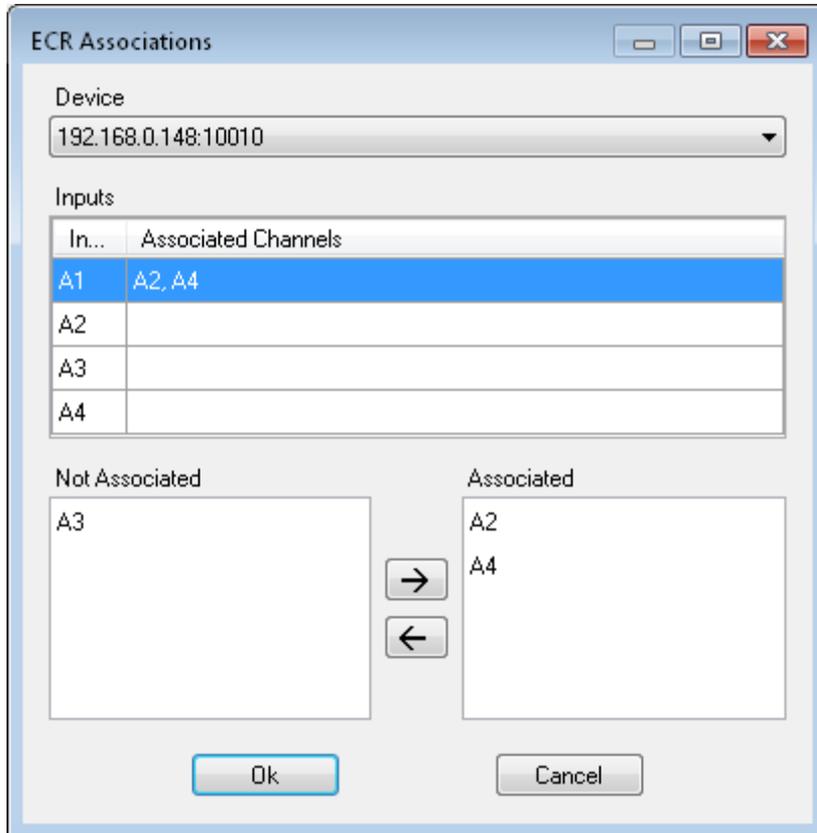


After the predefined number of saved blocks is reached (in this example 3), the recording stops.



8.4.2 ECR single channel mode

The signal data is being acquired on **each selected channel** on trigger command from each channel's internal trigger circuit and stored into memory. Signal data from selected associated channels will store their data parallel and synchronously with the triggered channel it is associated to. To **associate a channel**, press the button "ECR Associations" and a window as shown below will appear:



Simply select the desired input channel from the Input-field, choose not yet associated channels and press the right arrow .

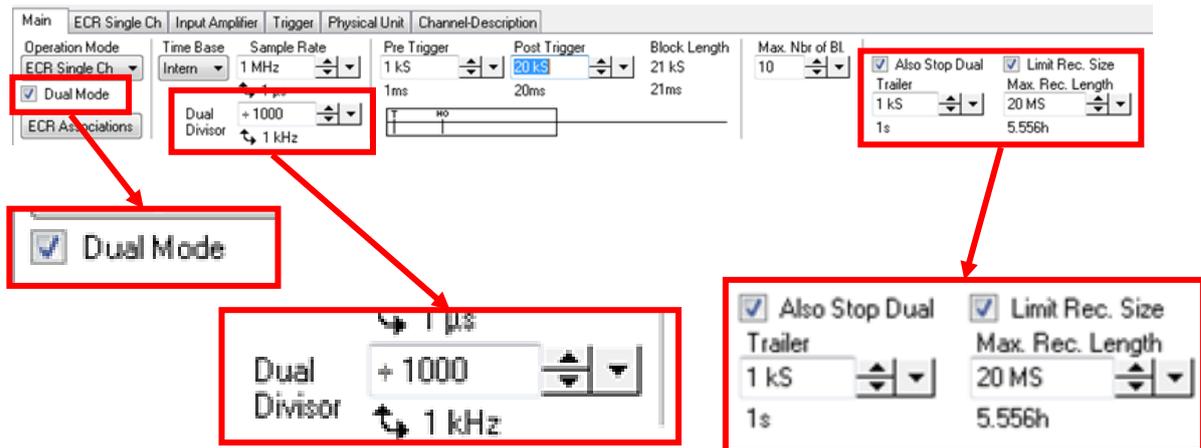
8.4.3 ECR multi-channel mode

The signal data is being registered parallel from all active channels which are not switched off in the Control Panel.

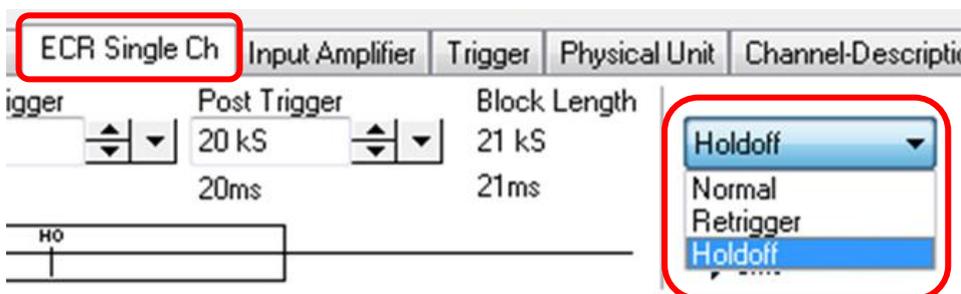
8.4.4 Dual mode

Switched ON, it will record (usually relatively slow) the signals of all active channels continuously parallel to the registration of (fast recorded) ECR trigger events. The **clock rate** can be adjusted (by a **clock divisor** parameter, Dual Divisor) for a slower continuous recording in relation to the faster sampling rate.

From start to stop conditions, the slower recording runs **synchronously** to the registered ECR events. The slower continuous record always stores the data of all active channels.



8.5 ECR Trigger option



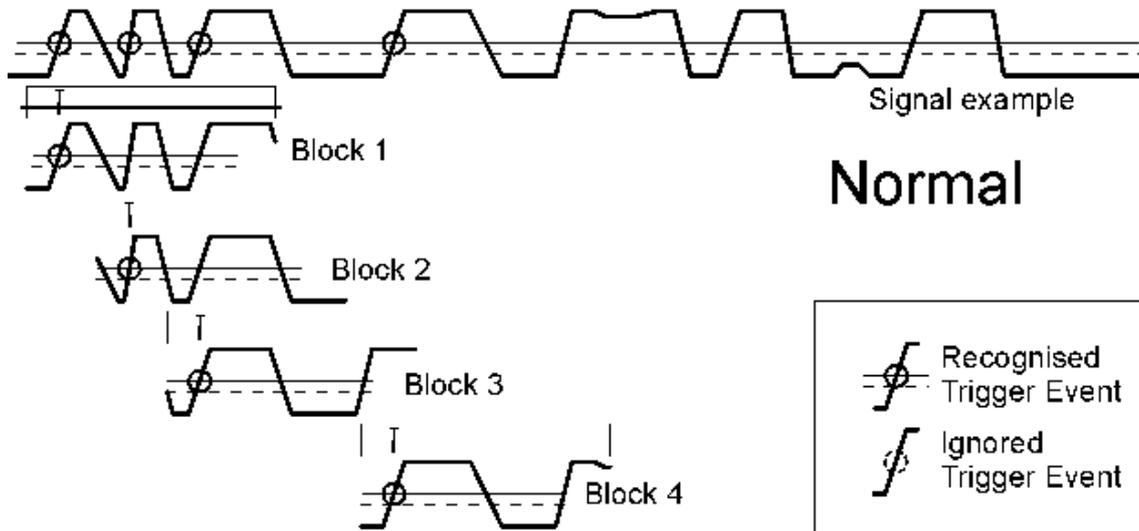
Additionally to the ECR mode settings which can be made in the [Main tab](#), in the control Holdoff in the *ECR* tab you can choose between the [Normal](#), [Retrigger](#) and [Holdoff](#) options and set the Retrigger & Holdoff markers.

The Pre-Trigger, Post-Trigger and Number of Blocks can be either set in the Main or ECR tab.

By leaving the option at *Normal* no further settings can be made than those made in the [Main tab](#).

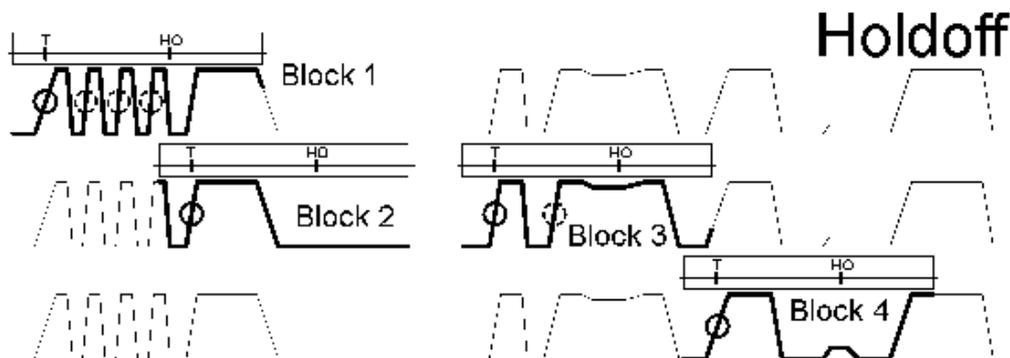
8.5.1 Normal

With the additional trigger settings Retrigger or Holdoff the recording of an unwanted number of overlapping blocks can be avoided.



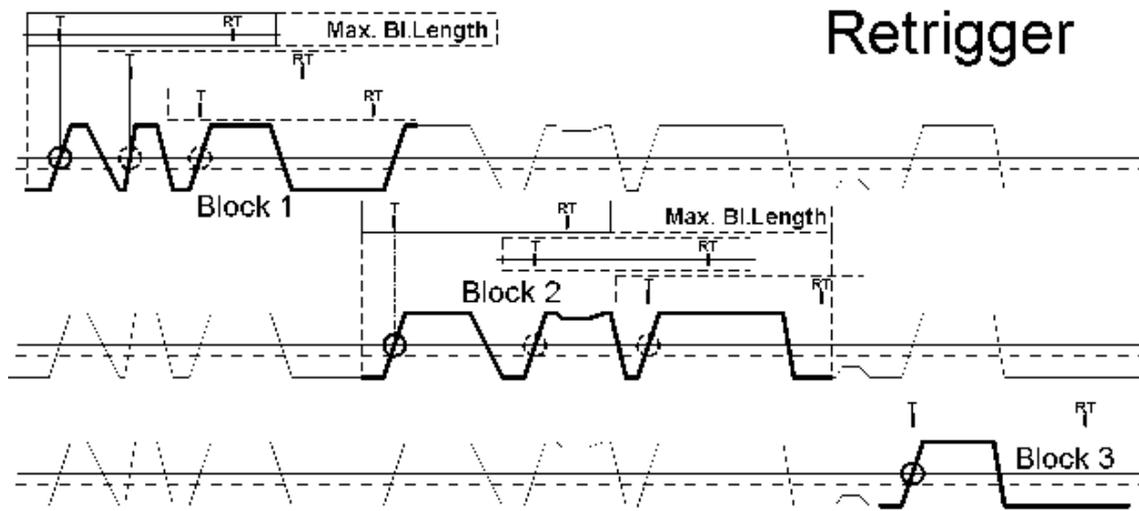
8.5.2 Holdoff

With the Holdoff control set to Holdoff, you can instruct TranAX to **ignore** all additional trigger events until to the Holdoff marker HO.

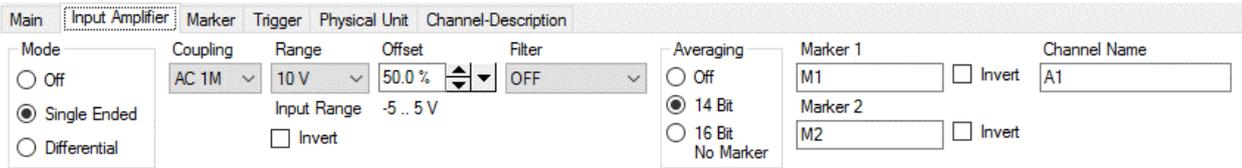


8.5.3 Retrigger

Choosing the trigger mode Retrigger with the Holdoff control you can set the retrigger mark in **sample lengths** or in time measurement. Contrary to the Holdoff option, trigger events will be **ignored** to be recorded, but as soon as a new **trigger event** occurs, the retrigger marker RT will be moved on and set newly relatively to the new trigger event (respectively retriggered). Only after the retrigger marker is passed, a new block will be stored. Additionally, a maximum Post Trigger block length can be set. TranAX will trigger according to the illustration below:



9 Input Amplifier



The following channel parameters are set in this tab:

- **Mode:** Single Ended (screen to ground), Differential or Off
- **Input coupling:** DC, AC or ICP (Integrated Current Power for Piezo sensors). For the modules 120MS and 240MS modules, the input impedance can be set to 50Ω. For all other modules, this value is set to 1MΩ.
- **Input voltage range:** Total range and offset
- **Filter:** Incl. Anti-Aliasing filter, (optionally available)
- **Input inversion**
- **Channel name**
- **Averaging:** Off, 14Bit or 16 Bit
- **Marker name** (optional digital inputs)
- **Marker signal inversion**

The above values may be set for **each individual channel**. All installed channels are detected at program start, these are then included in the table.

9.1 Averaging

The ADC runs always with the maximum possible sample rate. If the selected sample rate is less than the maximum rate, then the excess samples will be averaged. This way the **signal to noise ratio is improved** correspondingly. For applications which don't allow averaging (e.g. under sampling recording), it can be switched Off.



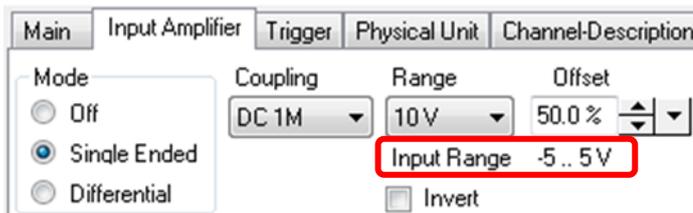
The parameter "**Averaging**" will be set for **all channels** within a module.



In some cases, averaging should not be used, e.g. for **under measuring** (sampling with a lower frequency then the measured signal). In this case, averaging has to be set to "off".

9.2 Amplifier options

The **input voltage range** settings are defined in two parts: **Range and offset**. The range sets the maximum possible data capture voltage amplitude. The offset sets the zero point of that range



and therefore the absolute minimum and maximum voltage limits. These limits are displayed both in the data input field and in the table. **Each input channel** can be set to operate in **inverted** mode i.e. the polarity of the input voltage is inverted.

Each channel can be given a **name** in order to **identify** it with its relevant signal.

9.3 Markers (Digital Inputs)

Every data acquisition channel has two digital inputs called **Markers**. Marker signals are digital signals with values 0 or 1 and they can be displayed in the dedicated **Marker Waveform Display**. The controls "Marker 1" and "Marker 2" allow defining names for the digital input signals. In case Invert is selected, the marker will be inverted before it is displayed. This is useful in cases of signal inversion, for example, with opto-couplers. [Also see the chapter on Limitations / Digital Inputs \(Markers\)](#).



Marker inversion will be marked with a "\" (Backslash) at the end of the name. "M1\" means the inverted Marker 1.



In case the TPCX/TPCPE digitizer module is set to 16-bit mode, there are no Markers (digital inputs) available.



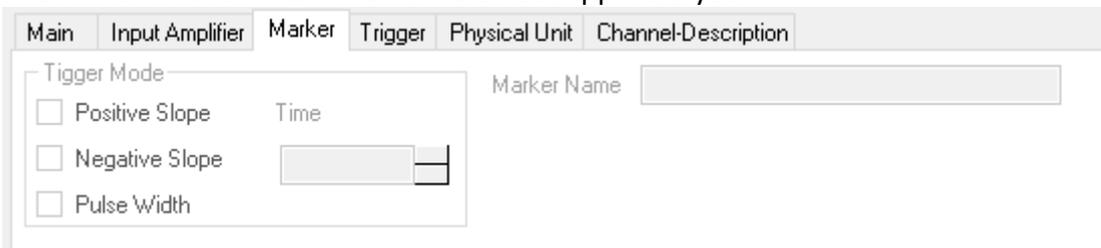
These settings are always visible in the Control Panel, even when there is no Marker option installed.



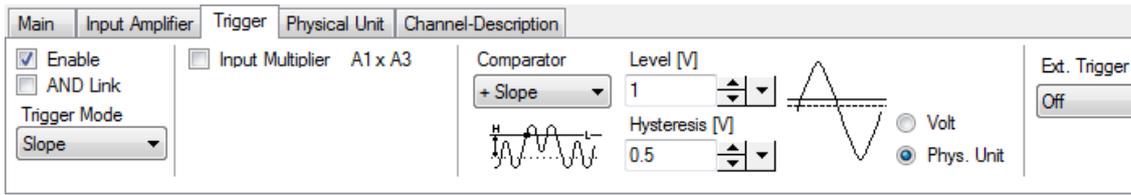
The corresponding analog channel must be switched **ON** (tab "Input Amplifier" in the Control Panel) to record the Marker signals.

10 Marker

The functions in the tab "Marker" are not supported yet.



11 Trigger



The following channel parameters are set on this tab:

- **Trigger Enable:** The **selected channel** will only be an active trigger source if Enable is selected.
- **AND Link:** The AND trigger logic can be configured per 4-ch or 8-ch module
- **Trigger Mode:** See "Trigger Mode" section for detailed information about the operating modes
- **Input Multiplier**
- **Comparator (Slope)**
- **Trigger Level [Volt or Physical Unit]**
- **Trigger Hysteresis**



The trigger condition is one of the most critical settings. If the trigger conditions are not set correctly, either unwanted trigger events occur or the settings are invalid so a trigger condition is never met.

If no satisfactory recording can be achieved, the trigger settings need to be checked. The trigger parameters can be set individually for each triggerable channel. At program start the software automatically identifies which installed channels are triggerable. The triggerable channels have entries in the table under *Trigger Mode* and *Level*.

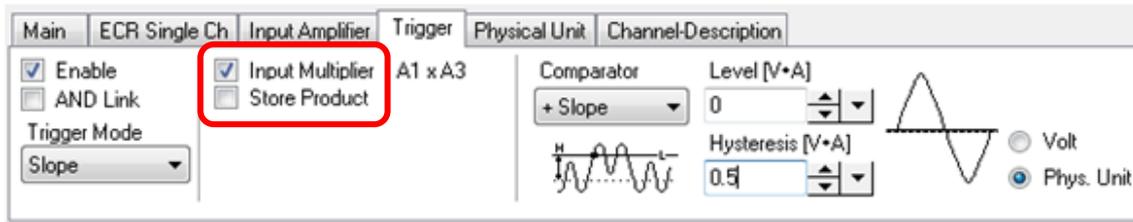
Depending on the selected trigger mode, either level and hysteresis ([+ Slope](#), [- Slope](#), [± Slope](#)) or an upper and lower level ([Trigger Window in](#), [Trigger Window out](#)) in volts or in physical units can be defined. By setting the hysteresis it can be avoided to trigger on an undesired edge, when the signal has noise superimposed on it. For information purpose, the current level and hysteresis settings are displayed in small graphic representation.

Ch.	Name	Mode	Coupl.	Range	Offset	Input Range	Avg	Filter	Trigger Mode	Link	Level	Hyst.	Phys. Range	M1
Device: EPC4 (192.168.0.111:10010)						Disk Space: 23.05 GB								
A1	A1	Diff+	DC	100	0	0 .. 100 V	14 bit	Off	+Slope	OR	1 V	0.5 V	0 .. 1000 V	M1
A2	A2	Diff-	-	-	-	-	-	-	-	-	-	-	-	-
A3	I	SE	DC	0.1	10	-0.01 .. 0.09	14 bit	Off	+Slope	OR	1 V	0.5 V	-0.6 .. 4.4 V	M1
A4	A4	SE	DC	10	50	-5 .. 5 V	14 bit	Off	+Slope	OR	1 V	0.5 V	-5 .. 5 V	M1



If in the trigger setup of a channel e.g. an invalid level or hysteresis is set, the corresponding field in the table will change its color to yellow/red as shown in the picture above. This for example happens when a trigger level of, let say 1V is set for an input and then the sensitivity range of this input is set to 0.5V. In this case a trigger level of 1V can never be reached.

11.1 Input multiplier



The function **Input Multiplier** multiplies the currently digitized **signals of two channels** (e.g. A1 and A3 or A2 and A4). The resulting signal (a product) will then be passed to the trigger discriminator instead of the original signal from channel A1 resp. A2, on which the module will trigger. The range and resolution of the products depend on the settings (range, offset, physical scaling factor and physical unit) of the two channels.

The full range and full resolution can only be exhausted to half. This will be the case if both channels are set to 0% or 100%. The multiplied product curve will then have a range from 0 to +Max or to -Max. There will never be a multiplied product in a range from -Max to +Max. **Having the offsets set to 50%, only one fourth of the maximum possible resolution is achieved.**



Note that the hardware multiplier doesn't take the constant of the [Physical Scaling](#) into account. It should therefore be set to 0 for both channels.

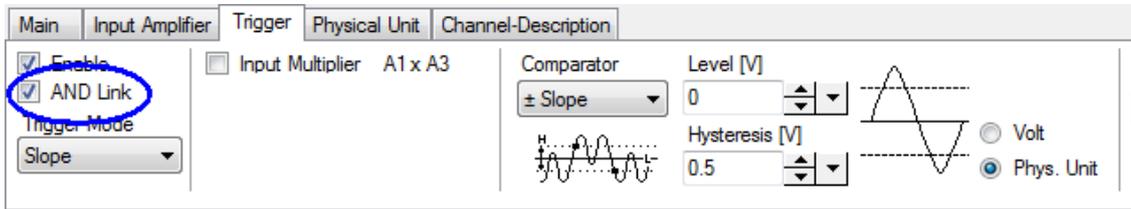
The checkbox **"Store product"** only visible when **"Input Multiplier"** is activated will replace the recording of one channel (e.g. A1) with the result of the multiplied two channels. The second channel (e.g. A3) will contain the normal recording (unchanged).

The amplitude resolution of the product signal is also 14-bit resp. 16-bit. If the original signals don't use the full dynamic range of the amplifier and the ADC, the resolution of the output signal can be strongly minimized. It's recommended to choose for both signals, **measurement ranges that optimize** the dynamic range of the input stages.

11.2 AND Link (logic AND operation)



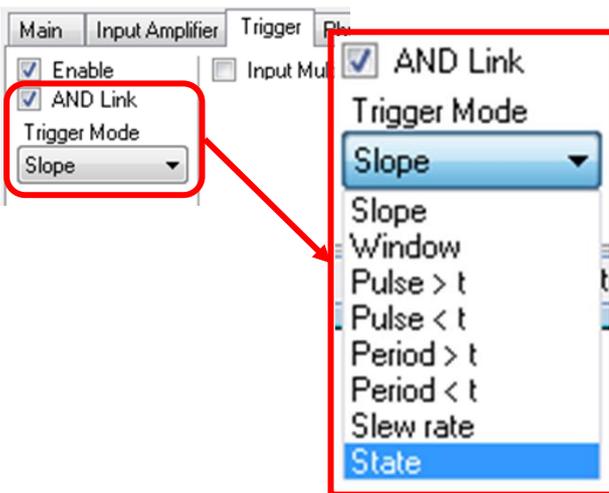
AND Link is a feature of the TranAX option "Advanced Trigger".



By default, all active trigger sources are combined in an OR-logic. This means, **any enabled trigger source** can trigger all the channels simultaneously of a TraNET instrument or a TraNET system synchronized with Sync-Link.

In cases where AND-logic is required, all trigger sources of a 4-ch or 8-ch TPCX/TPCE module, e.g. A1 - A8, can be combined in an AND-logic. Therefore, the AND Link needs to be activated on those channels of a module that need to form the desired AND combination.

The AND-logic trigger allows to combine all trigger modes, including the AND Link dedicated State trigger mode, across all channels of one module. This enables the user to trigger on complex signals.



As soon as **AND Link is activated** the trigger mode "**State**" becomes available.

State can be used as a **qualifier** for another trigger source. Only if the State of a trigger source is met AND when the conditions of another trigger source are met, it will be triggered.



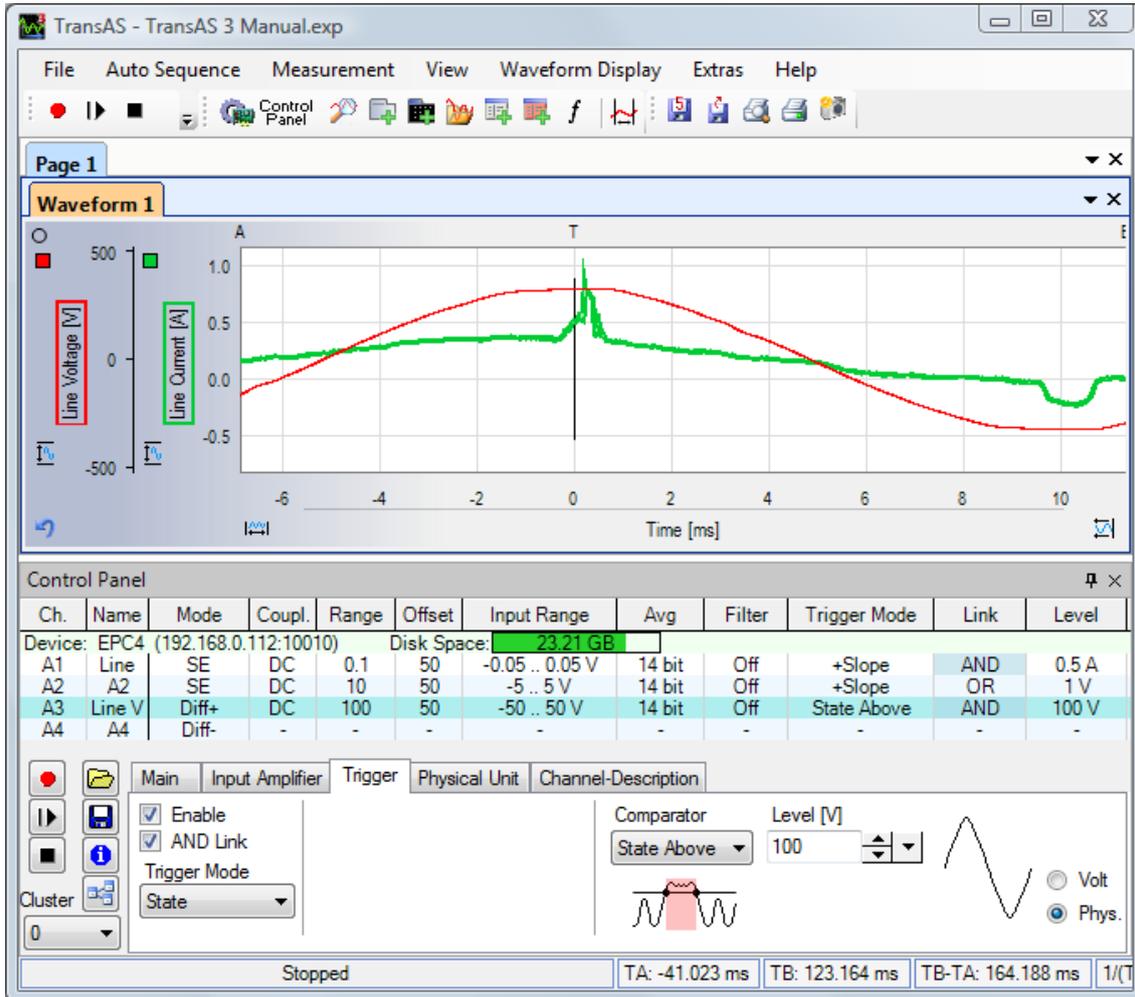
If only **one trigger source is set to "AND Link"**, it acts exactly the same as in OR logic. To use this function, there needs to be at **least two channels** with enabled "AND Link".



While the default OR-logic works with all the channels of a Transient Recorder system across up to 8 TraNET instruments synchronized with Sync-Link, the AND -logic is applicable just to a 4-ch or 8-ch module in a system.

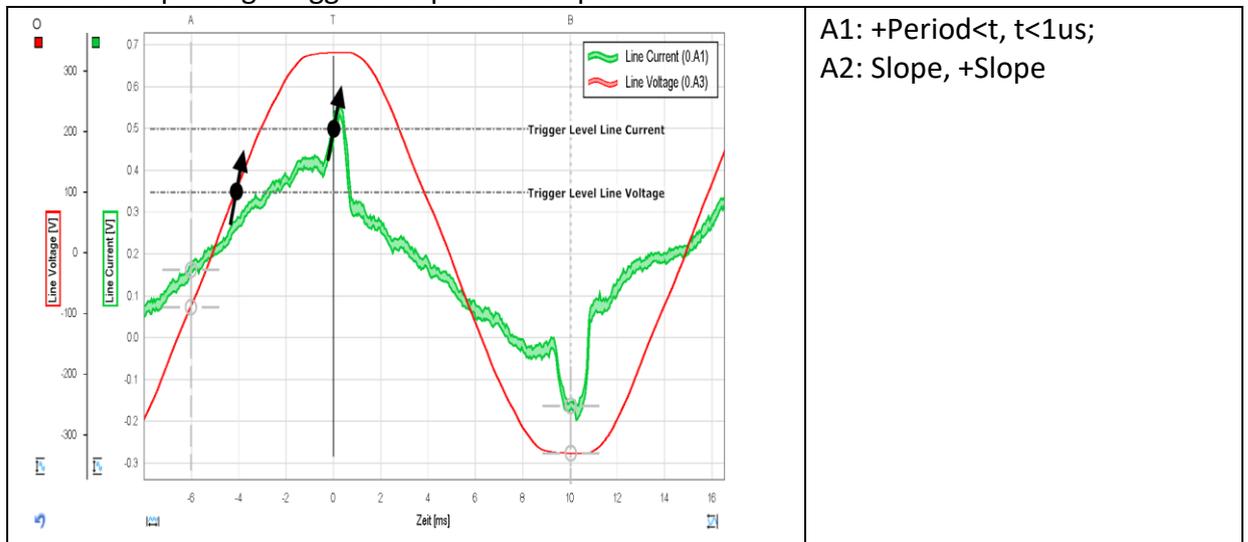
11.2.1 Example 1: AND-link (Slope and State)

Only after the signal "Line Voltage" is **above 100V** AND the channel "Line Current" goes above **0.5A**, the trigger condition is met. Both channels have to be on the same module, AND Link has to be enabled.

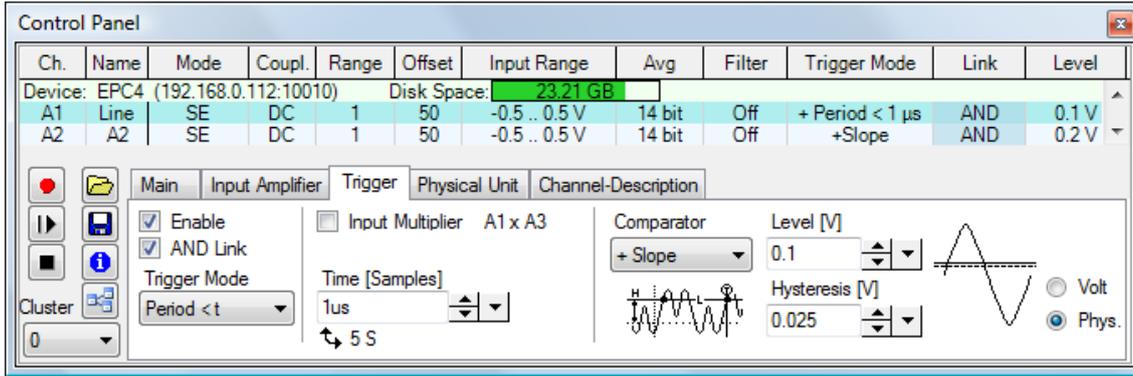


The examples below illustrate how the AND trigger can be configured. In the example on the left-hand side, one channel is configured as State and the second channel as positive slope. When both conditions are true the module will trigger.

In the second example channel one is watching out for pulses smaller than 1us and channel two just for a simple edge trigger of a positive slope.



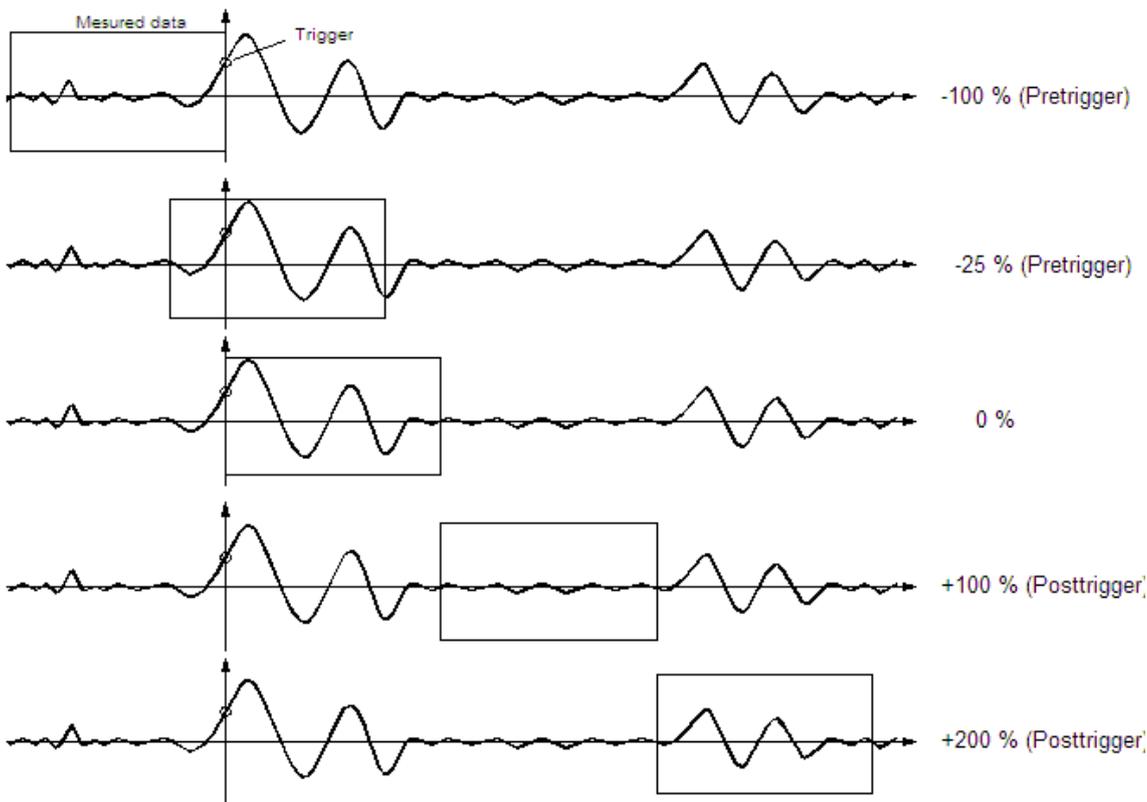
The Control Panel shows the current trigger configuration in the columns Trigger Mode, Link, Level and Hysteresis. All settings are performed in the Trigger tab.



11.3 Pre- and Post-Triggering (Trigger Delay)

Pre- and post-trigger is applicable to [Scope-](#), [Multi Block-](#) and the [ECR mode](#). The position of the **measurement window** (or block) can be adjusted relatively to the **trigger point** within limits. If it is required to capture a signal prior to the trigger point, this is called pre-triggering. Conversely, if it is required to capture a signal after the trigger point it is called post-triggering. These trigger delays (-% for pre; +% for post) are defined in terms of percent of the total block period. The TPCX/TPCE hardware allows a trigger delay between -100% and +200%.

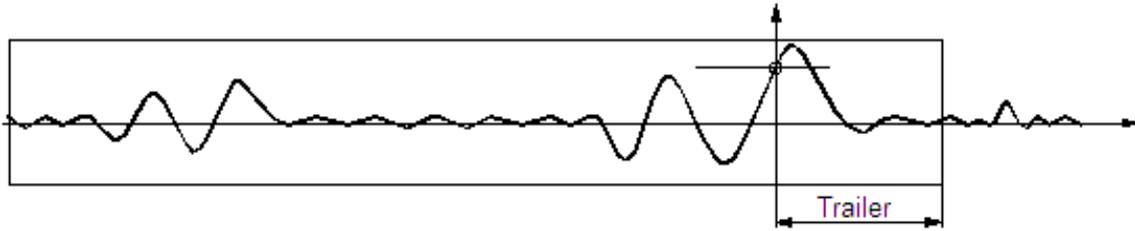
Trigger delay



Trigger delay (trailer)

There is no pre/post trigger in [Continuous mode](#). In this mode the stop trigger and **trailer** are utilized. The stop trigger is used to determine the end of the measurement i.e. data acquisition stops at trigger. However, sometimes it is required that the measurement continues for a predetermined time after the stop trigger - this is called the trailer. The trailer is defined in

number of samples after stop trigger. The TPCX/TPCE hardware allows settings from 0 (no trailer) to 16 MSamples. This option is also included in the [ECR dual mode](#).

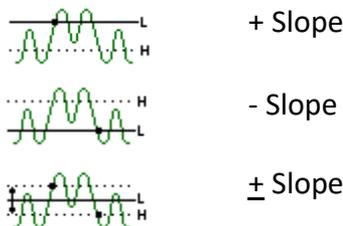


11.4 Trigger Modes

The following trigger types can be set on any analogue channel "L" is the **trigger level** and "H" the configurable **hysteresis band**

11.4.1 Slope

With the **slope trigger** settings, you can select the positive, negative or both slopes of the trigger signal. A trigger will be generated when the **hysteresis level** has been **passed** and subsequently the slope level has been reached.



11.4.2 Window

Selecting the **window in** trigger option, trigger occurs when the signal enters the window. With the **window out** trigger option, trigger occurs when the signal leaves the window.

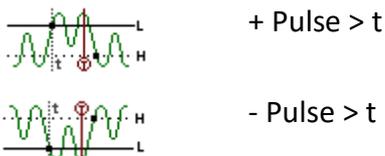


The following trigger modes require the advanced trigger option.

11.4.3 Pulse > Time

As soon as a positive or negative pulse is recognized, a trigger is generated if the pulse width is greater than the specified time, respectively if the signal doesn't reach the hysteresis level within the defined time domain.

Please note: To determine the end of a pulse it must be considered to set the trigger hysteresis.



11.4.4 Pulse < Time

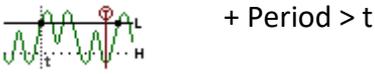
A trigger is generated as soon as a pulse width is smaller than the specified time.





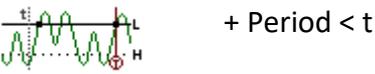
11.4.5 Period > Time

A trigger is generated if the period is greater than the defined time. Also, the hysteresis will be considered to detect level crossing of periods. The hysteresis allows suppression of illegal periods (e.g. high frequency noise).



11.4.6 Period < Time

A trigger is generated as soon as a period width is smaller than the specified time.



11.4.7 Slew rate

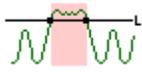
With the slew rate you can generate triggers on specified positive or negative slew rates. It's mainly used to detect fast parasites or spikes on slower periodic signals. It actually works like a trigger generator's low frequency suppression.

The slew rate has to be defined by **Delta Samples** (Delta times) and Delta-Y (Delta amplitude). Delta-Y should be set to a value at least twice the expected noise on the signal. The **Delta-Time** parameter is **limited to 1024 samples**. The resulting slew rate value can be examined in the column *Trigger Mode* of the channel list in the Control Panel.

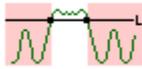


11.4.8 State

The **State trigger mode** is only available when the **AND Link is activated**. State trigger is used in an AND combination to qualify another trigger source or several trigger sources of one and the same 4-ch or 8-ch module.



State Above



State Below



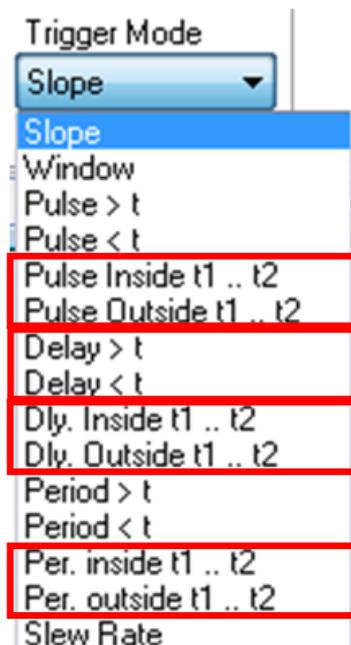
In addition, each device has an **external trigger** input available (TTL). Triggering can be enabled or inhibited using a second external input (TTL) called **disarms**. For more information about the pin layout of the external digital I/O connector please see the hardware manual.

11.5 Advanced Trigger-Modes (Overview)

In addition to the existing seven trigger modes, **eight more** have been implemented. They also need the Option *Advanced Trigger*. The existing trigger modes are described only rudimentary in this document.

The eight new trigger modes:

- [Pulse inside t1 .. t2](#)
- [Pulse outside t1 .. t2](#)
- [Delay > t](#)
- [Delay < t](#)
- [Delay inside t1 .. t2](#)
- [Delay outside t1 .. t2](#)
- [Period inside t1 .. t2](#)
- [Period outside t1 .. t2](#)



The **times t, t1, t2** can also be set in the **Control-Panel** as a **number of samples**. TranAX calculates (multiplied by the sampling rate) the corresponding times. **Internally** all numbers are handled as **number of samples** not as time value.



The setup of times changes, when the time base is adjusted!

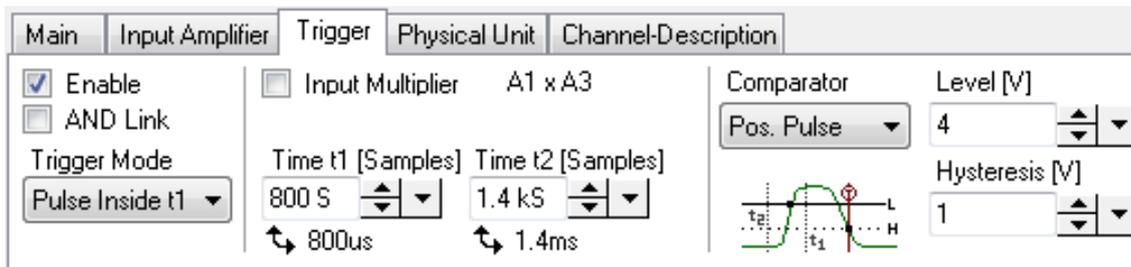


To use the new trigger modes, the installed software may need to be upgraded. The following **versions** are **prerequisites**:

TranAX:	3.2.1.624	(Menu " Help " / " About ")
TPC-Server:	1.3.2	(Control Panel / )

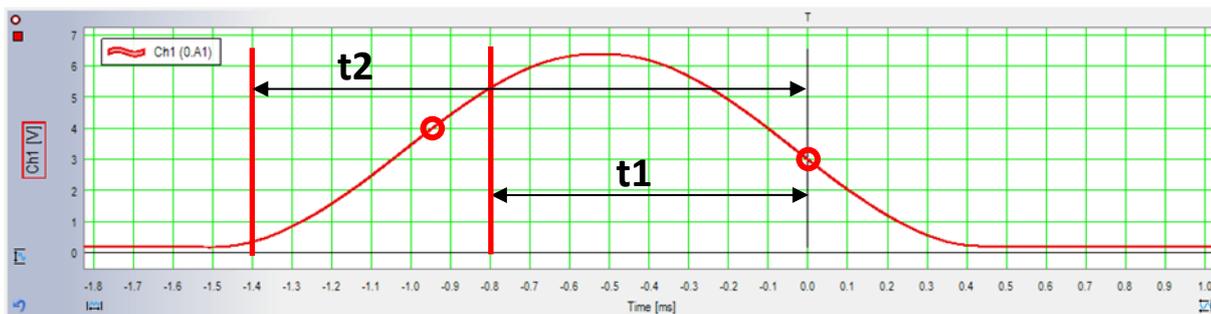
11.5.1 Pulse inside t1 .. t2

Triggering will take place, when a pulse appears within the set time limits t1 and t2. In this example, the limits are set to t1 = 0.8ms and t2 = 1.4ms. Sample rate is set to 1MS/s. 800 Samples correspond exactly to 0.8ms respectively 1.4KS correspond to 1.4ms.



Settings for channel 1.

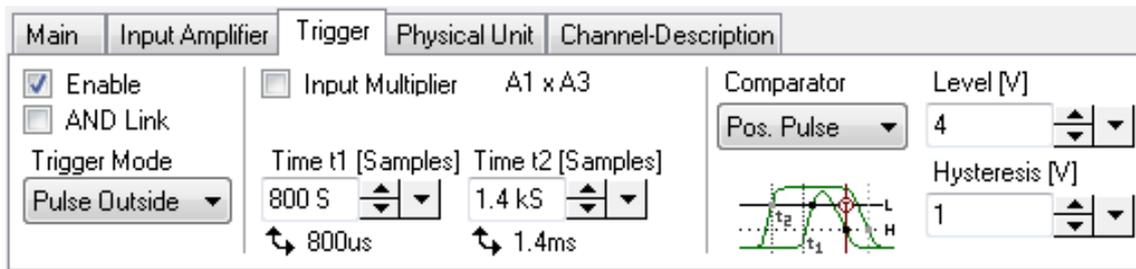
The trigger comparator is set to a positive pulse at 4V, hysteresis at 1V, therefore the condition is met at the falling edge of the signal. Trigger zero point is on the falling slope at a value of Level minus Hysteresis, thus 3V.



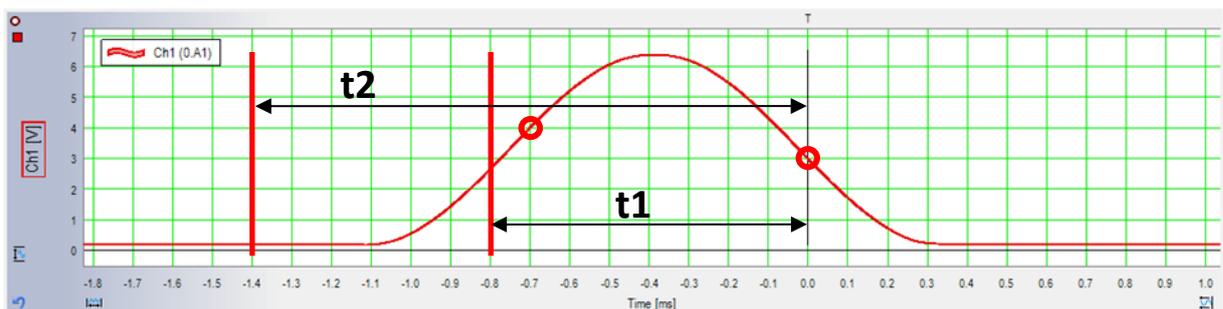
The pulse width is 0.95ms and inside the time limits t1 and t2 (0.8ms and 1.4ms).

11.5.2 Pulse outside t1 .. t2

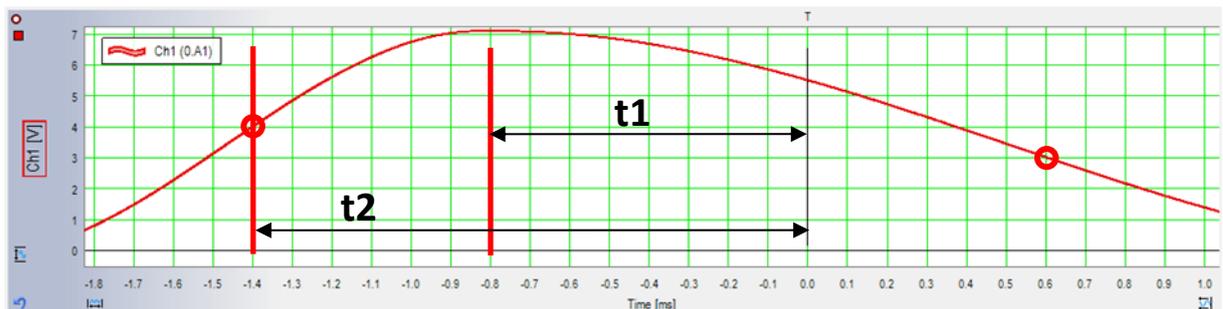
Triggering will take place, when a pulse appears outside the time limits t1 and t2. This means that the pulse width has to be shorter than t1 or longer than t2 to meet the trigger condition (pulse width < t1 or pulse width > t2).



Settings for channel 1.



Pulse width is shorter than t1 (*pulse < t1*).



Pulse width is longer than t2 (*pulse > t2*).

The trigger zero point is located exactly t2 behind the criterion for the start of the pulse (here pos. edge, level = 4V). At the trigger zero point, no trigger condition of the signal is met. The trailing end of the pulse crossing the 3V trigger level behind zero-point t2 thus generates a trigger.

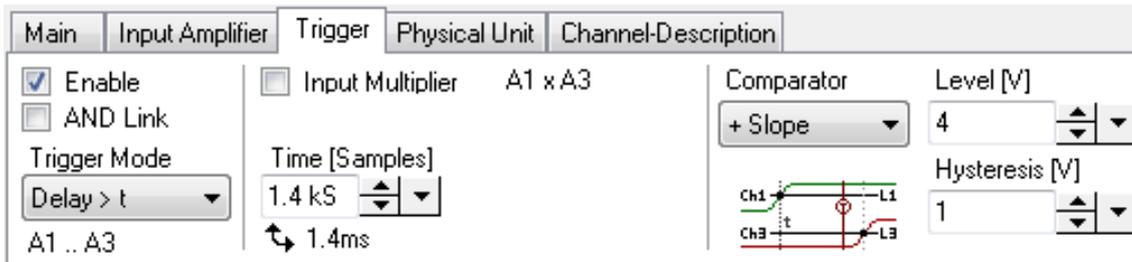
11.5.3 Delay > t

This trigger mode uses the signals of channel 1 and 3 (respectively 2 and 4). It captures when the time between trigger condition of **channel 1** and the condition of **channel 3** is longer than the pre-set time t . In this case, the system generates a trigger.

In addition to the **time** t , the comparator settings (edge, level, and hysteresis) for the two channels 1 and 3 must be set.

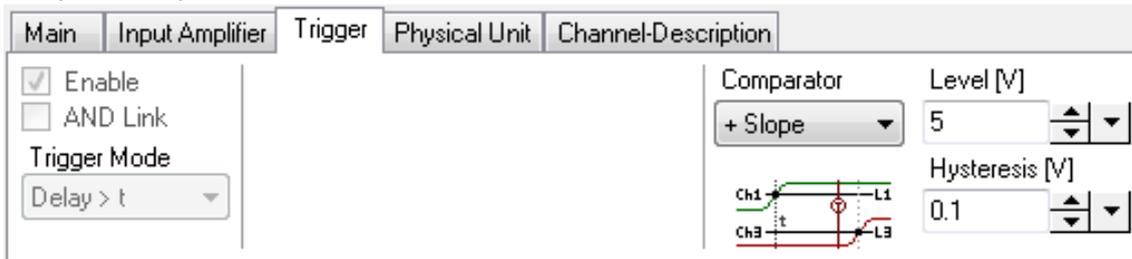


For this trigger mode, channel 1 and 3 as well as channel 2 and 4 are combined. Other combinations of channels are not possible.

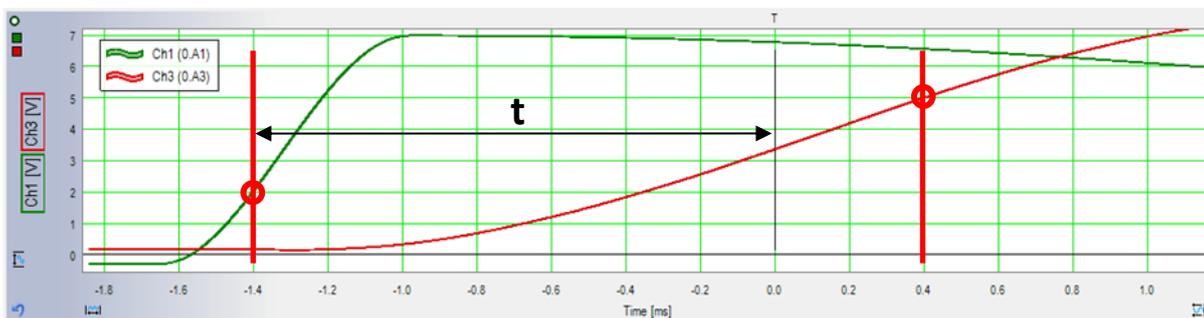


Settings for channel 1.

The trigger mode for the associated channel 3 (or 4) is determined by channel 1 (or 3). The comparator settings for these channels, Level, Hysteresis and slope ("+", "-" or "±") can be set independently.



Settings for channel 3. The trigger mode is determined by channel 1.

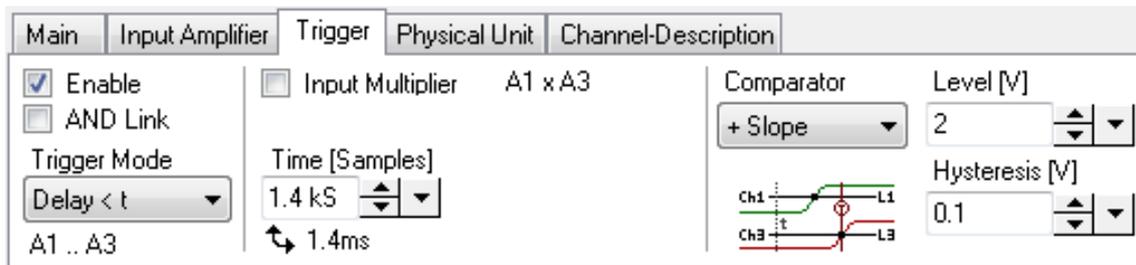


Delay is longer than t (delay > t).

The trigger zero point is exactly the time t behind the trigger condition for channel 1 (pos. slope at 2V). At the trigger zero point, no trigger condition of the signal is met. The rising edge of the pulse crossing the 5V trigger level is behind zero-point t_2 , thus generates a trigger.

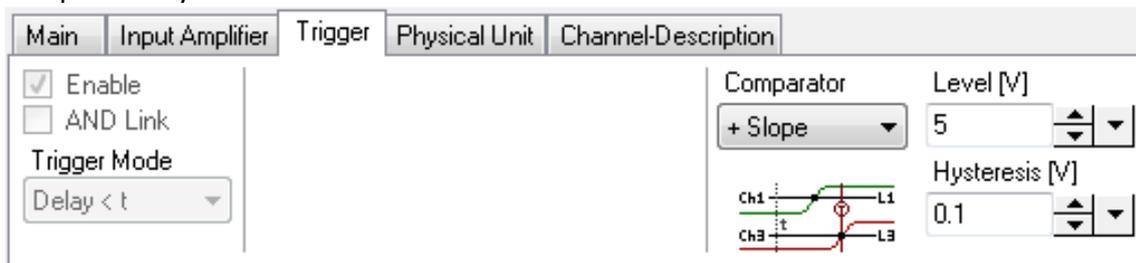
11.5.4 Delay < t

Here a trigger is caused, when the delay is shorter than the time t . In addition to the time t , the comparator settings for the channels 1 and 3 (respectively 2 and 4) can be set individually (Level, Hysteresis, Slope).

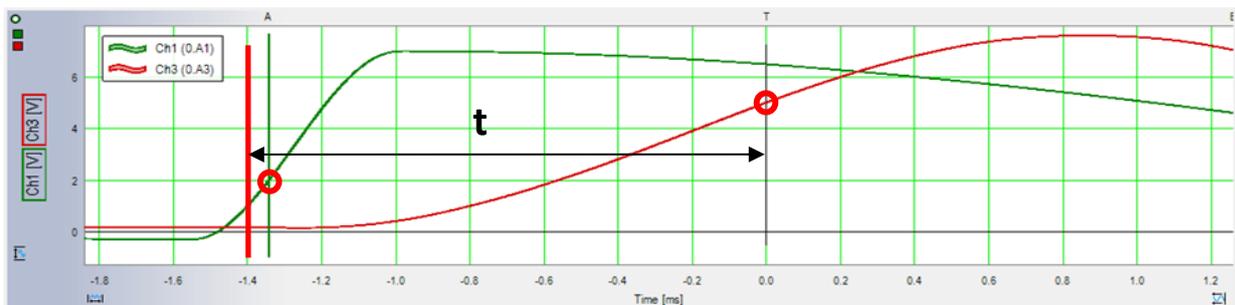


Settings for channel 1.

The trigger mode for the associated channel 3 (or 4) is determined by channel 1 (or 3). The comparator settings for these channels, Level, Hysteresis and Slope ("+", "-" or "±") can be set independently. .



Settings for channel 3. The trigger mode is determined by channel 1.

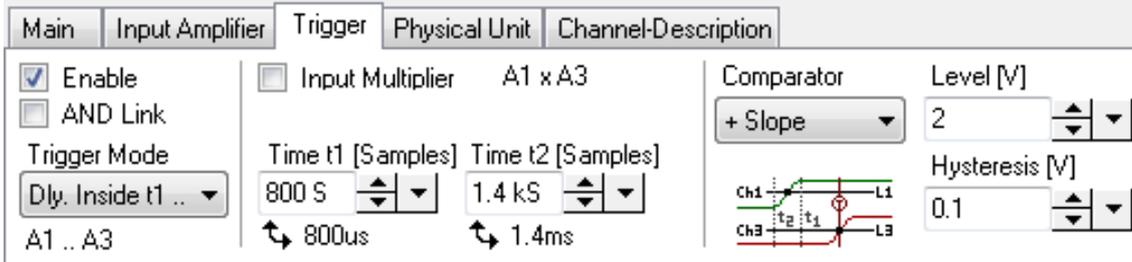


Delay shorter than t (delay < t).

11.5.5 Delay inside t1 .. t2

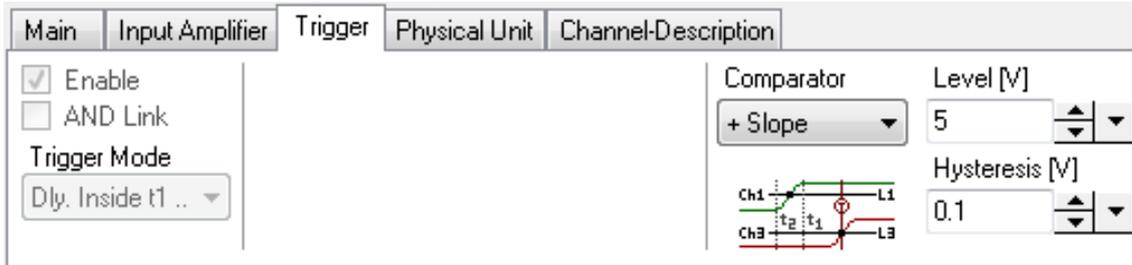
This trigger mode uses the signals of channel 1 and 3 (respectively 2 and 4). It captures when the **Time between** trigger condition of channel 1 and the condition of channel 3 is **within** the pre-set time limits t1 and t2. In this case, the system generates a trigger.

In addition to the times t1 and t2, the comparator settings (edge, level, and hysteresis) for the two channels 1 and 3 must be set.

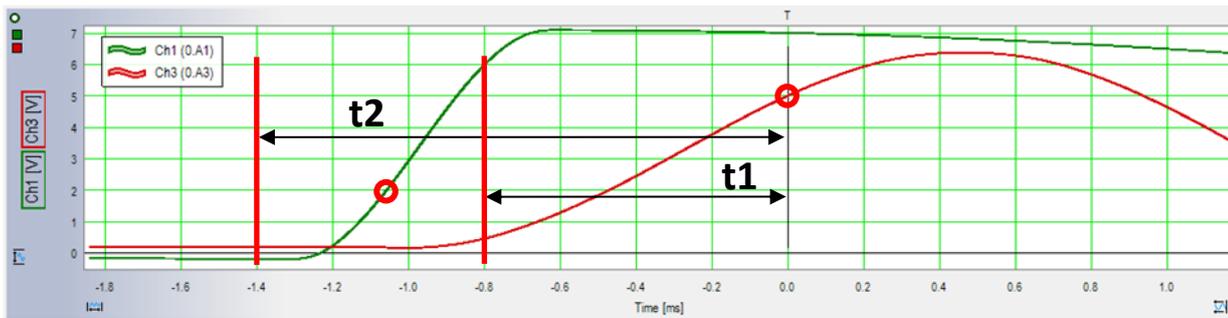


Settings for channel 1.

The trigger mode for the associated channel 3 (or 4) cannot be changed; this is determined by channel 1 (or 3). The comparator settings for these channels, Level, Hysteresis and Slope ("+", "-" or "±") can be set independently.



Settings for channel 3. The trigger mode is determined by channel 1.

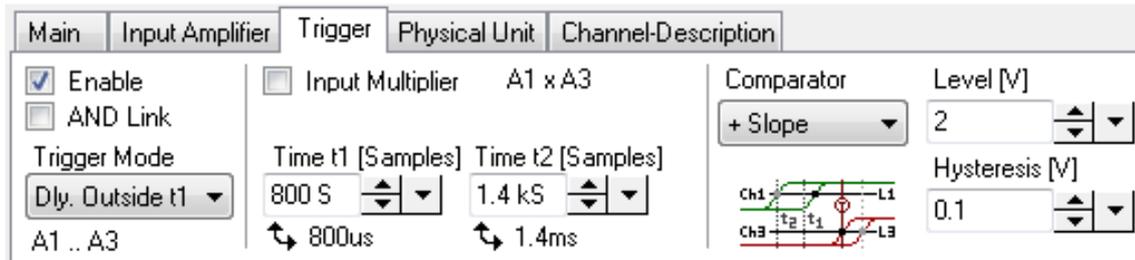


Delay between the two slopes is ca. 1.1ms (inside 0.8 and 1.4ms).

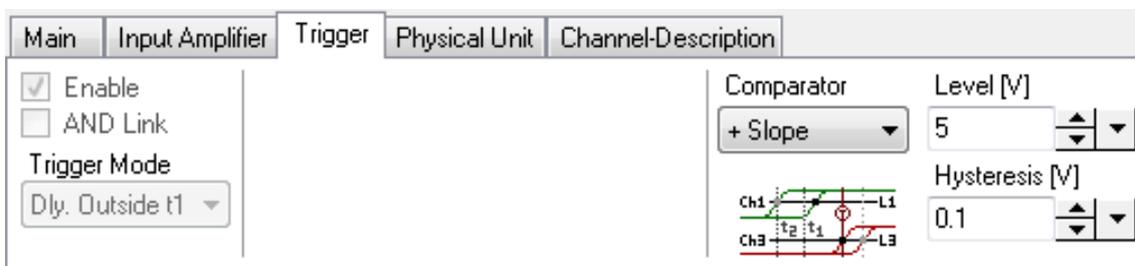
11.5.6 Delay outside t1 .. t2

This trigger mode uses the signals of channel 1 and 3 (respectively 2 and 4). It captures when the **Time between** trigger condition of channel 1 and the condition of channel 3 is **outside** the pre-set time limits t1 and t2. In this case, the system generates a trigger.

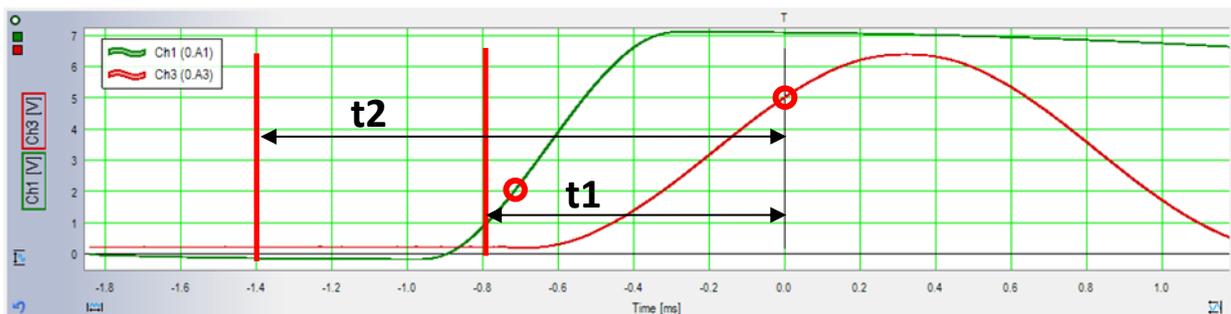
In addition to the times t1 and t2, the comparator settings (edge, level, and hysteresis) for the two channels 1 and 3 must be set.



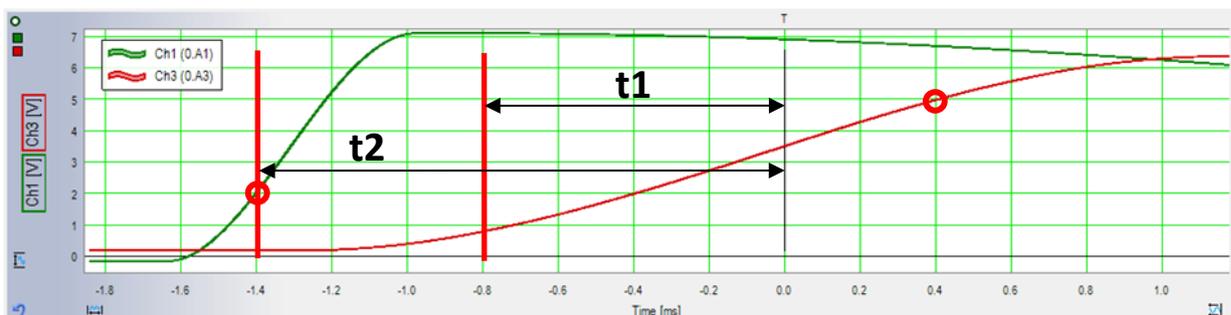
Settings for channel 1.



Settings for channel 3. The trigger mode is determined by channel 1.



In this picture, the delay is shorter than t1 (delay < t1).

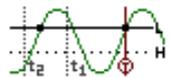


Delay is longer than t2 (delay > t2)

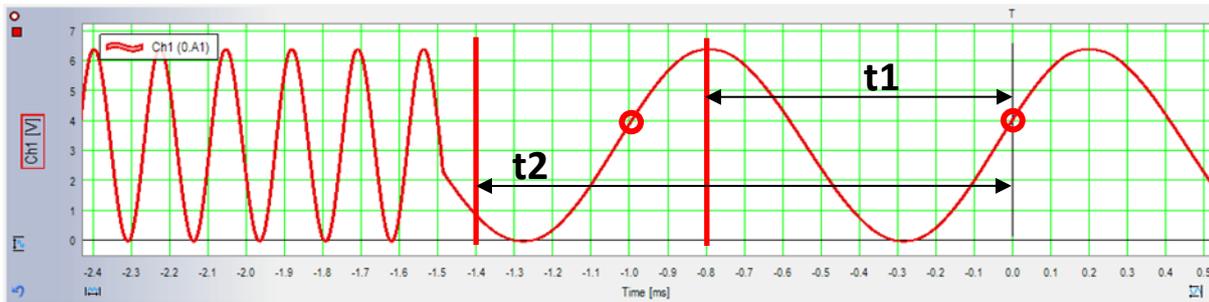
The trigger zero point is located exactly t2 behind the criterion for the start of the pulse (here pos. edge, level = 2V). At the trigger zero point, no trigger condition of the signal is fulfilled. The rising edge of the pulse crossing 5V trigger level is behind the zero-point t2, thus generates a trigger.

11.5.7 Period inside t1 .. t2

A trigger event is caused when the period is within the pre-set time limits t1 and t2.

Main	Input Amplifier	Trigger	Physical Unit	Channel-Description
<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Input Multiplier	A1 x A3	Comparator	Level [V]
<input type="checkbox"/> AND Link			+ Slope	4
Trigger Mode	Time t1 [Samples]	Time t2 [Samples]		Hysteresis [V]
Per. inside t1 ..	800 S	1.4 kS		0.1
	800us	1.4ms		

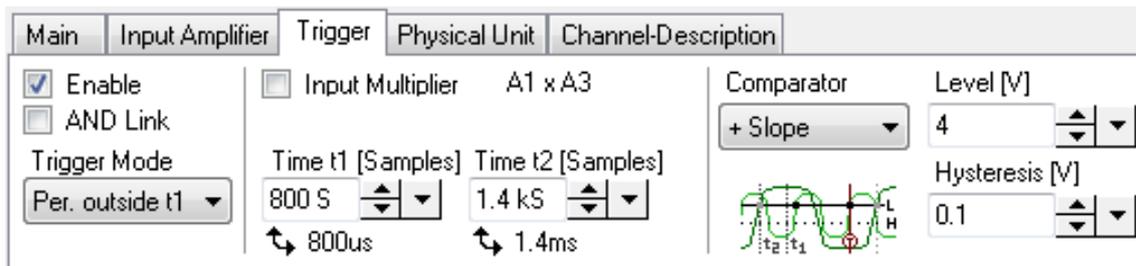
Settings for channel 1.



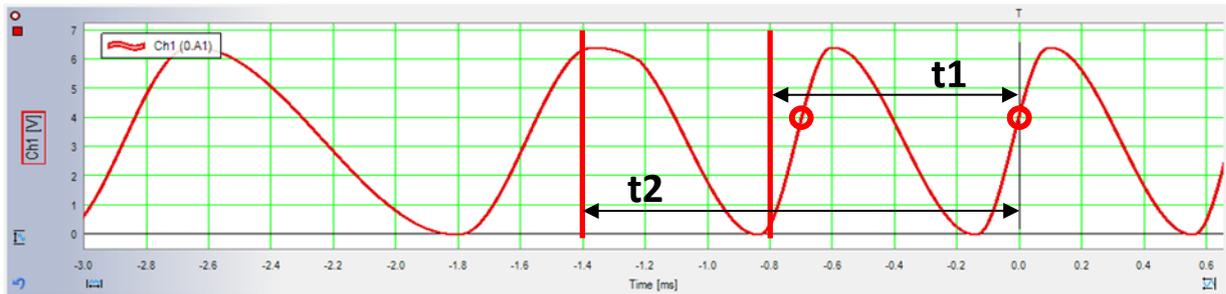
Period is ca. 1.0ms (between 0.8 and 1.4ms).

11.5.8 Period outside t1 .. t2

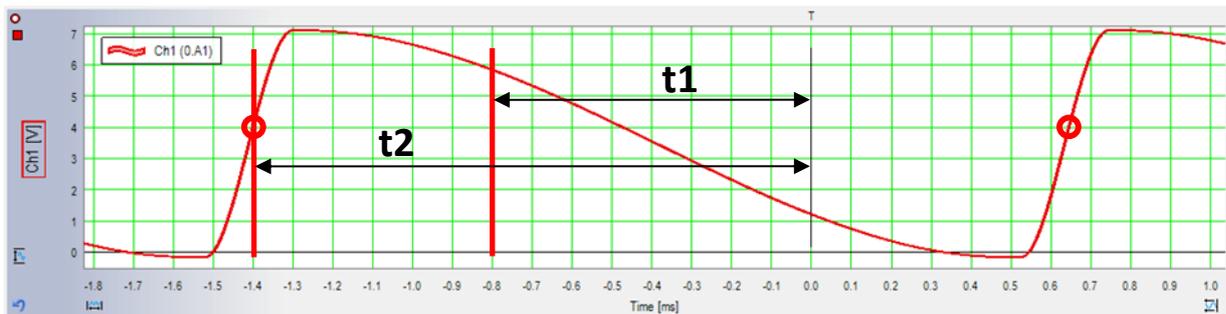
A trigger event is caused when the period is outside the pre-set time limits t1 and t2.



Settings for channel 1.



Period is shorter than t1 (*period < t1*).

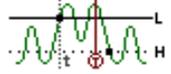


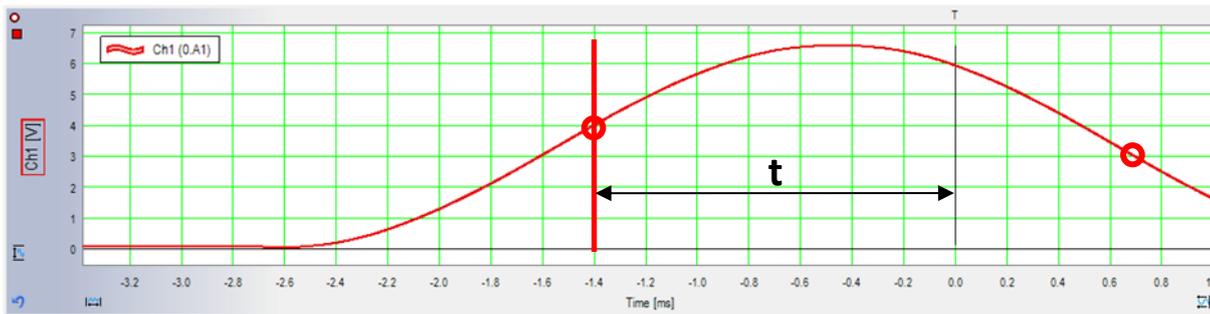
Period is longer than t2 (*period > t2*).

The trigger zero point is located exactly t2 behind the criterion for the start of the pulse (positive slope at 4V). At the trigger zero point, no trigger condition of the signal is fulfilled. The rising edge of the pulse crossing 4V trigger level is behind zero-point t2, thus generates a trigger.

11.6 Existing Trigger-Modes for Pulse / Period

11.6.1 Pulse > t

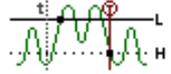
Main		Input Amplifier		Trigger		Physical Unit		Channel-Description	
<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> AND Link	<input type="checkbox"/> Input Multiplier	A1 x A3		Comparator		Level [V]		
Trigger Mode		Time [Samples]		Pos. Pulse		4			
Pulse > t		1.4 kS		Hysteresis [V]		1			
		↻ 1.4ms							



Pulse width is longer than t (pulse > t).

The trigger zero point is located exactly at time t behind the criterion for the start of the pulse (positive slope, 4V). At the trigger zero point, no trigger condition of the signal is fulfilled. The trailing edge of the pulse crossing the 3V trigger level, is behind zero-point t, thus generates a trigger.

11.6.2 Pulse < t

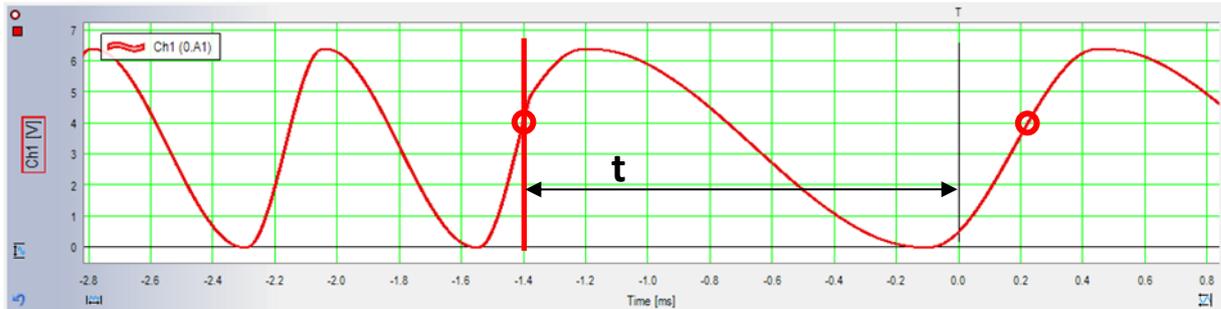
Main		Input Amplifier		Trigger		Physical Unit		Channel-Description	
<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> AND Link	<input type="checkbox"/> Input Multiplier	A1 x A3		Comparator		Level [V]		
Trigger Mode		Time [Samples]		Pos. Pulse		4			
Pulse < t		1.4 kS		Hysteresis [V]		1			
		↻ 1.4ms							



Pulse width is shorter than t (pulse < t).

11.6.3 Period > t

Main	Input Amplifier	Trigger	Physical Unit	Channel-Description
<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Input Multiplier	A1 x A3	Comparator	Level [V]
<input type="checkbox"/> AND Link			+ Slope	4
Trigger Mode	Time [Samples]			Hysteresis [V]
Period > t	1.4 kS			1
	↻ 1.4ms			



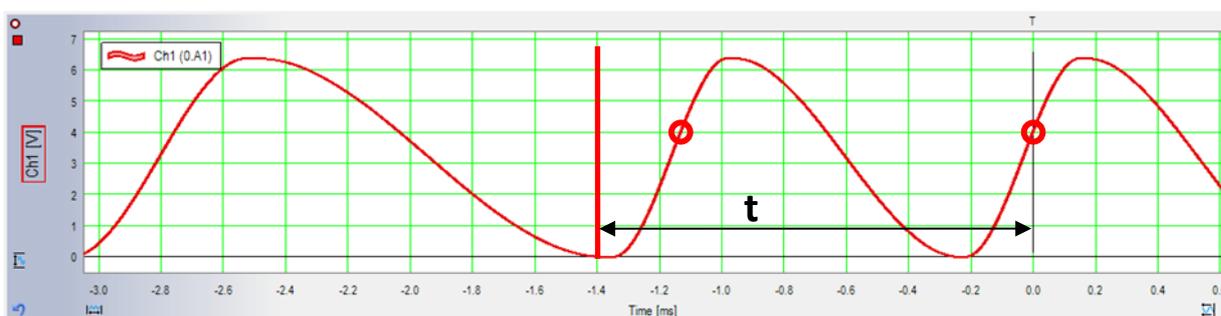
In this picture, period is longer than t (**period > t**).

The trigger zero point is located exactly at time t behind the criterion for the start of the pulse (positive slope 4V). At the trigger zero point, no trigger condition of the signal is fulfilled. The rising edge of the pulse crossing the 4V trigger level is behind zero-point t , thus generates a trigger.

The rising edge of the pulse crossing the 5V trigger level is behind zero-point t_2 , thus generates a trigger.

11.6.4 Period < t

Main	Input Amplifier	Trigger	Physical Unit	Channel-Description
<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Input Multiplier	A1 x A3	Comparator	Level [V]
<input type="checkbox"/> AND Link			+ Slope	4
Trigger Mode	Time [Samples]			Hysteresis [V]
Period < t	1.4 kS			1
	↻ 1.4ms			



Period is shorter than t (**period < t**).

12 Physical Unit

TranAX is capable of scaling a measurement in any selected user units (linear transformation).

The screenshot shows the 'Physical Unit' tab of a software interface. It contains the following elements:

- Navigation tabs: Main, Input Amplifier, Trigger, **Physical Unit**, Channel-Description
- Factor: Input field with value '1'
- Constant: Input field with value '0'
- Unit: Dropdown menu showing 'V'
- Buttons: 'Set 1:1', 'Set 10:1', and 'Scale Designer ...'
- Physical Input Range: '-5 .. 5V'

On this tab any change can be individually set for:

- **Factor and constant** for calculation
- The **physical unit**

The scaling calculation is as follows:

Physical unit = (measured value [V] * factor) + constant



The settings for the calculation must be made before the data capture starts.

12.1 Scale Designer

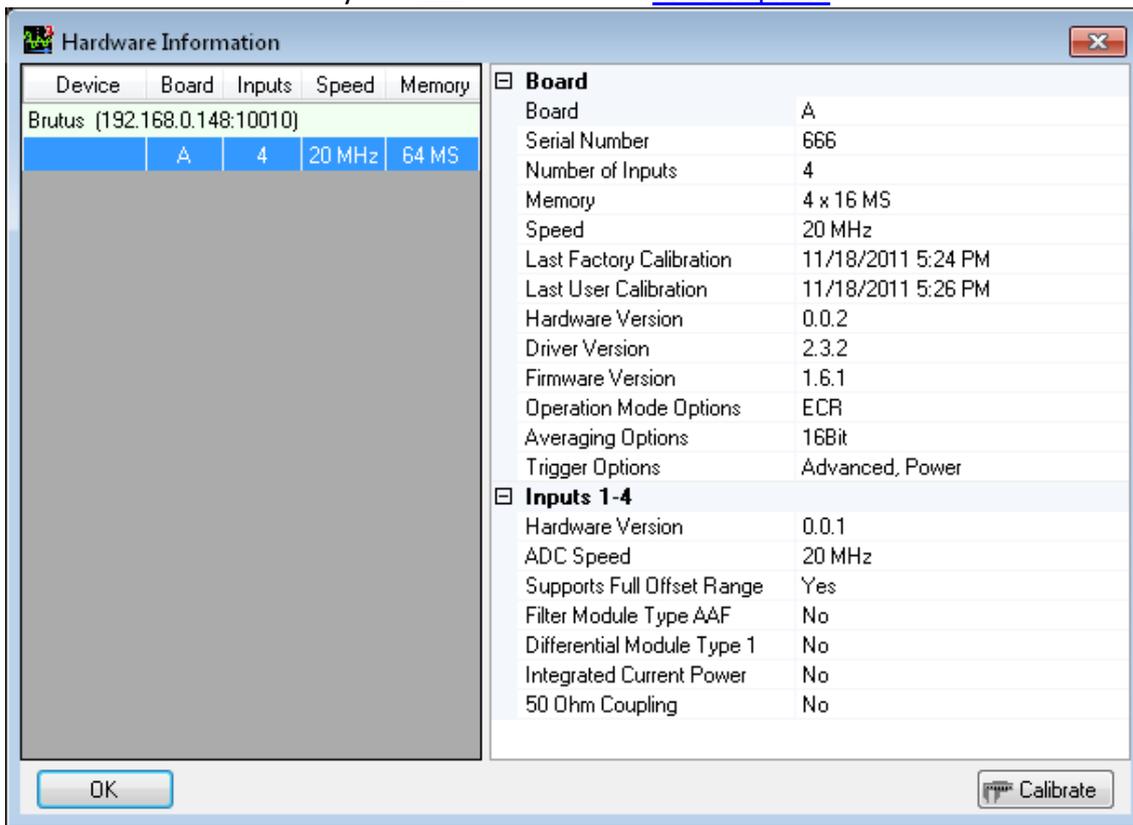
To set scale and factor from **two reference points**, the *Scale Designer* is used by selecting the appropriate command button.

The screenshot shows the 'Scale Designer' dialog box with the following settings:

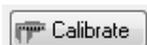
- Factor: 5.6
- Constant: -18
- Unit: °C
- Input Value 1: 0 V
- Input Value 2: 10 V
- Physical Value 1: -18 °C
- Physical Value 2: 38 °C
- Physical Input Range: -46 .. 10 °C
- Buttons: 'Ok' and 'Cancel'

13 Information Window

This window is accessed by the  button in the [Control panel](#).



It displays the current **installed hardware** (or in the case of an error, an appropriate error message). Either one or several on the local computer installed TPCX/TPCE-Modules or external devices like TraNET EPC or TraNET FE.



To calibrate the hardware, press the calibration button.
This usually takes some seconds.

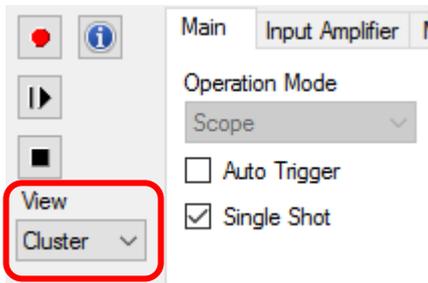


The hardware should only be calibrated after the modules have reached the operating temperature. Calibration during cold state can cause inaccuracies in the measurement!

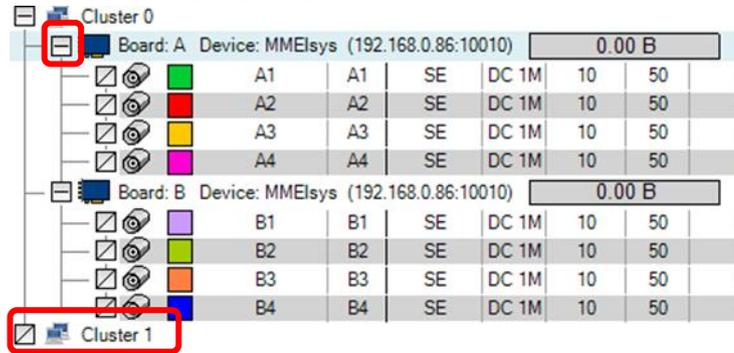
A calibration can be done and repeated at any time. Recording has to be stopped before calibration.

14 Cluster Configuration

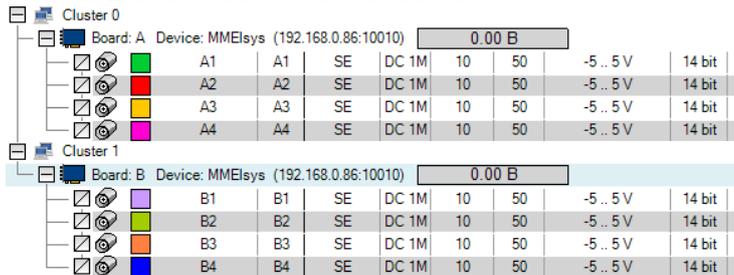
Clusters are used to **define groups of measurement channels**, which are running with the same configuration. With clusters it's possible to use different sampling frequencies and recording modes. Change View from "Normal" to "Cluster" to configure Cluster and Cluster settings



With the Cluster configuration mask, it is possible to configure different groups of TPCX/TPCE modules. **Initially all TPCX/TPCE modules are within cluster 0.** To add modules to a cluster, create first a new cluster, right click on left side node of the board icon, click "Add Cluster". This will create a new Cluster node.



Ad new Boards to this cluster via drag & drop. Left click and hold a board icon, drag it to the new Cluster 1 and release the mouse button.



Each cluster can be set individually to any operation mode with own recording settings (Sample Rate, Block Length, Trigger Delay etc.).

All clusters work synchronously concerning reference clock and trigger logic. This means: If one cluster detects a **trigger event**, then **all other clusters** would also react on this trigger, independent of the operation mode. This behavior should be considered when working with different operation modes (e.g. single shot and continuous mode with stop trigger).

15 Reference Pointers

In TranAX it is possible, **instead of curves from files to work with so called Reference Pointers**. Hardware channels can be assigned as well to these pointers.

With pointers it is possible to sequentially analyze single Tpc5 files that for example have been recorded in Auto Sequence mode. The reference pointer can be imagined as a place holder that can be used in a Waveform Display, Scalar tables or Formula editor. By assigning curves from a different file (or hardware-channels directly) only the data content is exchanged, while the profile (color, line thickness, zoom position, labels, etc.) will be retained.

This offers the **advantage** that the **planning** and **preparation** of a **screen lay-out**, e.g., curves display, scalar tables, texts, only once needs to be set up with the Reference Pointer curves. Subsequently all measurement curves can be assigned via Drag & Drop, without having to replace those in the lay-out or in a formula each time.

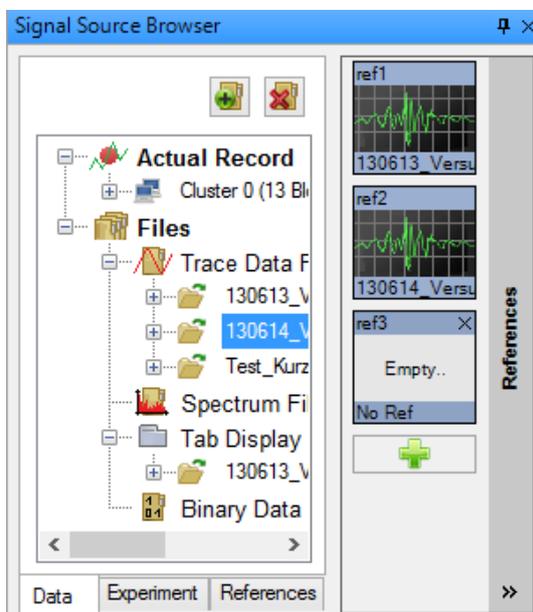


Copy files are not generated. The Pointer acts like an arrow referencing a selected Signal Source (file or hardware channel).

When such a **source file** is **removed** from the computer or **moved** then that pointer disappears, e.g. is empty.

15.1 Potential Sources for Reference Pointers

Handling of the Reference Pointers takes place in the Signal Source-Browser (🔍).



The Pointer Fields are found in the Signal Source browser next to the file-tree. They are visible by clicking the vertical grey bar on the right.

By clicking on the symbol  more fields (empty for now) can be added.

Only the lowest field can be removed. For that click on the small "x" in the field's top-right corner.

To designate a Pointer, the desired object (an entire file but also just single curves) can be moved onto a Pointer-field through **Drag & Drop**. Then, normally this Pointer-field will be moved via **Drag & Drop**, into a curve display.

The following objects are valid:

- **Stored files**, with the following formats being supported:
 - ***.Tpc5**: TranAX 3 Standard Format for measured curves in time domain
 - ***.TPS5**: TranAX 3 Standard Data Format for Spectrum and FFT curves
 - ***.TDP**: stored Tab-Pages in TranAX 3
 - ***.BDF**: binary Raw Data from Recorder and ECR-Mode
- **Hardware Channels**, all channels on the Control Panel respectively from an actual cluster. Thereby it is also possible to select multiple channels from the two first columns and via Drag & Drop deposit those on a Pointer-field.
- **Averaged YT and FFT curves** (@- resp. %-curves) several channels here as well can be selected and dragged onto a Pointer-field.

The designated Pointer-fields then can be via Drag & Drop moved onto a curve display. This relates always to all the curves designated to a given Pointer. When, for example only one curve is being dealt with, only that curve should be dragged onto the Pointer-field.

15.2 Reference-Pointers in the Formula Editor

Pointers can also be used in the Formula Editor.

With the File-command a curve in a Pointer-field can be accessed

```
trace = File(ref1,index) [.blkNo] ['markerNo']
```

The index designates the curve in the file (0...). Rather than to provide the full file name in the quotation mark only the term *ref* with the corresponding number needs to be given (e.g. "ref1").

In case only the first (no: 0) curve in a Pointer needs to be accessed, the formula also can be simplified:

```
trace = ref2 [.blkNo] ['markerNo']
```

This command relates to **trace=File (ref2 , 0)**

Optionally, identified Block- and/or Marker-numbers also can be allowed.

Likewise, the following functions are permissible:

```
val = FileIndexExist (ref1, 2) ; val = True or False
nTrc1 = NTracesInFile (ref1); is synonymously to Length (ref1)
nTrc2 = Length (ref2)
```

Ref, in the formula editor is used as a key-word.

15.3 Reference-Pointers in Autosequence

Pointers also can be used with Auto Sequences. At command **Save** instead of the file name, the corresponding Pointer (**without name extension**, e.g. TPC5, etc.) must be indicated.

`Save ref1, 0A1-4`

With this method it is possible in an Auto-Sequence-loop to archive measured curves (in number-progressive files) and simultaneously use the actual measurement data for display or extended calculations in the Formula editor.

Auto Sequence example:

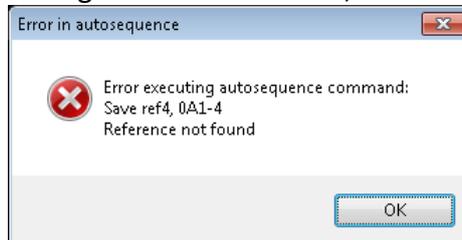
```
Repeat 10
Start Recording
Wait on EOR
Save xy_#.tpc5, 0A1-4 ; Generate a measurement series
Save ref4, 0A1-4      ; Generate (overwrite) a file
                        ; "ref4.tpc5" and allocate to the
                        ; corresponding Pointer

Calculate
Wait for Calculations
Next
```



The expression "ref" serves as keyword and is accordingly checked!

When a Pointer-field with the corresponding number (e.g. "ref4") is not available in the Signal Source Browser, the following error message appears:



By clicking on the symbol  in the Signal Source Browser, more Pointer-fields can be added.

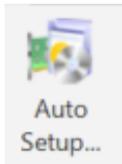


In Auto Sequence mode the curve is stored physically as a file (e.g. "ref1.tpc5") in the "data" register of the actual Experiment and assigned to the corresponding Pointer.

16 Auto Setup

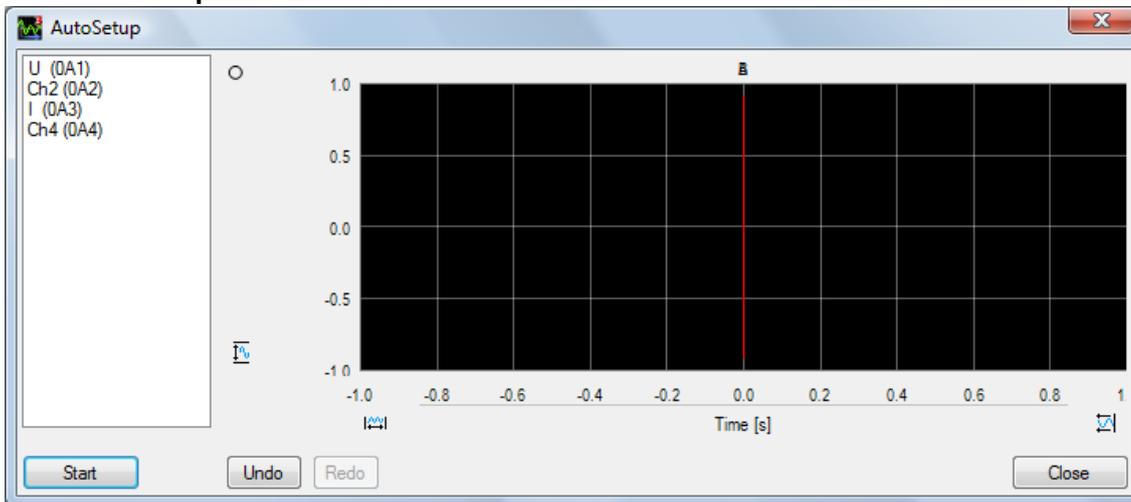
If a signal connected to the Transient Recorder is **unknown** and a quick set up is required, the built-in auto setup function may be of some help.

The auto setup function is looking for the **vertical input range** that the signal requires sets the **sample rate to the maximum** and the **record length to 20ms**.

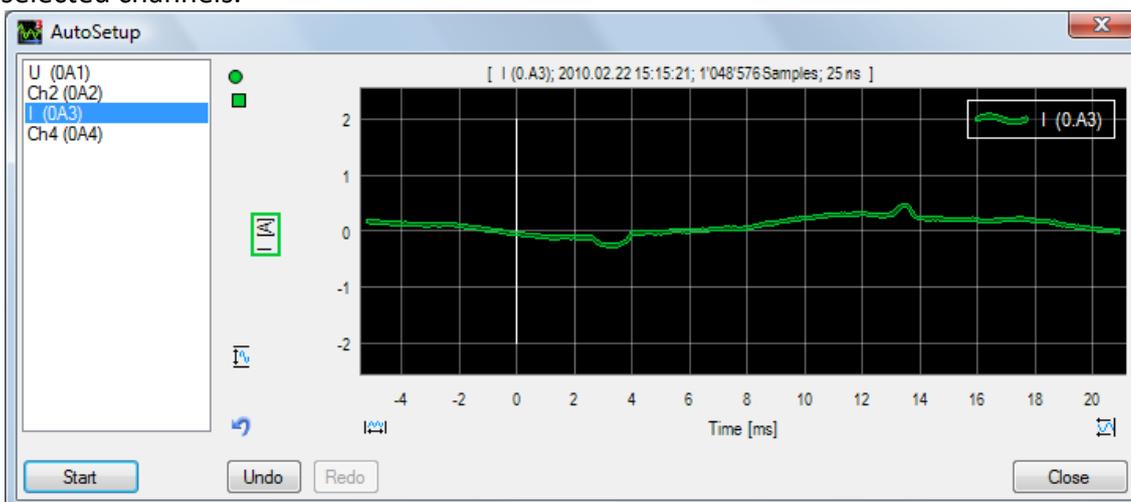


In order to quickly find the vertical range and to get the instrument to capture a signal without knowing the trigger condition the Auto setup process can be started, in the Ribbon bar, go to the tab "Measurement" and click "Auto Setup...".

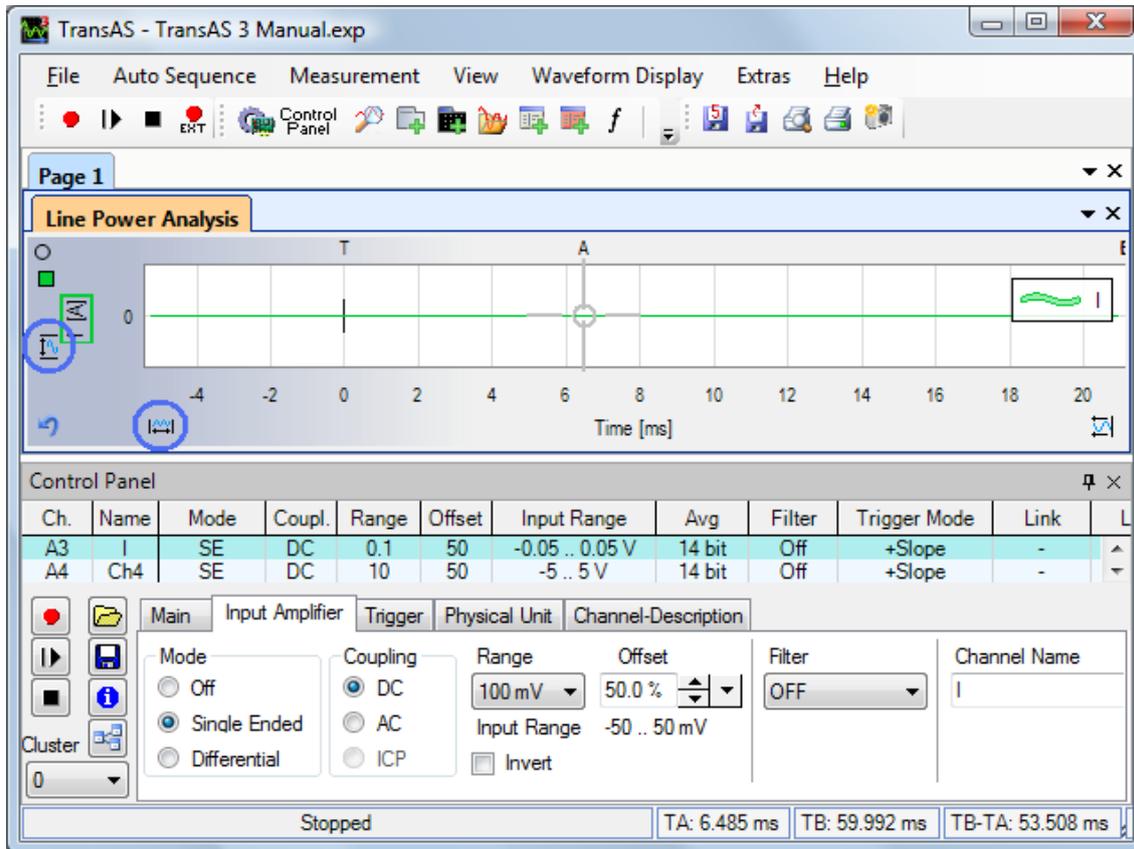
The "**AutoSetup**" window allows now to select all or a subset of the available channels.



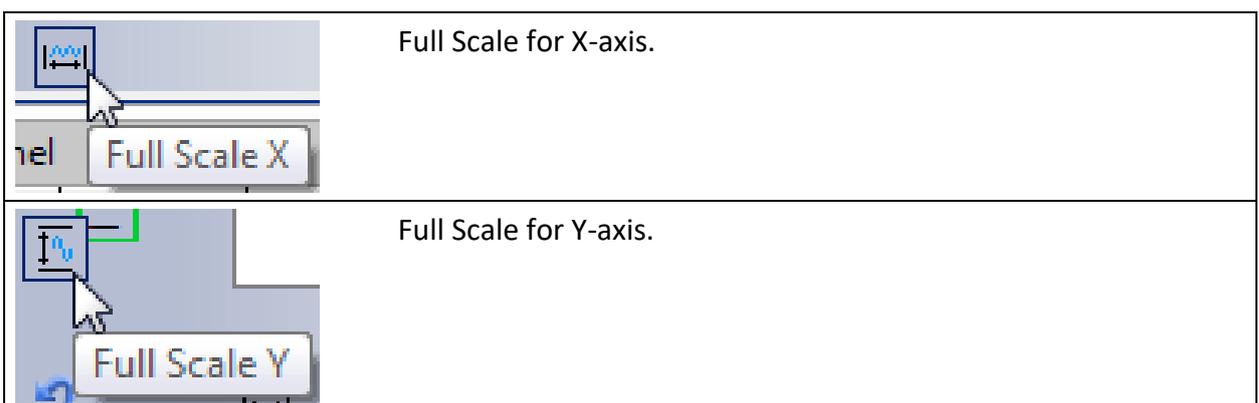
With a click on a channel this line is selected. Multiple channels can be selected and toggled on/off by **ctrl-left mouse click**. A click on the Start button starts the Auto setup process for all selected channels.



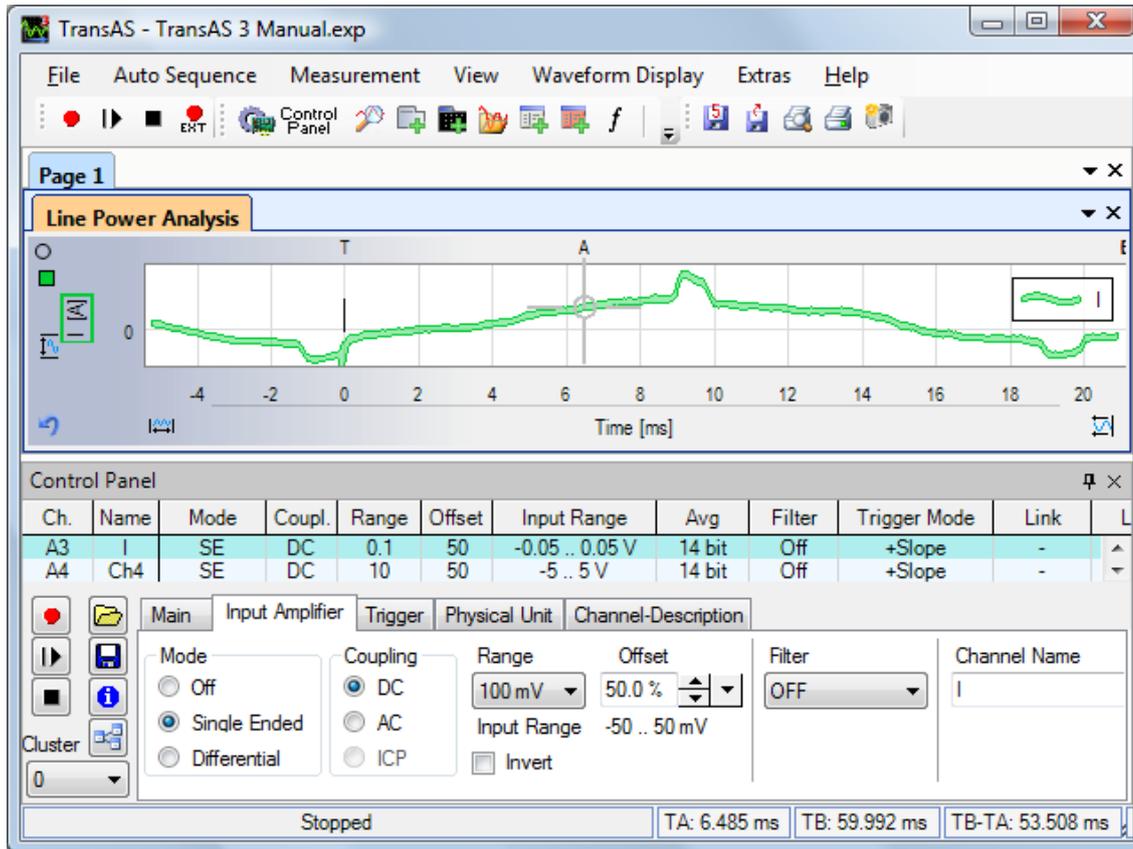
Once Auto setup is finished it displays the trace in the new scaling. At this point it is possible to **undo the modifications** to the hardware configuration by hitting **Undo** or to **accept** the new settings by clicking the Close button.



The vertical range changed now from 100 V to 100 mV while the trace is displayed in the same way as before. **The Waveform Display zoom is not updated automatically** but it is updated manually with the two Full Scale buttons within the Waveform Display., Full Scale X and Full Scale Y.



After the two buttons above were clicked the scaling in the Waveform Display is updated and reveals now more details about the shape of the trace.

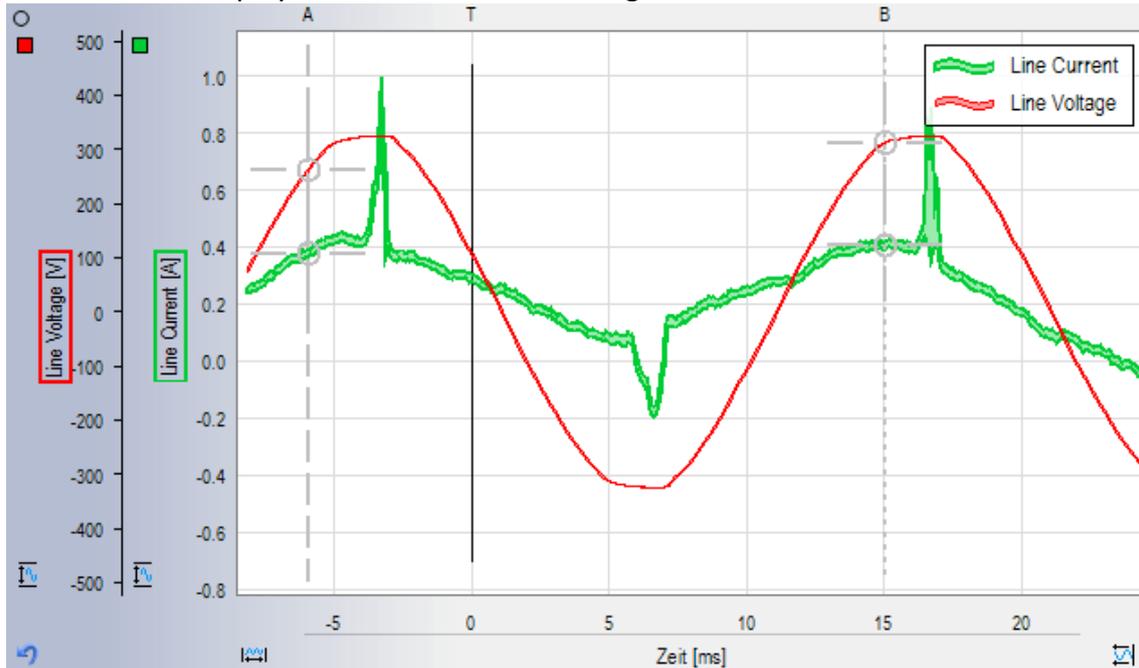


With the additional information about the signal it is now easier to configure the trigger with trigger level and -slope and to compensate for vertical offset if needed.

The trigger settings of the selected channels are automatically set to positive slope and the level is centered between the minimum and maximum of the captured trace. This is only the case if trigger of the selected channel was enabled on beforehand.

17 Waveform Display

The waveform display visualizes the recorded signals and traces.



17.1 Organizing and arranging

The waveform display shows the traces of recorded or actually recording signals. Simply by dragging a channel from the [Control Panel](#) or from the [Signal Source Browser](#) to a waveform window will display it as a signal curve. If several signals are overlaid in a waveform display, the signals are distinguished **by different colors** and the signals will be represented by small colored boxes on the left of the waveform display. Every waveform belongs to a Page. To add a **new waveform display**, hit the icon  or go to menu "View" / "New Waveform Display". Arranging waveforms can be performed in the same manner as arranging sub windows simply by selecting and moving the tabs.

<u>C</u> lose	Ctrl+Shift+C
Prominent	Ctrl+Shift+T
<u>R</u> ebalance	Ctrl+Shift+R
<u>S</u> et Title..	

If you open more and more waveforms, keep your workspace well arranged by **right clicking** on the page or waveform tab to open a context menu. Here you can close your displays, set the title or arrange your waveform windows vertically or horizontally.

17.2 Zooming



You can **zoom into an area** simply by pulling a **box** with the mouse pointer over the area.

Click on the upper left corner of the visible section, move with pressed mouse button to the lower right corner and release the mouse button.



Or click with the left mouse button on the axis labelling. A **zoom pointer** will then appear and by moving up/down and left/right respectively you may zoom in or out. You also may use the mouse wheel for this.

17.3 Moving traces

To move within the waveform just press and hold your **right mouse button** and move into the desired direction. Secondary, you can move your mouse pointer over the axis units and a double sided arrow  will appear. Click and hold your mouse pointer and move as long as required. Again, while your mouse pointer is on the axis units you may use your mouse wheel to move the traces.



Time range shifting of the traces (X-axis) with the mouse wheel is also possible with the mouse cursor in the waveform display window while simultaneously pressing the shift key.

17.4 Set to full scale

There are two buttons for this function:

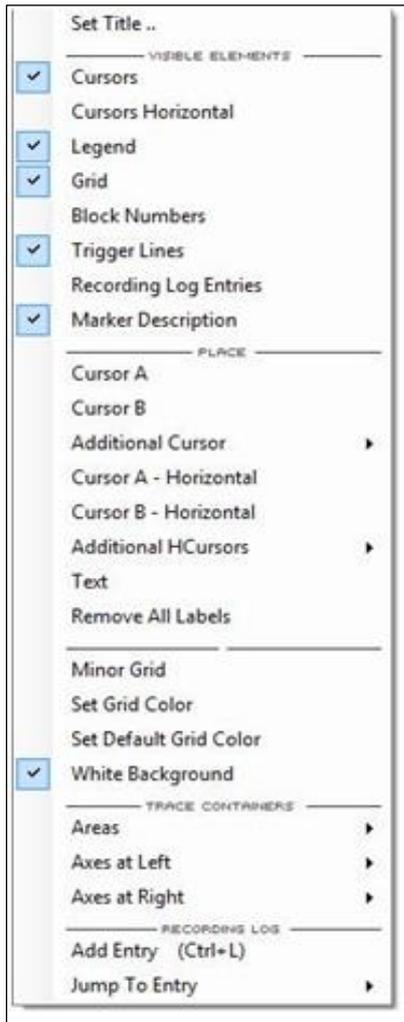
	Switch Y axis to vertical full scale (for each axis individual)
	Switch X axis to horizontal full scale

You can also **redo or undo your changes** in the waveform display by the following buttons:

	Undo the last changes
	Redo the last changes
	Auto scroll: Enabled, the waveform display will automatically scroll horizontally with the signal while recording in continuous mode (or ECR with dual mode). Disabled, it will pause the scrolling but not the acquisition!

	Pin Option to minimize a window as a tab (Signal Source Browser or Control Panel)
---	---

18 Display options in Y/T Waveforms



The following display options can be activated by right clicking in the **Curve-Display-Window** or extracting it via the menu **Curve-Display**.

Several parameters, i.e. Cursors, Grid, Trigger lines, etc. can be switched on or off.

Cursors A, B (or additional ones) can be placed on their actual positions with the mouse pointer. For that "Cursors" must be active.

The same is valid for the horizontal cursors.

The number of axes (left and right) are self-defined via a right click on the Y-axis.

Via "Add Entry", Recording Log Entries can be carried out at the actual mouse position, also afterwards.

The Curve-Display-Section is being placed directly at the corresponding Recording Log-Entry through "Go to Entry".

Also, the background color can be chosen on beforehand via Menu "Extras/Settings/User Interface..."

Entries in the curve display window can be carried out via "Text". Right clicking the text entry allows to modify the text type (font, size, etc.).

18.1 Legend

This legend can be enabled or disabled (Hide). If the legend isn't visible, click menu "**Waveform Display**" / "**Legend**".



Hide/show the legend. As you move with the mouse pointer over the legend, hold the left mouse button and move the legend to the desired position.

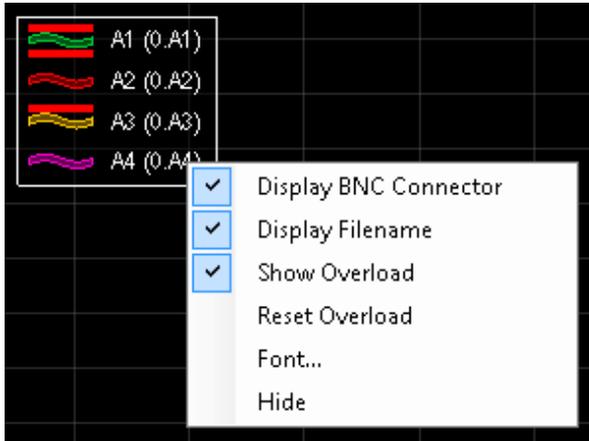
If Display **BNC Connector** is set to off, only the Channel Name will be shown.

In case a trace is either calculated by the Formula Editor or imported from TPC-file, the filename may be displayed.

The item "Show Overflow" signals if the amplifier of the channel had a positive or negative overflow. A line over or below the trace symbol will show the overflow.



If the current trace has a positive or negative overload of the ADC it is indicated by an orange bar below or above the waveform icon. If an acquisition is still ongoing, and there was an overload since the beginning of the acquisition, this is indicated by a red frame around the orange bar
A recorded overflow (yellow line) can be reset by clicking "Overload reset".



By clicking the menu item "**Overload Reset**", the yellow marking above and below the trace will be removed. These markings symbolize an overflow capture.

Resetting an overload event can be useful for example with long time measurements in Continuous mode. After a distortion or recalibration of sensors, the apparent overload can be reset. In a **later check-up** of the system you can then see if there was **an overload again**, or that all signals were captured without any distortion.



Overload means that the measured signals were **outside of the dynamic range** of the ADC. Example: Range is set to 2V, 50% offset, so the dynamic range will be between -1V and 1V. If there appears a signal burst at 1.5V, an overload will be detected and then marked with a **red bar during measurement**. A positive overload will be visible as a bar at the top, a negative overload as a bar at the bottom. A **yellow bar** means there was an **overload in the past** in the measurement.



By clicking the "Overload reset", an **entry will be written to the Recording Log**. Thus, all manipulation during the recording are logged for further documentation or analysis. Click Menu "**View**" / "**Recording Log**" to get a list with all entered logs.

The **letter type of the legend** is user definable. It is thus possible to select a small font such that the legend stays slim and does not take up too much display space.

18.2 Text Entries in Waveform Display

Via "Text" comment entries can be made in the waveform windows. By right-clicking on a text entry the font can be adjusted.

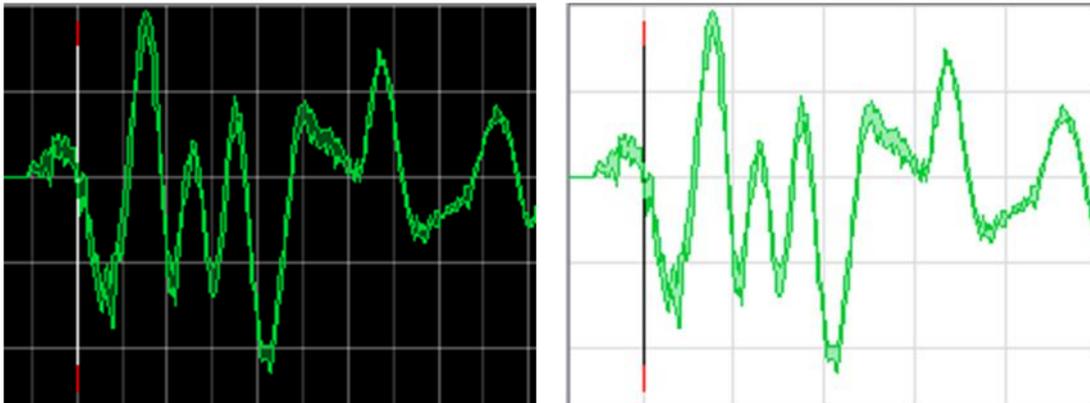
With text being added in the trace window, calculated single values in the formula editor can be linked. The Result-Names of those have to be written in angle brackets. Example: **<VarName>**. In the place of *VarName* the Result-Name must be written. In case that doesn't exist or has not yet been calculated "Not Defined" will be shown. Before and after **<VarName>** arbitrary text (shown below) can be written. In case *VarName* a number result is, it can happen that it is written with too many characters. To prevent that, the formula **StringFormat()** can change the number into a string (e.g. `xyzStr=StringFormat(xyz,"0.00")` ==> Text in Window: "... <xyzStr> ...").

18.3 Grid

A fine grid can be switched on. If needed, the grid can also be disabled. Furthermore, the grid color can be chosen. By the menu "Extras" / "Settings" / "User Interface" the grid color can be predefined.

18.4 Background color

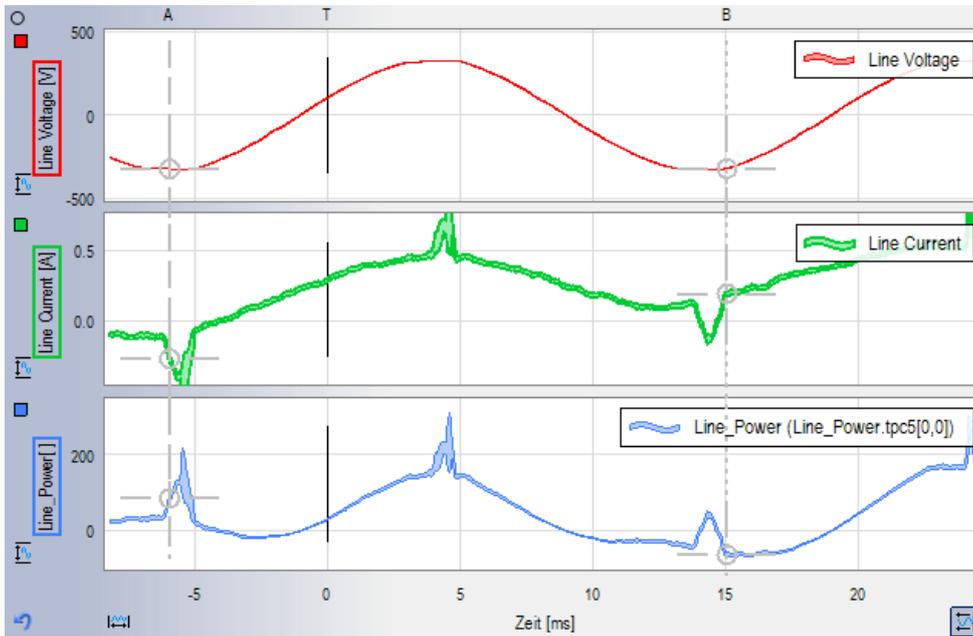
Background color of a waveform can be set to white or black



By the menu "Extras" / "Settings" / "User Interface" the background color can be predefined.

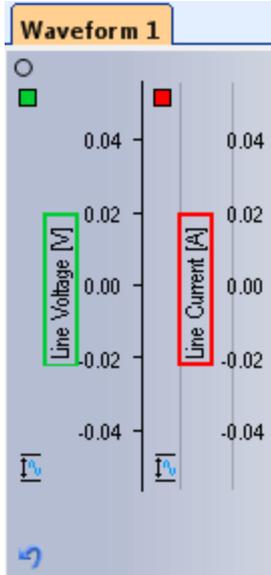
18.5 Areas

Instead of displaying the signal curves overlaid in one waveform display, they can be displayed in different areas. There are as many as 16 available areas within one waveform display.



18.6 Y Axes

18.6.1 Number of Rulers Left and Right



In case several signal curves are overlapped in one waveform display, it might be of interest to change the view parameters (zoom, x/y-axis view) for each channel separately. After adding more rulers to the display, just pull the signal curve indicators (little colored square boxes to the left of the waveform) to the new ruler to associate them.

You also may set up your waveform on beforehand with the axis and rulers and then pull the signals directly from the control panel or the signal source browser to the corresponding ruler.

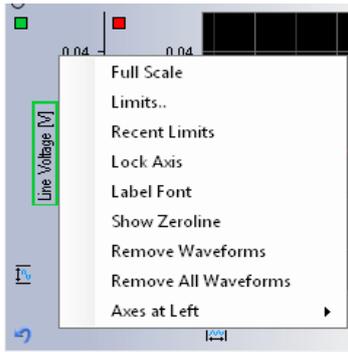


Up to 12 Y-axis, left and right, can be set up



Prepare first the number of rulers and axes to add traces from the Control Panel or Signal Source Browser.

18.6.2 Locking of Y Axes

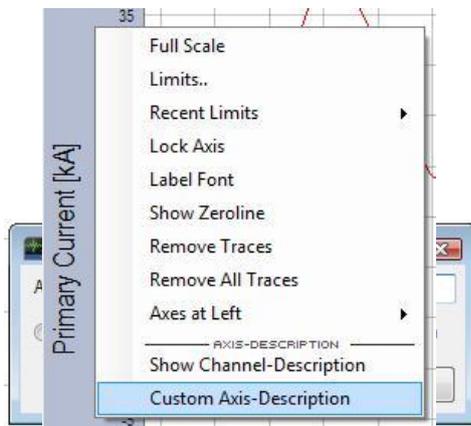


If using **several Y-axes**, it's possible to lock them. Right click on one of the Y-axis and click "Lock Axis" in the popup menu.

If one axis will be zoomed, all other axes will do the same now.

18.6.3 Labelling of Y Axes

The labelling of the Y-axes can also be customized.



To save your prepared Waveform display settings, click "**File**" / "**Save Layout..**". The layout with all pages, waveforms and settings for the waveforms will be saved.

18.7 Visualization of Traces



Each trace will be represented as a **small box** on the left hand side of the waveform window. A filled box will make a signal visible; an **empty box** will hide the signal. Furthermore, you can right click on the button and delete, change the color or display markers. To change the order of the boxes (from top to bottom) simply click on the box, **hold the left mouse button** and release after moving the mouse-pointer downwards.

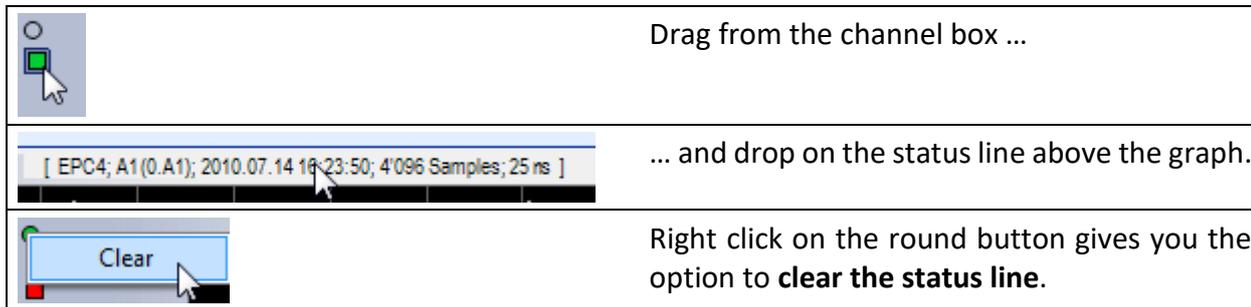


Deleting a signal from the waveform display will not delete the signal data. It will just remove it from the waveform display. However, the curve properties (color, labels, Y-scaling etc.) will be lost!

By dragging & dropping a channel from the [Control Panel](#) or from the [Signal Source Browser](#) to the waveform display, the trace will be displayed again.

Every Waveform Display has space for one **status line** that can be turned on/off with the little **round button** in the upper left corner of the window. The status line takes the information from any of the displayed channels. With drag from any of the small boxes described above and drop right above of the waveform graph the status line will be displayed with the actual settings of the chosen channel.

The information in the status line may be useful information for a screen dump or a report as it includes the name of the instrument, the name of the channel, the timestamp of the acquisition, the number of samples in the trace and the time interval between the samples.



In case the status of another channel should be displayed, just drag & drop on top of the previous one.

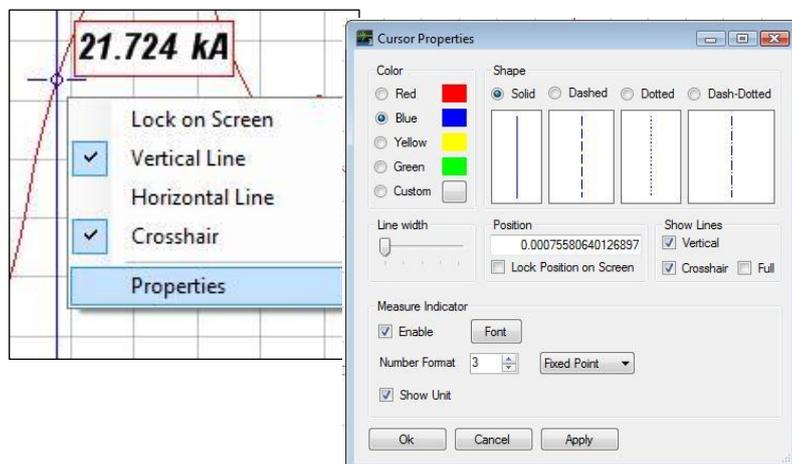
If the status line is not needed anymore, it can be turned off with a click on the round button described above or the content of the status line can be deleted with a right click on the round button followed by a "Clear".

18.8 Cursor Properties

When the mouse is placed over a cursor its property mask can be opened by right click.

Appearance and other characteristics of the cursor can be set there.

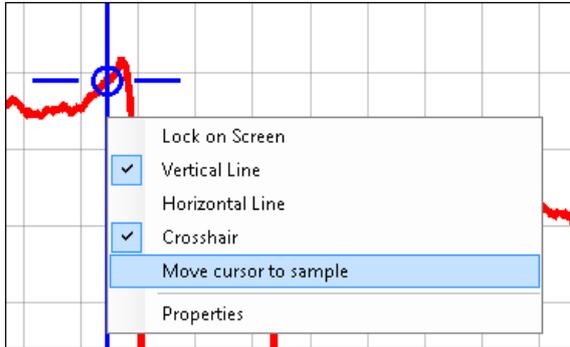
If the "Measure Indicator" is enabled, the Y values at the cursor position will be displayed for each curve.



In the menu "Extras -> Settings" in the rubric "User Interface" it is possible to select show cursor letters (A, B, C ...) at the top inside the curve display or on the overhead status line (the status line can be hidden via the small circle-symbol top-left).

18.9 Cursor on Sample points

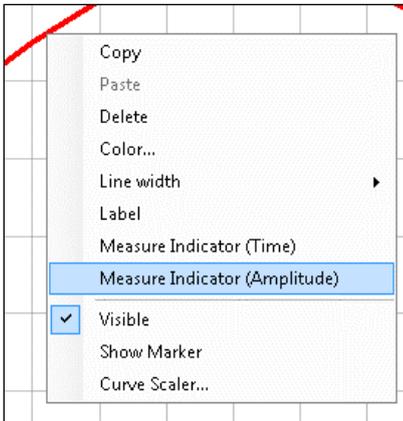
All cursors individually can be adjusted such that they are fixed on a single sample point. By right-clicking the cursor, "Move cursor to sample" can be selected from the menu.



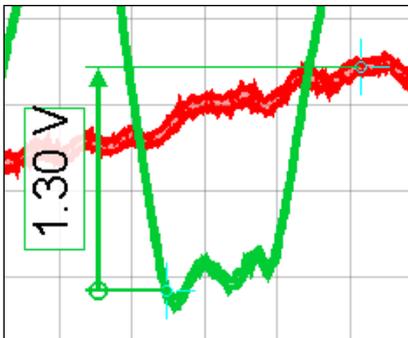
The cursor will then jump from sample point to sample point. In between values do not exist, or only then when more curves with different sampling speeds are shown in the same waveform window.

18.10 On-Curve Measurements

Next to text labels now also **Time and Amplitude measurement values** can be placed on the curves.

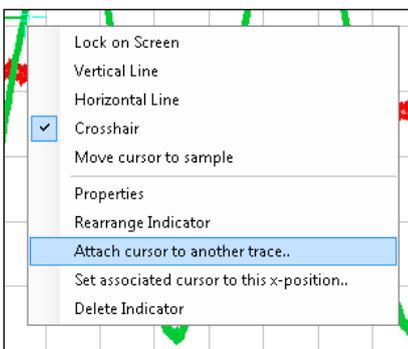


Right clicking on an actual curve gives the proprietary menu text where choices can be made.



When the location of the desired measurement on the curve has been picked, then the cursors can be **expanded** along the X-axis. The difference in value between cursor 1 and cursor 2 (see arrow) is thereby displayed in a text label in between the cursors.

The text and number format as usual can be set by right-clicking on the text via "Set Text...".



To position a cursor on **another curve**, right-click on that cursor and select in the upcoming menu "Attach cursor to another trace..". Then in an upcoming dialog box the desired signal can be selected.

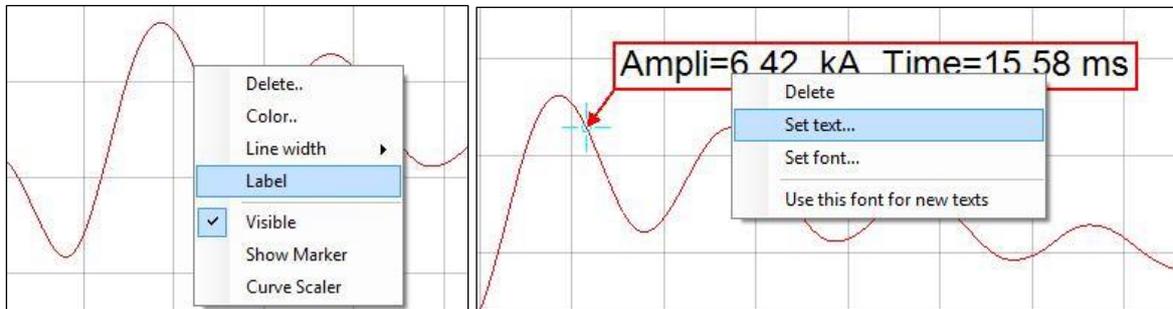
This way the amplitude differences of two signals are determined (this usually only makes sense when both signals are of the same measuring unit).

By right-clicking on "Set associated cursor to this x-position" the other cursor will be placed on the same X-position.

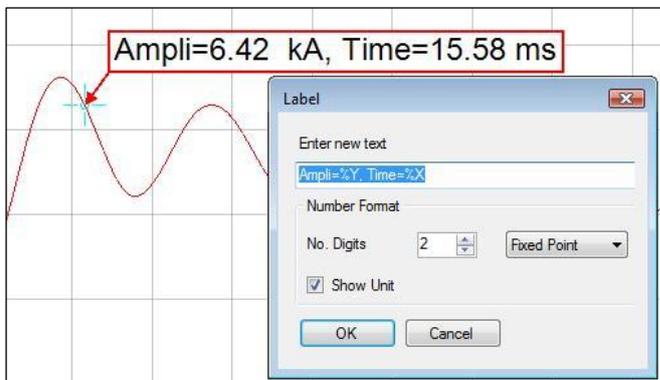
When clicking the icon  ("Move Cursors simultaneously") measurement cursors will move synchronously. Then the difference of both curves will be shown on a given position on the X-axis.

18.11 Labels on Curves

Curves can be characterized with so called "labels". Bring the mouse to the position on the curve where the label must appear. Right-click, then click on "Label" and, by default, the curve will be flagged with the amplitude value in the same color.



In the menu that appears when right-clicking within the label, click on "Set text" and an arbitrary text can be inserted.

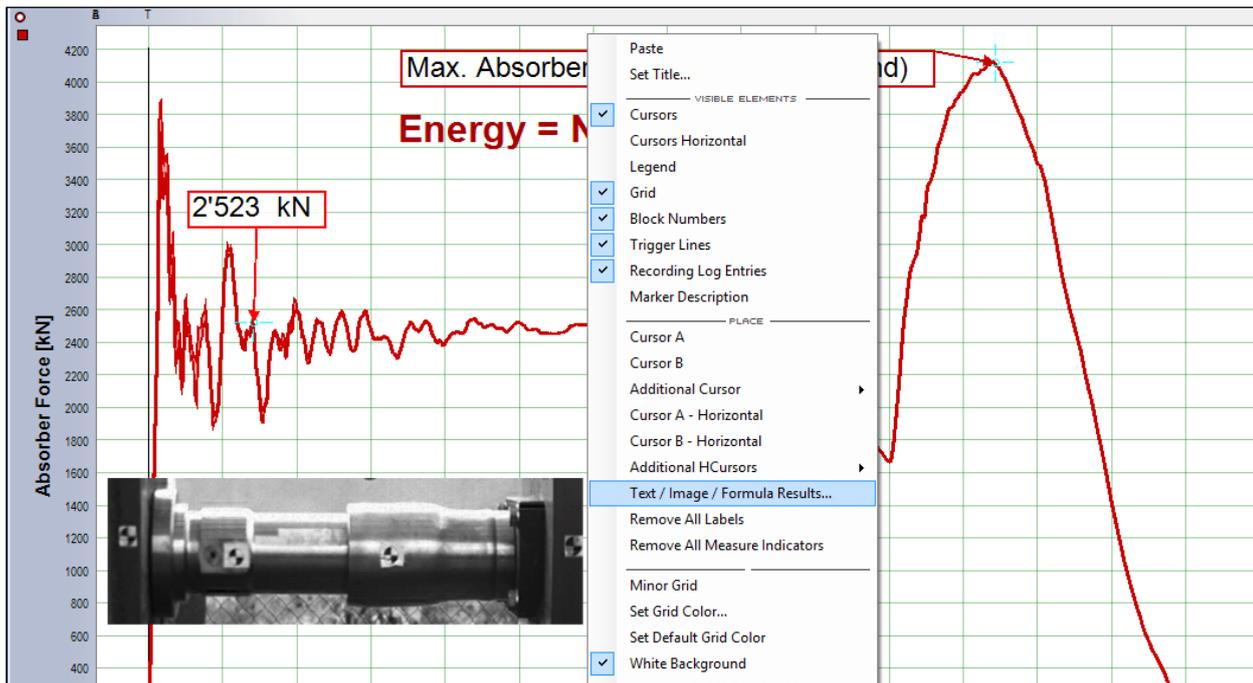


The key notations "%Y" respectively "%X" are being replaced by the values Y (amplitude) and X (time). Also, the number format of Y respectively X values can be defined here.

The properties of the crosshair-cursor-label can be defined in the same way as with any cursor.

18.12 Adding images and formatting of calculation results

Via a dropdown-menu (right click on a curve display window), an image can be added (like a text).

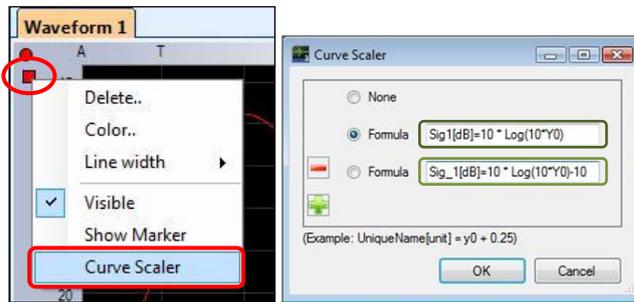


By clicking on **Add Image**, the desired image can be searched for and selected. When selecting, usually, first the files in the *images folder* of the actual Experiment are presented. But then, also by navigating an image can be selected somewhere and added.

In the same menu also the number-format of the, via the Formula-Editor calculated results, can be set.

18.13 Y-scale adjustments of curves

Each individual curve can be converted with a formula.



The user is free to select any formula. On beforehand multiple formulas can be defined. The selected formulas are activated by clicking the corresponding button in the formula mask

This formula is then part of the curve's properties (like the color). With moving or copying (via Drag & Drop) into another

window also the scaling formula is moved along.

Naturally only calculations with regard to amplitude values are valid (no X-axis values). The conversion calculation only deals with the curve on display in the curve window and possibly a scalar table as well. **Curves are always stored unscaled.** Scalar table functions take into account the Y-scaling of those curves (incl. name and unit).

In situations, where the formulas cannot give results (e.g. LOG of negative values) no curve will be drawn.

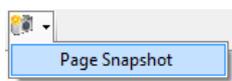
The calculations succeed on the basis of Raw Values **y0**. Raw Values also include, converted values, possibly obtained via the Formula-Editor or those that through relevant settings under *Physical Units* in the control panel have been defined before recording.

The new channel name (ChName[Unit]= ...) cannot be a Keyword from formula editor (e.g. Cos, Sin, If, for, to, as, Pi, etc.). With errors in the scale adjustment formulas, calculations are simply suppressed. The Raw Values will remain.

When errors appear in the scale adjustment formula this icon  will be shown. Calculations are suppressed with error-containing formulas.

18.14 Snapshot

By clicking with the left mouse button on the icon  the image of the **actual Waveform** will be copied to the clipboard of the PC.



By clicking the mouse on the arrow in the icon, the screen content of all windows in the **actual page** (not only trace or scalar windows) can be copied to the clipboard.

Under "Extras / Settings / User Interface / Snapshot", "Bitmap" or "Vectorized" can be selected. Additionally, the size, the display section should have afterwards, can be chosen.

The size of multiple windows in the page are proportionally adjusted. All text entries are stored in-tact. Attention should be given to the position of user specific text entries. Otherwise it may be placed awkwardly or entirely cut off.

To store the display intermediately with a white background can be selected here also. In case a screen with black background is used, trace colors should be selected darker.

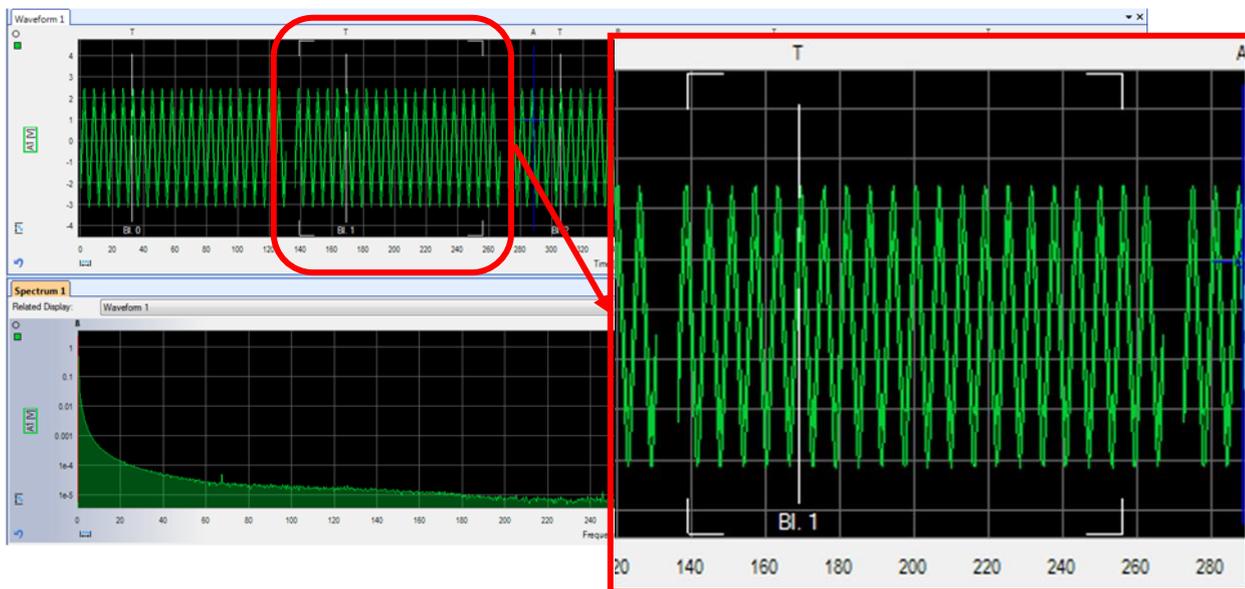
18.15 Analysis of Multi Block records (Block Jumping)

For easily moving (jumping) from one recorded block to another, a "Block Jumping" control is available. This can be used for analysis of Multi Block- or ECR Records.



Select the waveform to set the focus for these traces and click on the icon in the [toolbar](#) to move forward or backward. Alternatively, it's also possible to use the keyboard: "Page Up" for moving forward, "Page Down" for moving backward. With the keys "Home" and "End" selects the first block, resp. the last block.

In case of using a second area or waveform (XY, Marker, FFT, Zoom) which has the focus, only the time markers will move in the main waveform display.

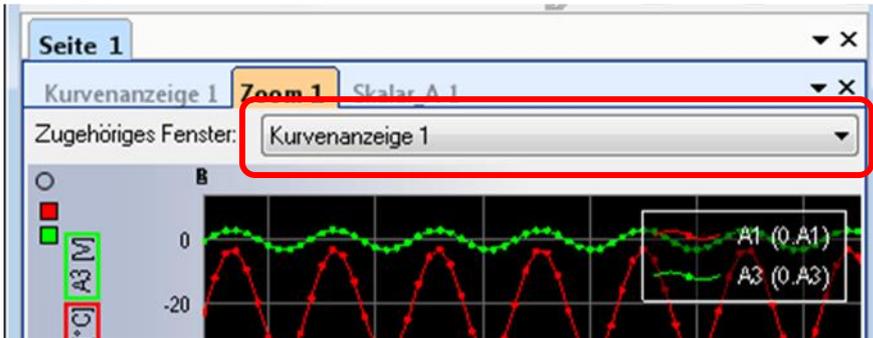


18.16 Additional Waveform displays

In addition to the standard YT - waveform display, there are additional types of display windows. They are available in through the Ribbon Tab "**Layout**":

- New **Zoom waveform display**: Enlarges an area of a Waveform.
- New **XY waveform display**: XY visualization of two or more traces.
- New **Marker waveform display**: Shows the digital marker signals.
- New **FFT waveform display**: Frequency spectrum visualization of recorded signals.

If you add any of these waveform display windows, signal curves can be visualized using different settings. The Related Display window may have to be determined (see picture).



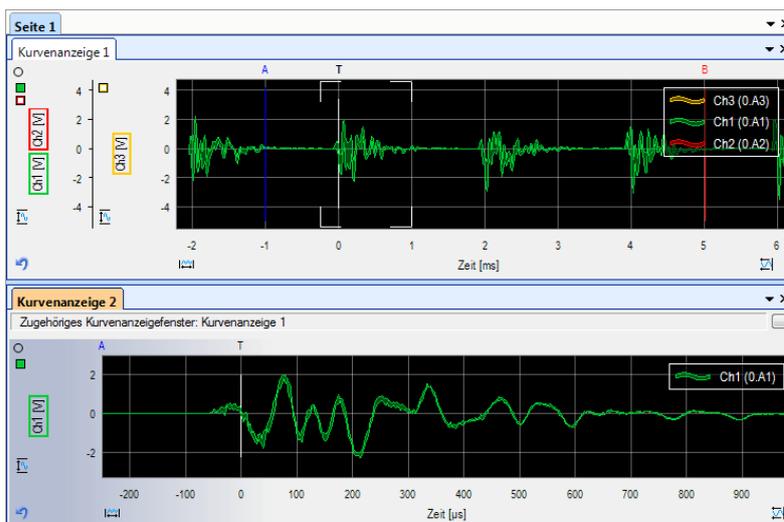
The Related Waveform display is mainly used to define limit marks for the time segment to which the signals in the special display windows relate.

You may also need to drag and drop the traces from the Control Panel or the Signal Source Browser. The traces can also be copied directly from the original window using drag & drop.

Four white corners (time window limits) now appear in the associated trend view, spanning the area that is relevant for the added trend window. These markings can also be moved with the mouse.

For a zoom curve display, this means that the area marked (in the associated window) is shown enlarged.

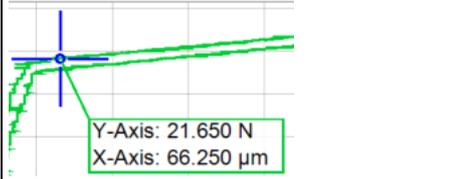
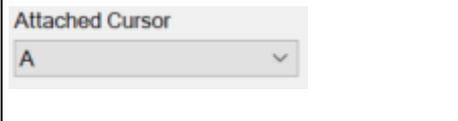
With an FFT curve display, the spectrum from the marked area is calculated and displayed. The same applies to the XY and marker display.



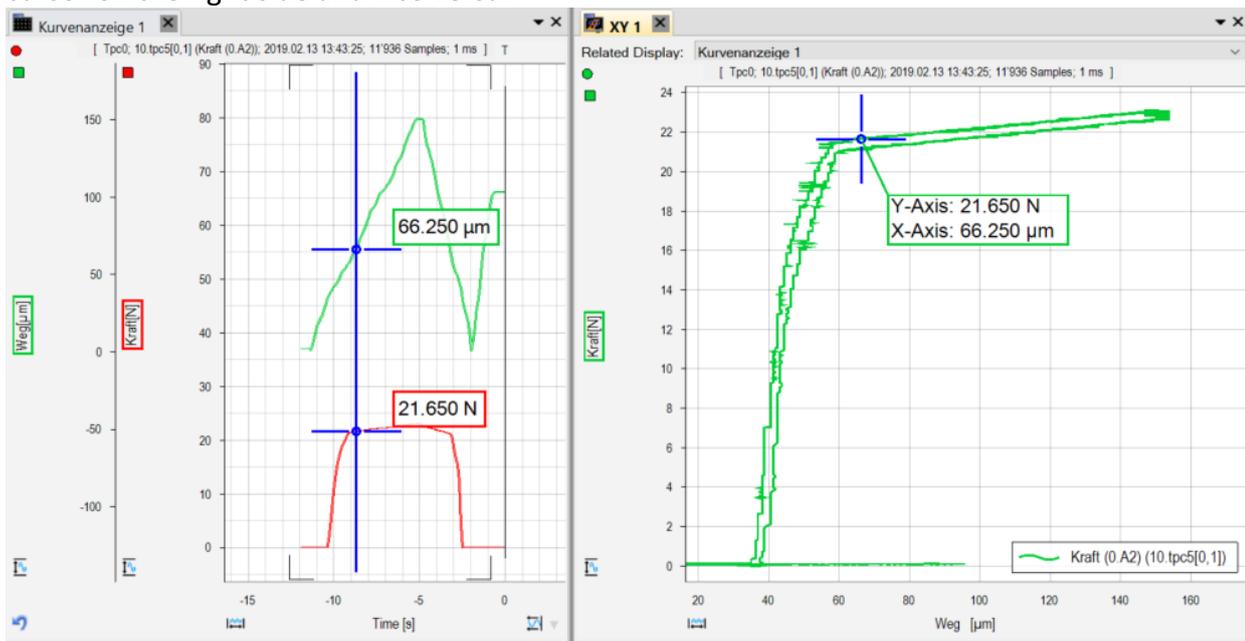
The additional Waveform display window can (like all windows and tables) be docked on any side of the other windows (see also chapter TranAX overview).

19 XY Waveform

The implemented XY Waveforms allows to synchronize data from the YT Waveform together with the XY Waveform itself.

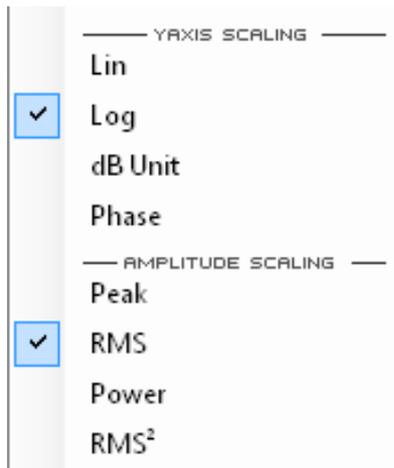
	<p>Right click on the XY Trace allows to place a XY Cursor. This Cursor can be moved around on the XY trace.</p>
	<p>This Cursor shows per default its corresponding data from the X and Y axis.</p>
	<p>Right click on the cursor, then left click "properties..." opens a dialog which allows to attach the XY cursor with a YT cursor in the "Related Display".</p>

With the cursor in YT- and XY Waveform attached, moving the cursor A left also moves the XY cursor on the right side and vice versa.



20 FFT Waveform

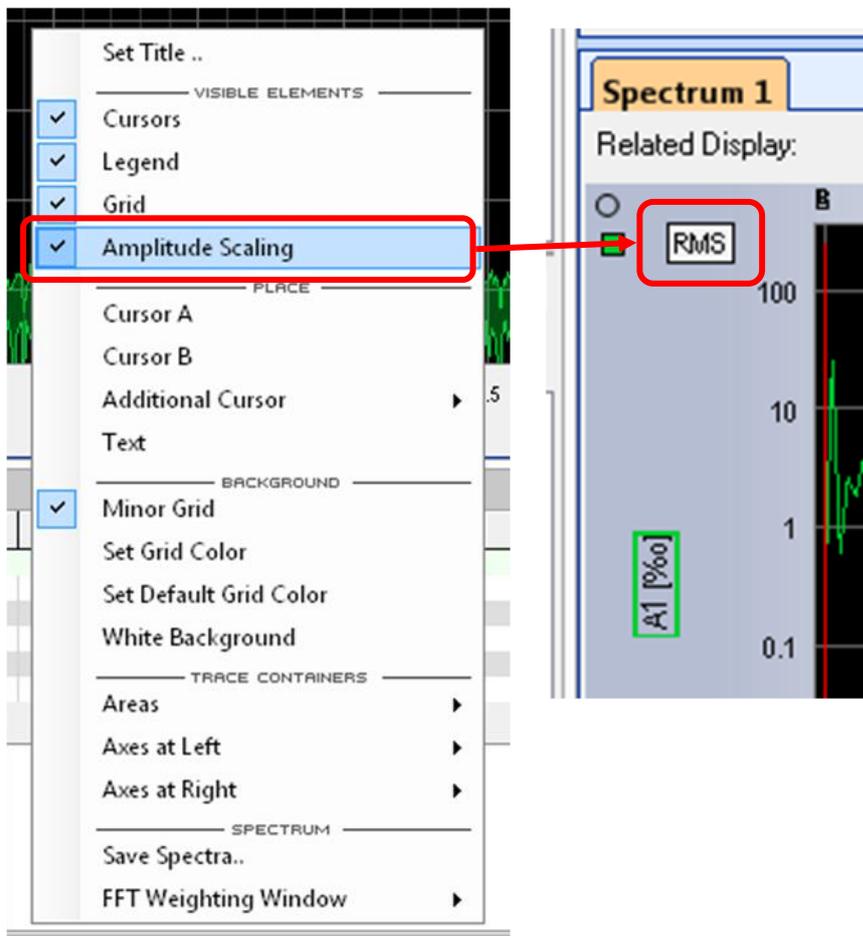
20.1 Vertical and Horizontal Scaling



The scaling for the Y-axis of an FFT spectrum display can be changed. Right click on the scale to open the following popup menu.

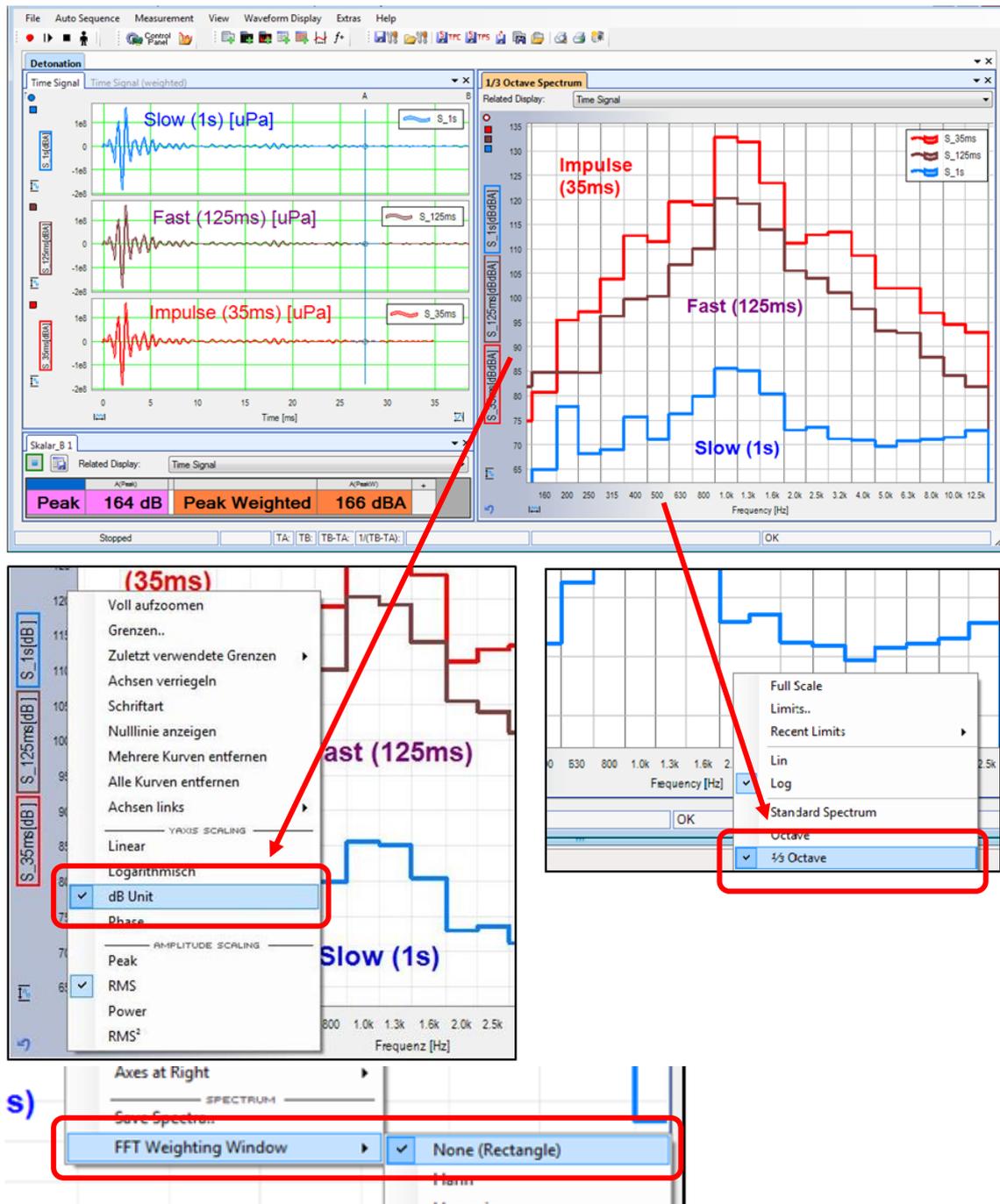
The scaling for Y-Axis will be set to **linear** (Lin), **logarithmic** (Log), **dB Unit** or **Phase**. The amplitude scaling can be set as **Peak**, **RMS**, **Power**, or **RMS²**.

To see the selected scaling in the upper part of the Y-axis, right click into the waveform display and check "Amplitude Scaling".



20.2 Octave and 1/3 Octave scaling

In addition to the "Standard Spectrum" display, the FFT waveform has now the option to display spectra traces as "Octave" or "1/3 Octave". These two are primarily used for the analysis of acoustic applications.



With a right click on the frequency scale, the display format "Octave" or "1/3 Octave" can be selected. For this format, **the Y-scale has to be set to "dB Unit"**.

Usually the **FFT Weighting Window** has to be set to "None" (Rectangular) for Octave and 1/3 Octave scaling.

21 Show Videos synchronized to recorded Traces

Video movies can be played in TranAX for-and backwards or simply displayed as a still image. Movies, recorded simultaneously with the captured traces, can be played back simultaneously, synchronized by the frame rate, for ease of analyzing traces and related video together.

The most common file formats are supported It is recommended to copy the recorded video files in a newly created subdirectory called "**Video**" in the directory of the currently used Experiment.

File	Directory
Trace Files (*.tpc5)	...\ <i>experimentName.exp</i> \ data default directory of the trace files
Video (*.avi, *.mpg, usw.)	...\ <i>experimentName.exp</i> \ videos This folder may have to be created manually

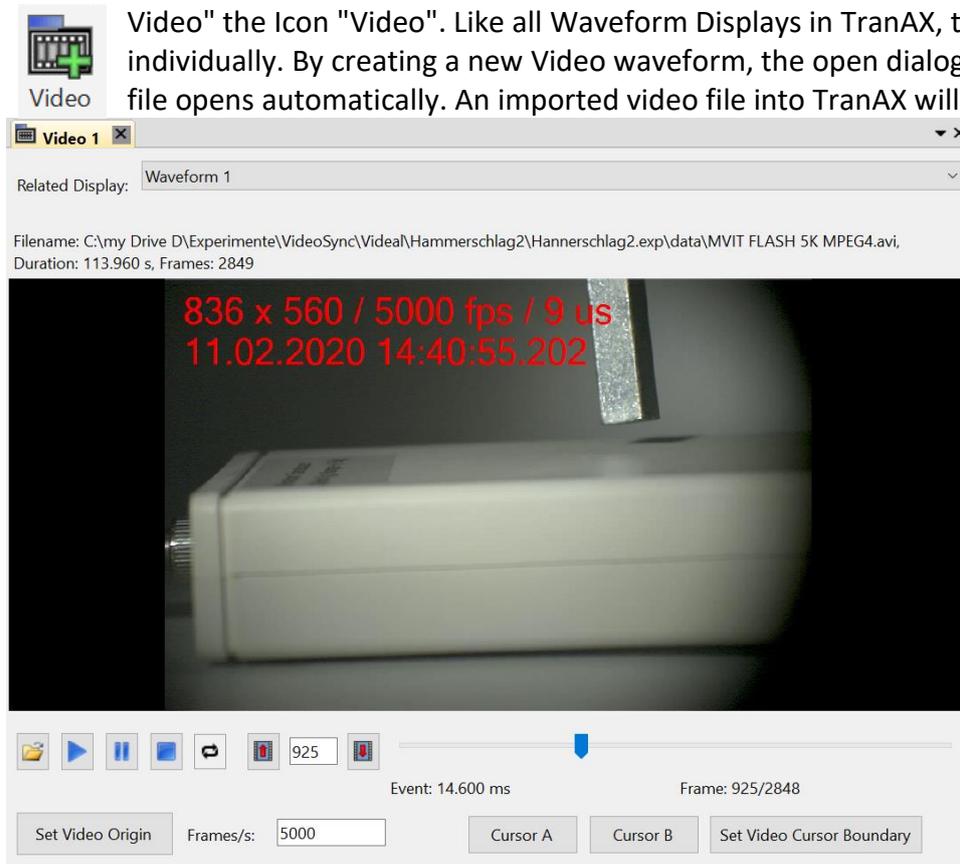


A video consists of a **video container** (AVI, MP4, MOV, OGG, etc.) plus a **video codec** (h264, mpeg4, xvid, divx, etc.)



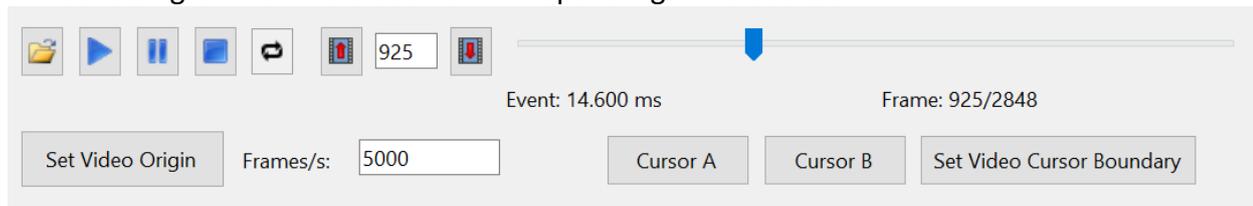
Since **TranAX version 4.1.2.2292**, the implemented video player engine, based on the VLC Media player, has been replaced by **OpenCV engine** (www.openvc.org). Reasons for this was a better support for the used file format und more flexible usage for single frame visualization.

To open a new Video Display, please click in the menu Ribbon Tab "Layout", "Documentation & Video" the Icon "Video". Like all Waveform Displays in TranAX, this one can be placed individually. By creating a new Video waveform, the open dialog for importing a video file opens automatically. An imported video file into TranAX will look like this:

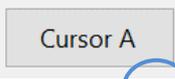
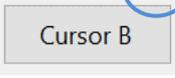
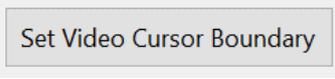


21.1 Video Waveform usage

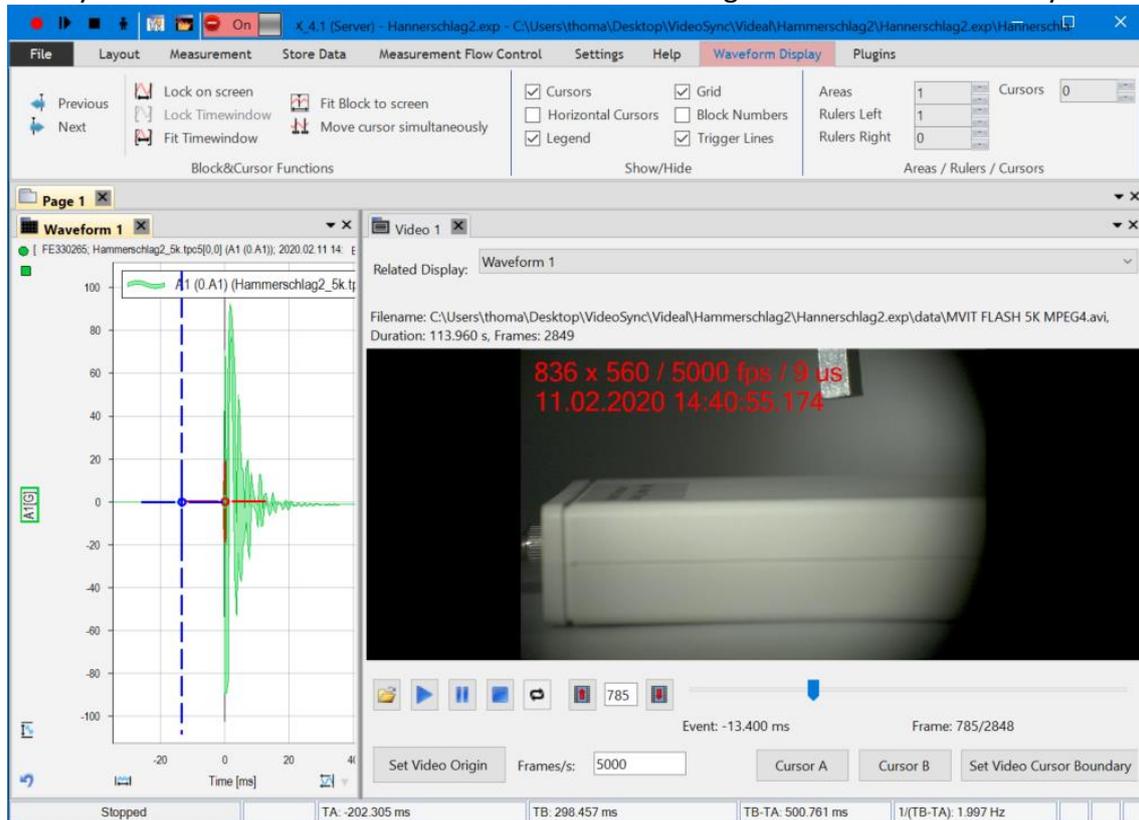
The following elements are available for operating the video waveform:



Related Display: <input type="text" value="Waveform 1"/>	The parameter "Related Display" will be automatically set to the name of the last active Waveform display with signal traces. This parameter can be adapted afterwards.
<input type="text" value="\\data\MVIT FLASH 5K MPEG4.avi"/>	"Filename" indicates the opened video file.
Duration: 113.960 s, Frames: 2849	"Duration" and "Frames" gives an overview regarding the duration of the video and the includes frames.
	Opens a file dialog to select another video file. Please note that the settings for Frame Rate have to be set again.
	Starts video playback, synchronized with the "Video Play Marker" in the "Related Display".
	Pauses the active video play back.
	Stops the active video play back and moves back to the first frame (frame 0).
	Repeats the video play back, either the whole video duration or within the set Cursor boundary.
	Move one frame backwards.
<input type="text" value="925"/>	The actual visualized frame.
	Move one frame forwards.
	Frame position slider, allows to move from first to last frame, if option "Set Video Cursor Boundary" is active, the slider position will be limited to its given position by the Cursors A and B.
Event: 14.600 ms	Indicates the timestamp of the position of the "Video Play Marker" compared to the "Video Origin Marker".
Frame: 925/2848	The current frame position, starts with 0.
<input type="button" value="Set Video Origin"/>	Sets the Video Origin in assertion with the "Video Origin Marker" of the YT Waveform.
Frames/s: <input type="text" value="5000"/>	The Framerate of the recorded Video. Has to be defined to have the Video in synchronization with measured traces.

	Set Cursor A from the "Related Display" to the current position of the "Video Play Marker".
	Set Cursor B from the "Related Display" to the current position of the "Video Play Marker".
	Plays the Video within the defined limitation between Cursor A and Cursor B, also locks their position.

The synchronization of the video with the associated signal can be done this way:



- **Place the captured traces** into a waveform display (e.g. drag and drop them from the Signal Source Browser into the waveform display). **Zoom into the trace to find a distinctive section in the signal**, where a single frame of the video can clearly be allocated.
- **Right click with the mouse** at the position of the distinctive section in the waveform display and click **"Video Origin Marker"** to set this marker.
- Set the Frame Rate field **"Frames/s"** according to the original **frame rate** of the video. This parameter cannot be read from the video camera in any format, electronically. Please check this parameter to be sure that the number is correct.
- Start the movie and **stop at the corresponding distinctive section**.
- Click the button **"Set Video Origin"** to synchronize the **actual frame** of the video with the **origin of the measured trace**. The "Video Play Marker" will be set to the origin of the signal trace.

By moving the "Video Play Marker" with the mouse, the corresponding frame of the video will be displayed. The waveform display can be zoomed in like in any waveform display to closer to the captured signal. After getting the exact frame for synchronization, the button "Set Video Origin" can be clicked again.

Now the video can be played forwards and backwards. In the Waveform display, the "Video Play Marker" also goes forwards and backwards, synchronized with the video. Conversely, by moving the "Video Play Marker" in the waveform display, the visible frame of the video will be synchronized too.



Please note the synchronization of the video and the traces is only given when the frame rate "**Frame/s**" is set correctly!

22 Documentation Window

The documentation window offers the possibility to easily and quickly create a page (A4, A3, etc.) with components (Waveforms, Scalar Tables, etc.), which can then be printed directly or saved as a PDF.

So, called template elements can be used as header or footer on the pages to place text elements, images, etc. which will be printed on each single page. Text fields gives you the possibility to display scalar results from the formula editor.

These items are available for the Documentation Window:

Displays	
	YT
	FFT
	Zoom
	XY
	Marker

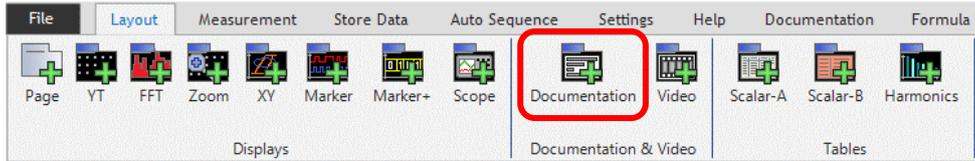
Tables	
	Scalar Table A
	Scalar Table B
	Harmonics Table

Shapes	
	Drawing lines
	Drawing von Rectangles

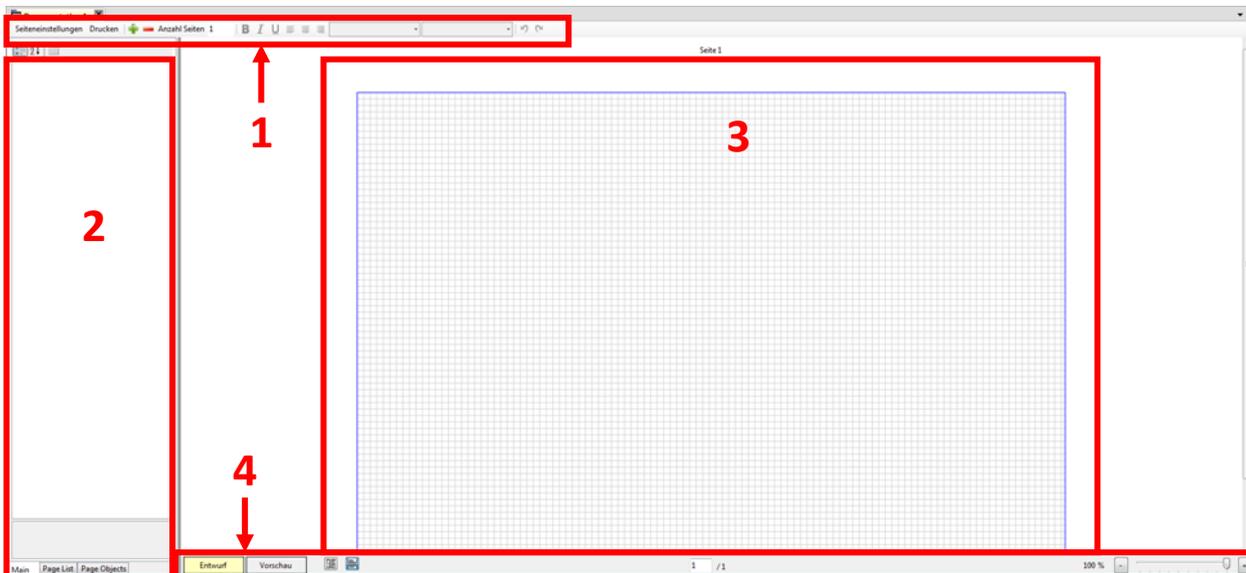
Misc.	
	Placement of Text
	Placement of Images

 Template	Templates (used for Header and Footer)
 RecLog	Recording Log
 Attributes	Attributes

To create a new Documentation window, click on Ribbon tab "Layout" Group "Documentation & Video" on the icon "Documentation":



This creates an empty Documentation Window:



1. Menu bar

Page layout, (A4, A3, portrait, landscape, etc.), printing, number of pages, font size for text elements can be set in the menu bar.

2. Settings window

Settings windows shows properties of a selected Item from the design area. Depending on the item, its properties can be different.

3. Design or preview window

The items (YT, FFT, Scalar table, etc.) can be placed on the design area or canvas. Select the item from the Ribbon tab "Documentation" then drag and drop it on the design area. Per default, this window is in "Design" mode, click on the button "Preview" to change to the "Preview" mode.

4. Status bar

In the status bar you will find the option to switch from Design mode to the Preview mode. There is also the option to change the page view either from left to right or top to bottom. In the middle of the status bar is the number of pages. There you can also jump directly to the required page. On the right side, you will find the zoom settings (0-100%). If the Zoom setting is selected below 100%, the elements like Scalar table A / B and harmonics table shows only drawn. With a double-clicking on the element, the control is displayed in a window. Thus, you can then configure the item as usual.

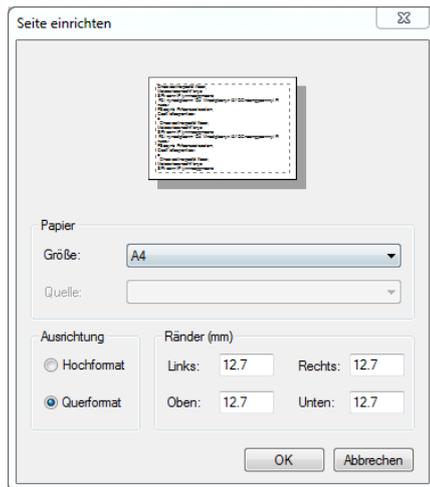
22.1 Place a design item

Design items (YT, FFT, Scalar tables A/B, etc.) can be placed onto the design area. Select the element from the Ribbon Tag, left-click and hold the mouse button on the design area and draw the outline of the element. Release the mouse button when it is on its final size and position,



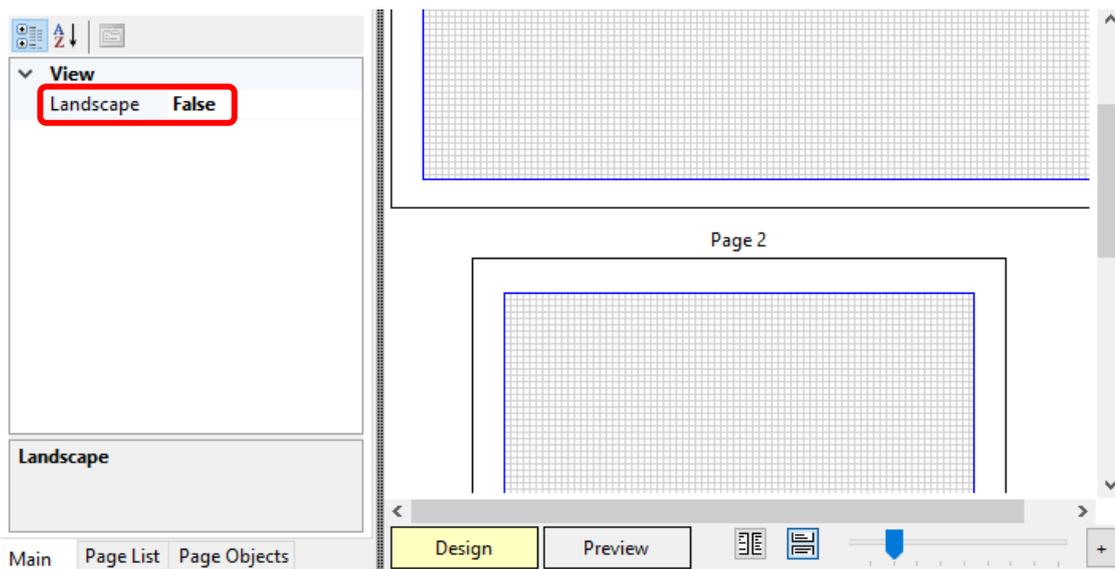
Make sure that "Design" mode is selected before you start placing the elements.

22.2 Page Settings



Click on the menu bar on the button "Page Settings" to open the settings dialog. This dialog will be opened by Windows itself, therefore the look depends on the installed version of Windows.

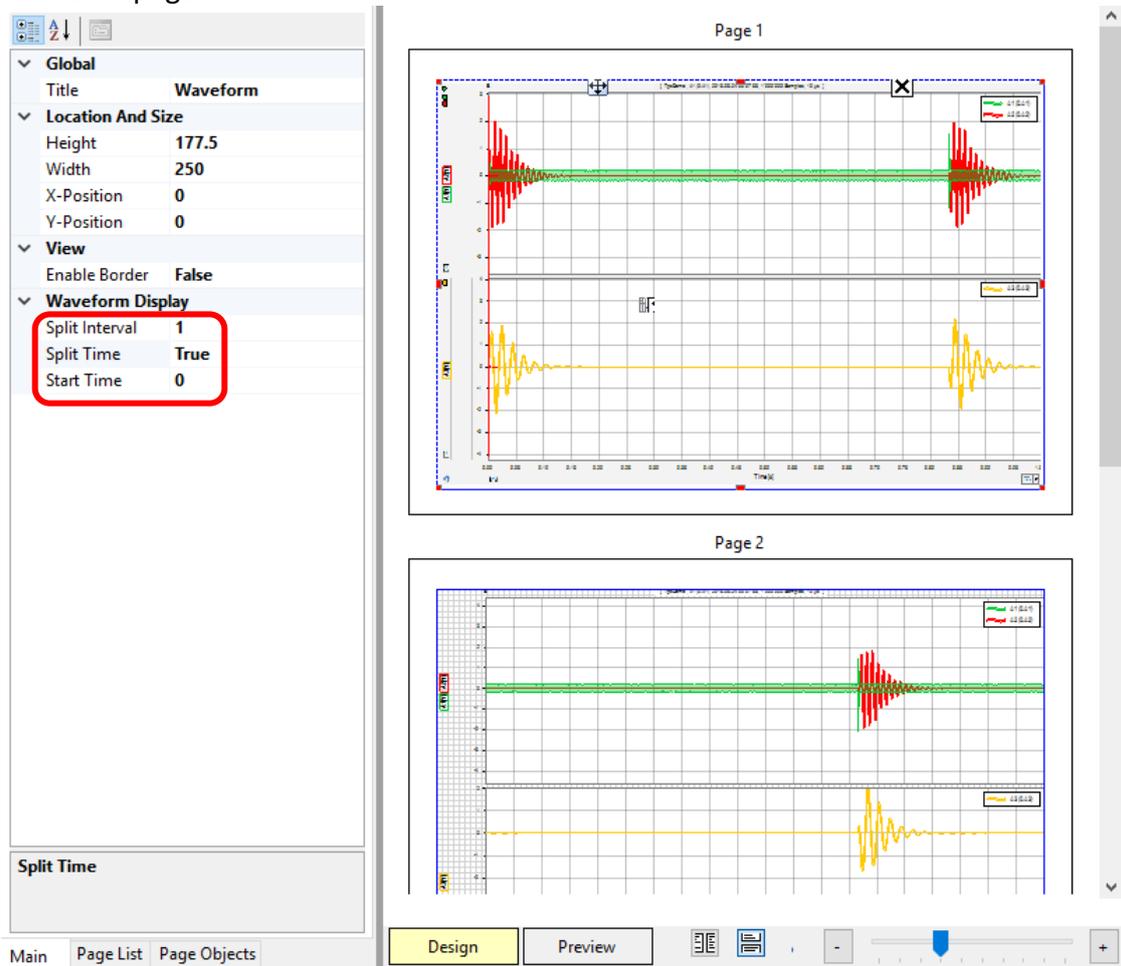
All settings in this dialog applies to all pages in the Documentation Window. However, there is the possibility to change the orientation of each single page in the design window. Click in the design window in an empty area and change in the properties window the orientation (landscape = false means portrait):



22.3 Split Waveform on multiple pages

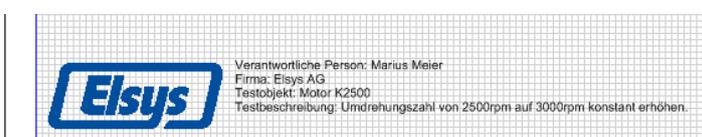
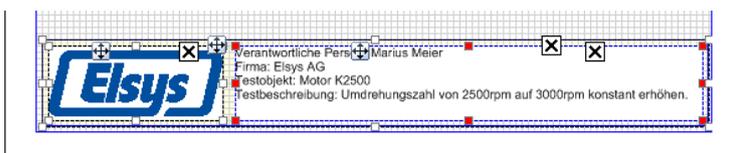
Waveform displays can be split up onto multiple pages. This could be useful for documentation of Continuous recordings, if you want to document complete recording and wants to keep the level of detail of the curves as high as possible.

To setup this splitting, clicking its curve window (for example a YT) and set the interval (in seconds), the start time (in seconds) and activated the time interval (True). After this the Waveform display automatically distributes to the additional pages. If necessary, add manually additional pages:



22.4 Templates (Header and Footer)

Templates are repeating objects on every page and are mainly used for Headers and Footers of the document. A template will be copied automatically on every new page. To create a template,



click in the Ribbon tab "Documentation" the icon Template. Draw this item on the page at its required position and dimension.

In this example, we added an Image and a Text element into this Template object. The template will be visible on every page at the same position.

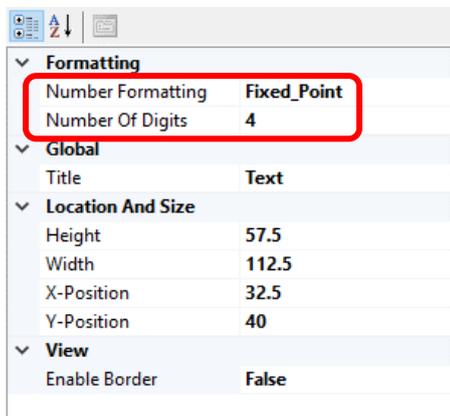
It is possible to set different templates for portrait and landscape orientation of the page. For this, select the template, and set in the object settings the parameter "Print only on same paper layout" to True.

22.5 Reserved keywords in Text objects

Text objects can handle some keyword, which will be replaced during printing or Preview mode. The syntax starts with angle bracket and percent sign (<%) followed by the keyword and ends with percent sign and angle brackets (%>). Supported keywords are:

Keyword	Description
<%pages%>	The number of pages in the Documentation
<%page%>	Number of the current page
<%date%>	Actual date

22.6 Scalar values in Text objects



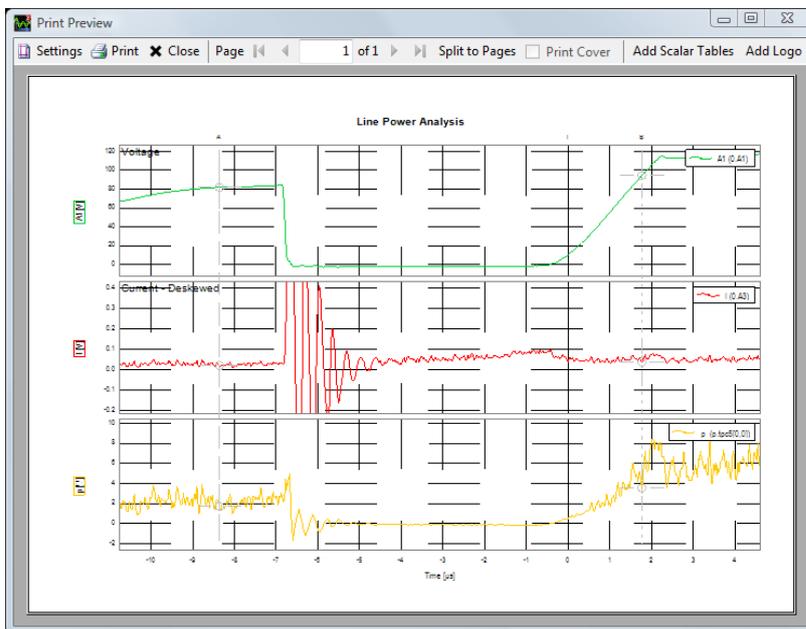
To use calculated scalar values, form the Formula Editor in a Text object, the name of the variable has to be written between angle brackets (<value>). The properties dialog gives additional possibilities for the formatting of the number.

22.7 Printing (Legacy Mode)



The print dialog, known from TranAX 3 should no longer be used for new applications and Experiments. This dialog is for compatibility purposes for existing applications remain available. It is recommended to work with the Documentation window.

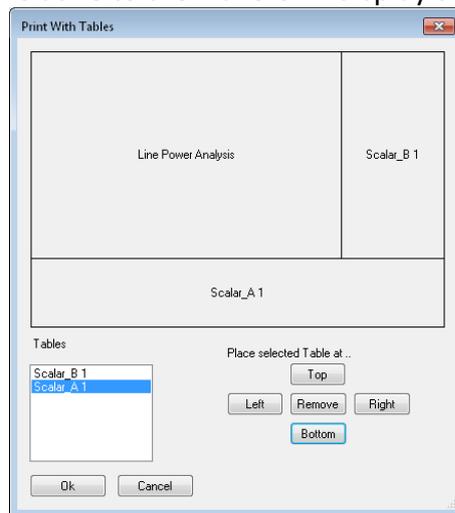
In order to print the recorded curves and traces, select the Waveform display and related scalar tables you would like to print. Then click the button "Print" in the Ribbon Tab "Store Data" in the group "Print & Snapshot". For a print preview click the button "Printer Preview" in the same Ribbon Tab.



In the Print preview a dialog window opens in with the active Waveform display. Depending on the used monitor, resolution, and other setting this view can be different from the view on the screen. Therefore, it is recommended to work with the documentation window.

Settings	Select the printer, paper size and orientation
Print	Click this button to send the image in the preview to the selected printer.
Split to Pages	<p>The time-axis can be expanded to reveal more details. Therefore, the waveform display in the preview is expanded over multiple traces depending on the setting in the window below. It is defined how much time to print per page and then calculated how many pages will be required to print it in the expanded time base.</p> <div data-bbox="510 1653 1032 1921" data-label="Image"> </div>
<input type="checkbox"/> Print Cover	In case there are Attributes or Recording Logs linked to the actual waveform display Print Cover can be selected. It will print the available logs in form of a list.

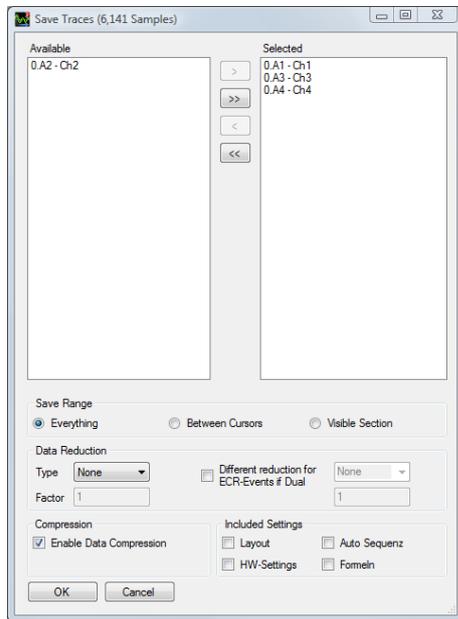
Add Scalar Tables In case there are Scalar Function tables related to the actual waveform display, the window Print with Tables allows to define at what position relative to the waveform display the table(s) should be printed.



Add Logo In case a logo or image should be included on the print-out, click Add Logo and browse for the graphic file to be loaded. Supported formats are bmp, jpg, gif and png.

23 Saving records and traces

To save your recordings, open the save window by clicking the save icon  or the menu **"File" / "Save Traces as tpc5"** (HDF5 format defined by NCSA, the National Centre for Supercomputing Applications, USA).



The list on the left side shows all **available traces** which can be added to the right list with the arrow buttons. You may select from all activated channels in Single Ended- or Differential mode of the actual hardware and all displayed signal traces in the waveform display.

Additional setting for data compression can be done in the menu **"Extras" / Settings" / "Import/Export"**.



If the stored data is to be used and displayed again in TranAX, save the recordings in HDF5 format (*.TPC5).

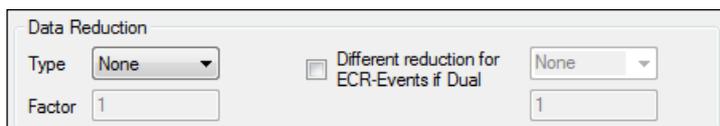
23.1 Save Range



stored to file.

It can be selected whether the entire trace, the section between the cursor A and B or the visible section should be

23.2 Data reduction



There are different methods to **reduce the data** before saving:

- **None:** No reduction.
- **Skip:** Only every n_{th} sample will be saved.
- **Average:** The moving **average of n samples** will be calculated and then saved.
- **MinMax:** For every n_{th} sample the smallest and largest value will be saved. With this method, two traces will be saved, one containing the lower and one the upper envelope value.

In case the data acquisition is **ECR mode and Dual** is activated a different type of data reduction may be applied to the data of the ECR events.

23.3 Compression

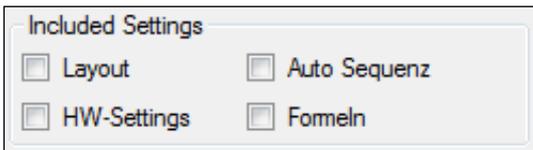


An additional data compression can be performed before the storage to file. The data compression may result in a smaller file size of about **10% to 15%**, depending on the trace shape.



The compression is lossless; there will no data be lost!
Depending on the amount of data it may take more time.

23.4 Included Settings



The TPC5 data file format allows storing not only multiple traces to one file but also the Layout- and HW-Settings and the contents of the Auto Sequence and the Formula Editor.

23.5 Export

Besides the capability to store traces in TPC5 file format, it is possible to export recordings in TPC- (TransAS 2 compatible), in ASCII-, DIAdem- or in Krenz-format. Open the export window via the Ribbon tab **"File" / "Export"**. And click **"Export Traces"**.



23.6 Saving pages and waveforms

This function allows saving a page, the **actual layout part** (waveforms, scalar tables, etc.), to a file. The Save dialog will automatically check the focused (selected) page. It allows checking and unchecking all the existing pages in a current Experiment before saving. The Save Page dialog gives the option to choose between **"Keep trace source"** and **"Include trace data"**.

"Keep trace sources" will store the layout settings of the page. The traces in the page windows will be saved as references (indicator to trace source) only. This means no trace data will be stored into this file.

"Include trace data", will also save the layout settings of the page. Additionally, the trace data will be stored into the file and the references of the original traces will be replaced to the just saved traces in the file. This file can be opened again for **further analysis** of the traces providing the same appearance as during the measurement, when the traces were directly acquired through the hardware channels. Without this option, the trace data will not be stored into the file.



To save a page, click in the Ribbon tab **"Store Data"** in the group **"Save and load Data"** the Icon **"Save Page"**. Then enter a filename and choose from the following options:



"Keep trace sources": Just the **references** will be saved to the file.

"Include trace data": The **trace data** will be saved to the file. The original trace references will be replaced to the saved traces in the file.

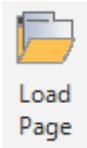
Tab Page Selection

Page: Page 1
 Page: Page 2
 Page: Page 3

"Page Selection": Check the pages for saving to the file. The focused page will be checked automatically. Uncheck the pages which are not needed.



The file with the extension ***.tdp** will be stored to the "Data" directory of the current Experiment.



To open a page, click in the Ribbon tab **"Store Data"** in the group **"Save and load Data"** the Icon **"Load Page"**. Then select the file in this dialog and make sure that in the group box "Page selection" only the pages are checked which should be opened.

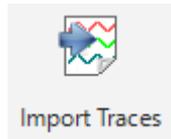


A Page file (*.tdp) can also be opened by the Signal Source Browser to get access to the saved traces.

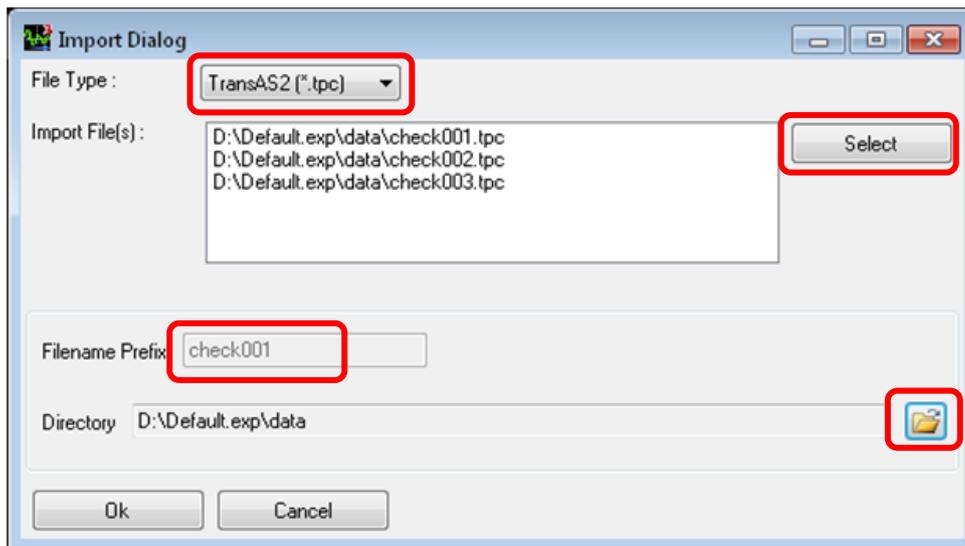
23.7 Import Option

Several file formats can be imported into TranAX. The imported traces will be converted (copied) into TPC5 files, so they can be analyzed in TranAX. The following file types can be imported:

- TPC files, default format from TranAX version 2
- ASCII files, ASD files also from TranAX version 2
- general ASCII-Files, imported with the import wizard
- Audio files



Click in the Ribbon tab **"Store Data"** in the group **"Import"** the Icon **"Import Traces"** to open the import dialog. Please note that an import option has to be installed before this menu item is visible.



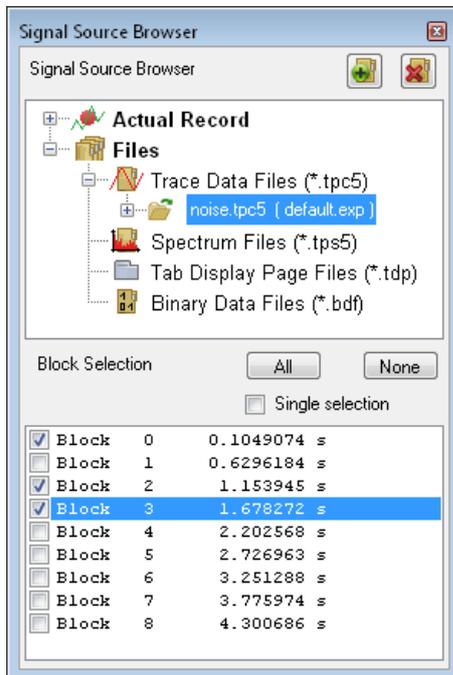
With the parameter **"File Type"** can be selected which kind of file should be imported (TPC, ASCII, Audio, etc.). Then click the button **"Select"** to choose the original files for importing. If just one file is selected, the filename for the destination file can be entered by the field **"Filename Prefix"**. For multiple file selection, the name of the original files will be taken over. The ending ".tpc5" will be added automatically.

Click the button with the folder icon  do change the **"Directory"** for the destination file. Normally, this will be the data folder of the actual Experiment. An existing tpc5 file will be overwritten without any warning.

The converted traces will be added automatically to the Signal Source Browser  in the menu bar. The converted traces can be dragged and dropped into the Waveform Display.

24 Signal Source Browser

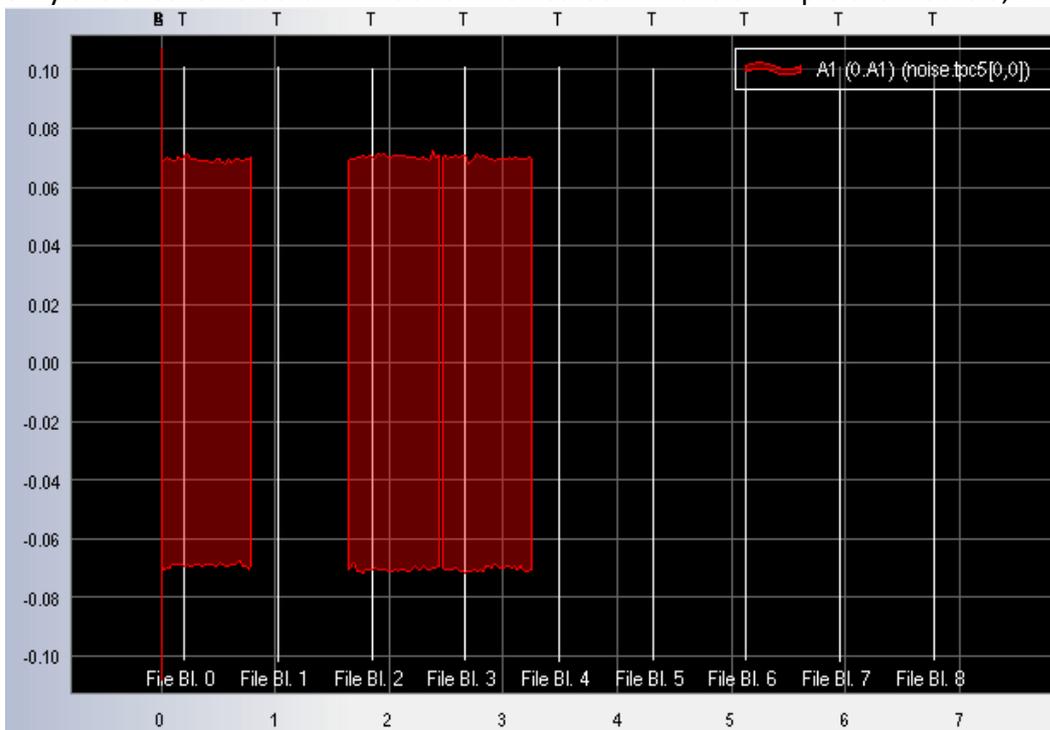
The Signal Source Browser can be opened by clicking the icon  in the menu bar. Per default it will be docked on the left side of the TranAX Display.



Additionally, to the control panel list, the **signal source browser shows all system channels**. Use drag & drop to place a measurement in a [Waveform Display](#). Signals saved in files are made accessible in the browser tree. To add waveform-files to the Signal Source browser, use the *Signal Source Browser*-icon  in the top right corner of the window.

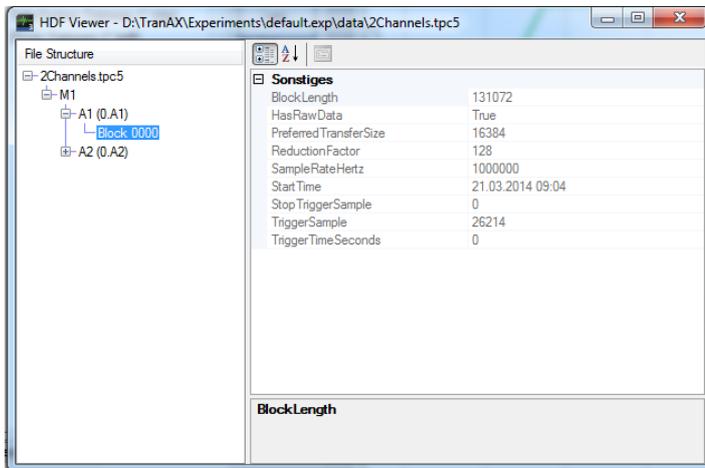
For recording in Multi-Block or ECR mode, the bottom part of the Signal Source Browser window shows each recorded block corresponding to the selected channel. With the button "None" the curves of all blocks can **be hidden**, the button "All" displays **all recordings** again. By checking a box, only the trace of this block will be displayed while all the others are hidden. Holding the < Ctrl > button you can either add blocks to your selection or remove them.

Only the checked blocks will be shown on screen. In this example the blocks 0, 2 and 3 are visible.



24.1 HDF Viewer

With the HDF Viewer an HDF file (*.TPC5, *.TPS5, *.TDP) can be observed. All the related meta-information that is being stored in the file, is obtained that way. By right-clicking an HDF-file “Open HDF-Viewer” can be selected in the menu.



24.2 Excel Importer

When the **Excel-Importer** was installed at set-up time, then in the menu of a HDF5 file, the entry “Export to Excel...” will appear. With that it is possible to export directly from the Signal Source Browser one or more curves into an Excel worksheet.

25 Scalar Functions

TranAX gives direct access to **more than 60 built in Scalar Functions**. A Scalar Function is a measurement or **calculation** of a **waveform parameter** such as Maximum, Peak-Peak, RMS or the Phase between two traces.

A complete list of Scalar Functions can be found in the [Appendix](#).



Marker symbols can be attached to the traces, referencing particular scalar calculations. They are small circles for amplitude values (e.g. Maximum, Peak-Peak, etc.) or small squares for time values (e.g. RMS, Period, etc.).

These require global adjustments under "Extras / Settings / User Interface / Scalar Function Tables / Show Readout Markers on Trace"

The Scalar Functions are set up and configured in two different types of tables which allow the user to measure any input trace from the current acquisition or a trace from file versus one or multiple of the available Scalar Functions.

[Scalar Functions Table A](#) shown in the next figure is a matrix type of table of traces versus Scalar Functions. It is very **quickly set up** and ideal for Scalar Functions such as *Frequency* that are calculated of a single trace between the same pair of cursors at common baseline and hysteresis.

Trace	A	B	B-A	RMS	PP	Freq
A1 (0.A1...)	0.005 V	-0.294 V	-0.299 V	0.950 V	5.253 V	8.584 kHz
A2 (0.A2...)	-0.002 V	-0.054 V	-0.052 V	0.697 V	3.311 V	19.990 k...
A3 (0.A3...)	0.000 V	0.112 V	0.112 V	0.556 V	2.763 V	9.001 kHz
A4 (0.A4...)	0.000 V	0.035 V	0.035 V	0.575 V	2.485 V	4.287 kHz

[The Scalar Functions Table B](#) on the other hand is **very flexible** and designed for **complex Scalar Functions**. It gives the possibility to define per function the primary trace, the reference trace if required, the pair of cursors, and baseline with hysteresis if required.

RMS(A1 (0.A1)) [A,B]	PP(A1 (0.A1)) [A,B]	Freq(A1 (0.A1)) [A,B]	+
0.950 V	5.253 V	8.584 kHz	

The sections set up [Scalar Functions Table A](#) and set up [Scalar Functions Table B](#) describe step by step how a new Scalar Function Table is opened and configured with the Scalar Functions of interest.

A description of all available Scalar Functions is directly displayed within the window Scalar Function of the TranAX software and in the [Scalar Functions Description Table](#) of this manual.



A new Scalar Function table will be positioned as defined under the menu "**Extras**" / "**Settings**" / "**User Interface**" / "**Default Tab Window Placement**".

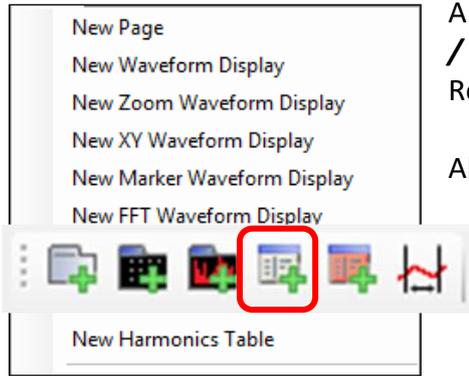


Selected cells of a Scalar Function table can be copied (copy & paste, <Ctrl>+c / <Ctrl>+v) into another application (e.g. Excel).



By pressing <Shift> + <Ctrl>+c instead of <Ctrl>+c, **column header** of a **Scalar Function table A** will be copied.

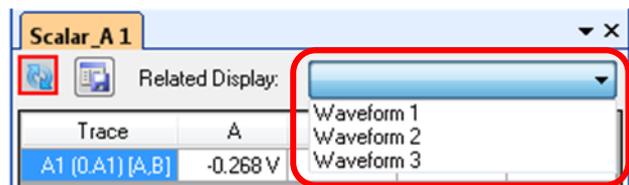
25.1 Scalar Functions Table A



A new Scalar Functions Table A can be opened in menu **"View" / "New Scalar Function Table A"**. This will create a new Register inside the actual page.

Alternatively, you can also click the icon in the Toolbar.

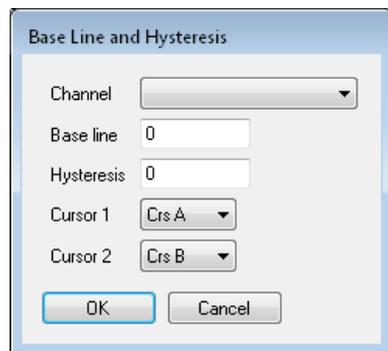
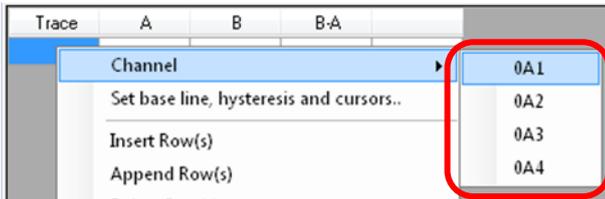
The columns **readout of cursor A, B** and the vertical **difference** between the two cursors are inserted by default. Click on the Dropdown list in the top right corner in order to select from which graphical display window the scalar functions will be calculated.



Function Table can be positioned either on any side of the current waveform display (Drag & Drop) or it can be kept as a tab, the default position.

25.1.1 Select a Trace for the Scalar Functions

To apply a Scalar Function to a trace, **right click** on the **"channel" column** and select an available channel in the context menu. Scalar Functions may only be applied to channels that have been added to the related waveform display.

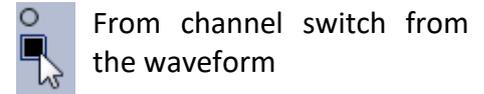


Alternatively, a double-click into a field of the Trace-column opens the **Base Line and Hysteresis** dialog where the trace, base line, hysteresis, and the two cursors for the measure gate can be selected.

As soon as the channel is selected, the cursor readout values are filled into the table.

i In a Scalar Table, only traces can be selected which are in the **related display available**.

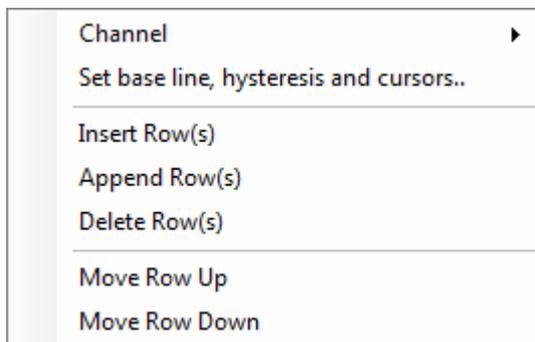
i Traces can be added via Drag & Drop to the column "Trace" in two different ways:



i The Scalar Function Tables are updated in **Single Shot** mode only or during the delay in a run of a loop in an Auto sequence.

In case enabled [Auto-Refresh](#) , the table will be updated also if Single Trigger Mode is disabled

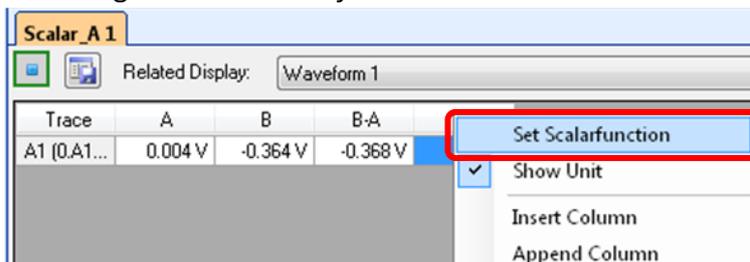
i With "Set base line, hysteresis and cursors.." additional parameters can be set for the calculation of level-based scalars.



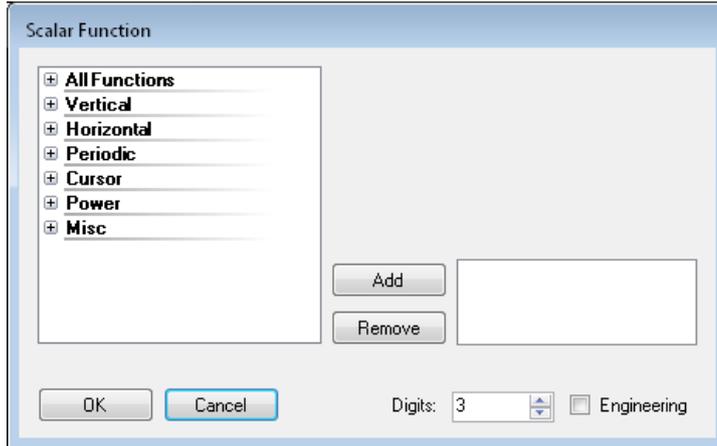
New rows can be inserted, appended or deleted via the **Trace-context menu** that is opened with a right-click on the line to modify in the Trace-column. In order to modify the order of the rows, they can be moved up or down.

25.1.2 Select Scalar Functions

To select the Scalar Function to be calculated and displayed, right-click on a free column header and navigate to *Set Scalarfunction..*



It is also possible to just double-click on the column-header of a column to open the Scalar Function dialog.



The **Scalar Functions** are listed in several **categories**:

- **All Functions:** Shows all available Scalar Functions.
- **Vertical:** Lists the Scalar Functions used to measure vertical values. Some functions such as Overshoot+ are typical parameter measurements on pulses.
- **Horizontal:** Lists the Scalar Functions used to measure horizontal values. These Scalars are typically timing parameters of a trace and often require a baseline level to be set.
- **Periodic:** Lists the periodic Scalar Functions. These Scalars require to be applied on cyclic signals.
- **Cursor:** Lists the Scalar Functions related to the cursor readout and cursor position.
- **Power:** Lists the Scalar Functions typically used in electrical power analysis applications.
- **Misc:** Lists some special Scalar Functions such as Area or Number of Triggers that don't belong to a category above.

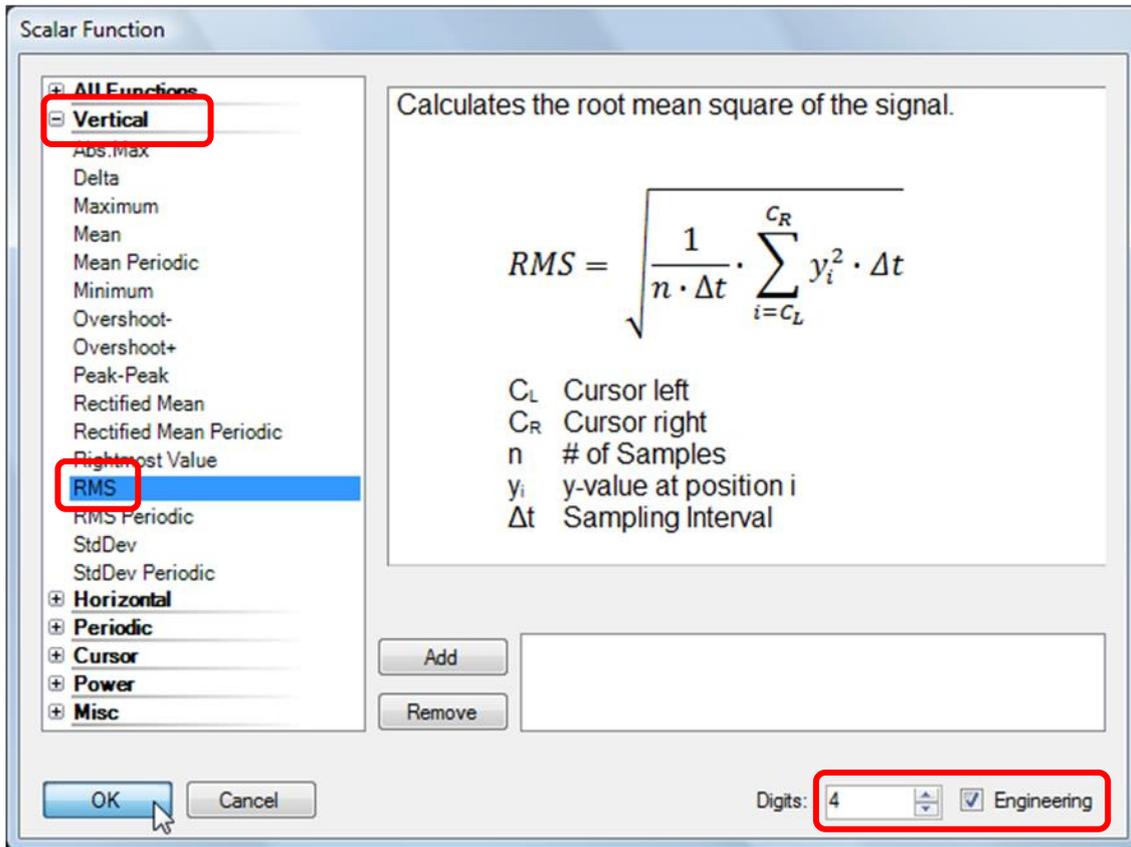


Some Scalar Functions are measured at a baseline level with a hysteresis. The [Scalar Functions Description Table](#) describes all Scalar Functions and gives information about the requirement to set the baseline level and the hysteresis.

25.1.3 Example 1: RMS

- RMS - Vertical Scalar Function; w/o Reference Channel; w/o Baseline and Hysteresis

Double-click on the **column header** of an existing Scalar Function to replace it. Open the list of the **Vertical Scalar Functions**. Select "**RMS**". The equation is shown in the right hand side of the list. By default, results are displayed in non-Engineering mode with 3 Digits. Change it to **Engineering mode with 4 Digits**. Click "OK" to accept the modifications to the Scalar Function table. At the time of closing the Scalar Function window, *RMS* will replace the previous scalar.



The RMS value (between cursors A and B) of the selected trace is now calculated and shown in the column "RMS".

To the right of the scalar *RMS* a new empty column was automatically inserted.

Scalar Functions 2				
Related Waveform display: Waveform 2				
Trace	A	B	A-B	RMS
Pulse (Pulse.tpc5[0.0])[A,B]	1.01 V	-0.38 V	1.40 V	710.4e-03 V



Normally, the Scalar Function will only be applied between two cursors (see [Scalar Function Description Table](#)). Move to the associated Waveform tab and place the active cursor in order to cover the desired waveform region.

Some functions allow defining additional cursor pairs to set the time window individually.

25.1.4 Example 2 Frequency

- *Frequency* - Horizontal Scalar Function; w/o Reference Channel; w. Baseline and Hysteresis

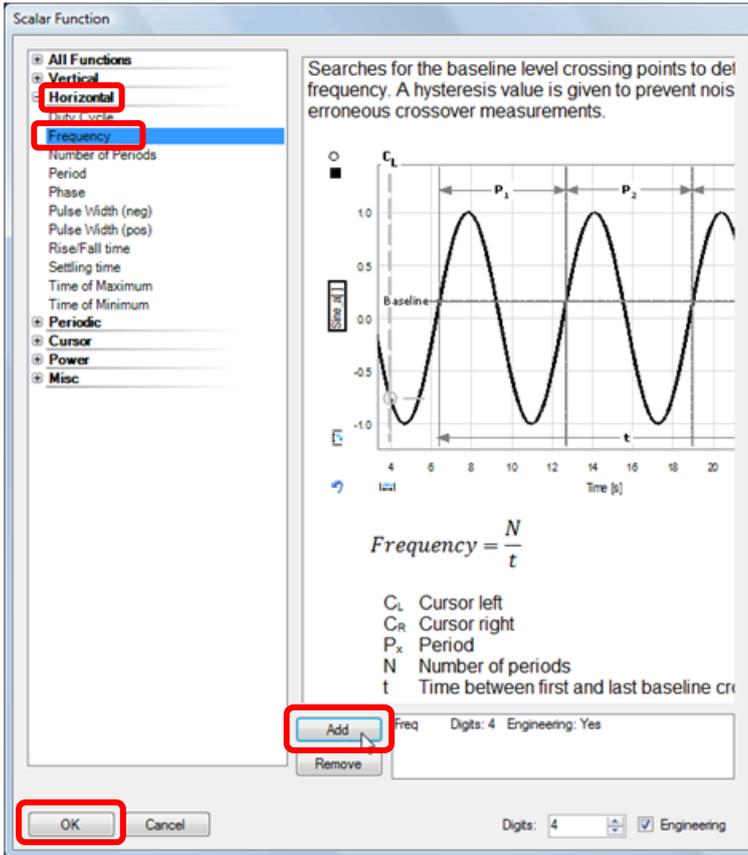
Scalar_A 1				
Related Display: Waveform 1				
Trace	A	B	B-A	RMS
A1 (0.A1...	-0.005 V	-0.523 V	-0.519 V	1.079 V

Double click on the empty column header to add a **new Scalar function**. The Scalar function dialog will open.

Open the list of "Horizontal" Scalar Functions and select "Frequency". The equation and a graph are shown on the right side of the list. Engineering display mode is selected with 4 Digits, because

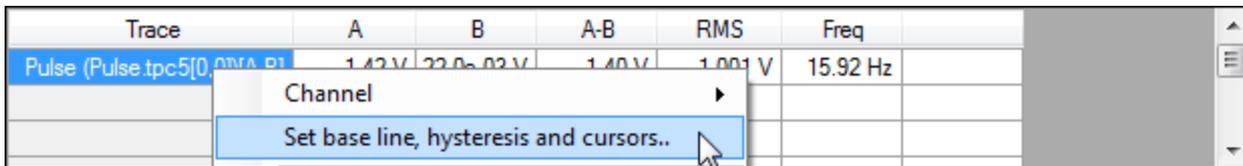
the Scalar Function window remembers the last selection.

Click "Add" to insert the selected function "Frequency" to the list. Click "OK" to close the window and to add (in this case append, because the procedure was started this way) the function *Frequency* to the table. If there were multiple scalars in the list, they would be added from top down to the table in the order left to right.

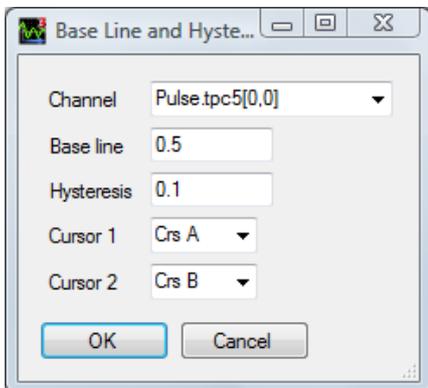


Right-click on the Trace of the active line, select "**Set base line, hysteresis and cursors..**".

Alternatively, double-click on the Trace of the active line.

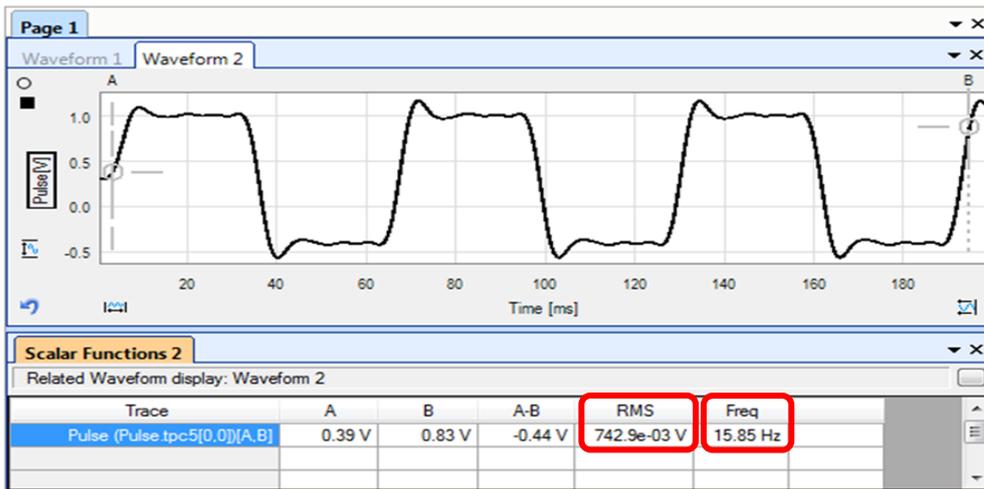


In the "Base Line and Hysteresis" dialog the Base line level, the hysteresis and the pair of cursors can be specified. The Cursor 1 and Cursor 2 are the measure gate and can be chosen from any activated cursor in the related waveform display.



The frequency is now measured at the baseline level 0.5 V at a hysteresis of 0.1 V between the cursor A and B.

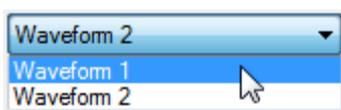
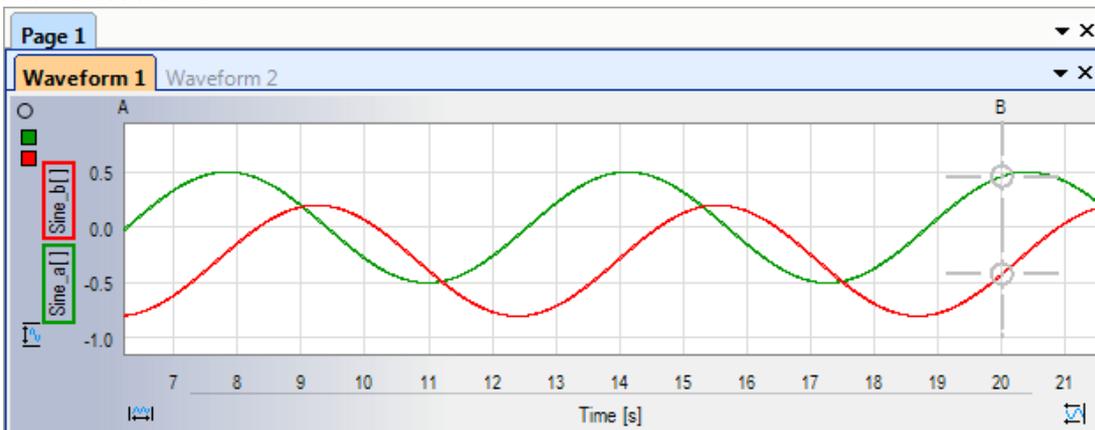
The table Scalar Functions 2 has now calculated RMS and Frequency.



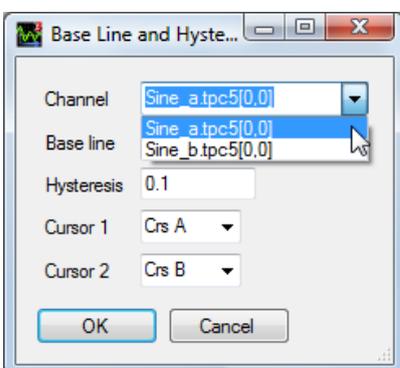
25.1.5 Example 3: Phase

- *Phase* - Horizontal Scalar Function; with Reference Channel; with Baseline and Hysteresis

There are two waveforms in the Waveform 1 tab that we'd like to use for measuring the Scalar Function *Phase*.



To get access to the two traces in Scalar Functions 1, the Scalar Function Table needs to be **related** with this **Waveform Display**. As described in the beginning of this chapter, click on the button on the upper right-hand side and select Waveform 1.



Once the waveforms in 'Waveform 1' are accessible within the Scalar Functions 1 tab, they can be selected as the active channels. With a **double-click** into the first line in the Trace column, the input source for this line can be selected.

Scalar Function

Reference Trace: Sine_b.tpc5[0,0]

Returns the phase in degrees between the sign and the signal used as a reference of two periodic signals frequency (usually voltage and current). The reference sign selected from the corresponding drop down list is analyzed amplitude.

A min. of 3.5 signal periods are required between the calculate the Phase. A negative result will be returned of $\phi \geq \pm 180^\circ$, if the reference signal leads the other on

$Phase = 360 \cdot \frac{t}{P}$

CL Cursor left
 CR Cursor right
 P Period
 t Time difference at baseline level to 50%

Buttons: Add, Remove, OK, Cancel

Digits: 4 Engineering

In this example we'll replace the Scalar Function *Frequency* with *Phase*. **Double-click** on the column header "Freq" to open the Scalar Function dialog. Select "**Phase**" in the category Horizontal and select the Reference Trace. Select the desired trace in the drop down menu.

Click "**OK**" to accept the selection and close the Scalar Function dialog. The Scalar Function Phase reports "No periods". This is probably the case, because the Base line level is not set appropriately. It is required to have a closer look at the levels of the trace Sine_a.

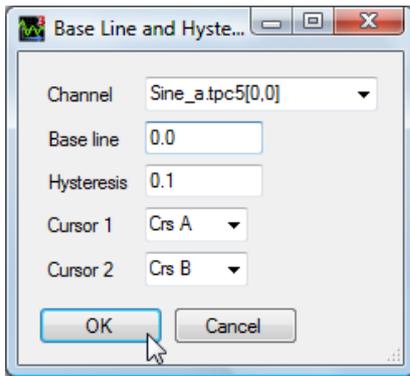
Page 1

Waveform 1 Waveform 2

Scalar Functions 2

Related Waveform display: Waveform 1

Trace	A	B	A-B	RMS	Phase(Sine_b.tpc5[0,0])
Sine_a (Sine_a.tpc5[0,0])[A,B]	-0.24	0.50	-0.74	355.9e-03	No periods.



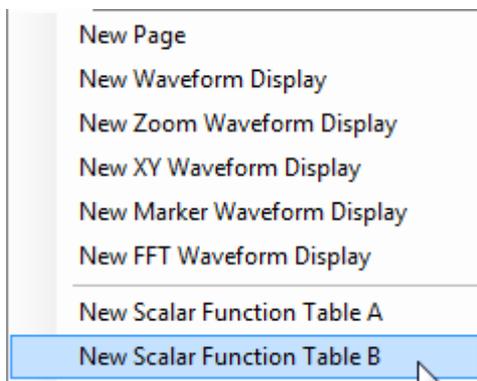
It seems appropriate to set the Base line level of the trace Sine_a to 0V. The Hysteresis value (should never be set to 0V) and the cursor we'll keep the same. Click "OK" to accept the values and close the window.

The calculation of the Phase can now be performed because the Base line level crossings are found.

Trace	A	B	A-B	RMS	Phase(Sine_b.tpc5[0,0])
Sine_a (Sine_a.tpc5[0,0])[A,B]	-0.24	0.50	-0.74	355.9e-03	-81.24 °

Phase(Sine_b.tpc5[0,0])	-81.24 °
-------------------------	----------

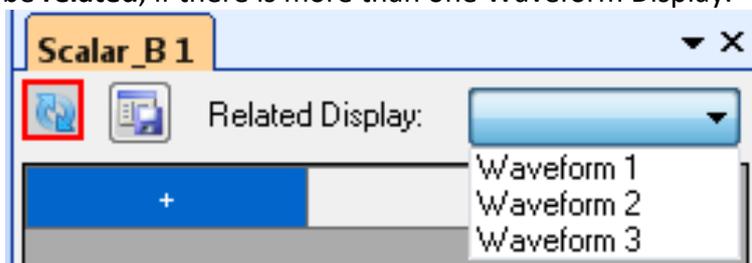
25.2 Scalar Functions Table B



Select "**View**" / "**New Scalar Function Table B**" from the menu bar or click the Icon  in the Toolbar to open a new Scalar Function Table B.

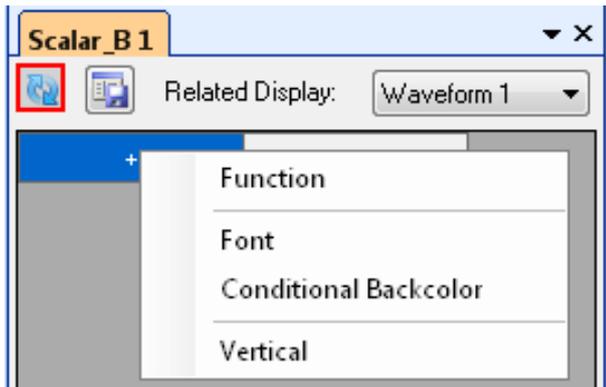


In the upper right dropdown list, you can choose to which **waveform** tab the scalar function will be **related**, if there is more than one Waveform Display.



The new scalar function table B will be positioned relative to the related waveform display as defined under menu "**Extras**" / "**Settings**" / "**User Interface**" / "**Default Tab Window Placement**".

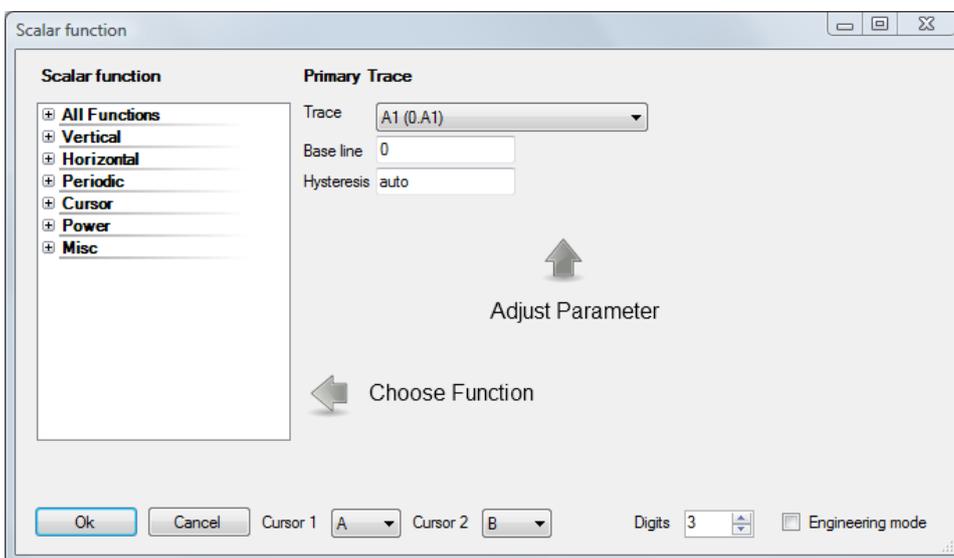
25.2.1 Add a Scalar Function to the table



To **apply a Scalar Function** to a trace, right-click on + and select Function in the context menu. Alternatively, a **double click** into the + -field opens directly the window Scalar Function shown above.



Scalar Functions may only be applied to channels that have been added to the related waveform display.



In this window all the **configurations** for the Scalar Function are made. Under Scalar function the scalar that should be calculated is selected. The Scalar Functions are listed in several categories (like Scalar Function Table A):

- **All Functions:** Shows all available Scalar Functions.
- **Vertical:** Lists the Scalar Functions used to measure vertical values. Some functions such as *Overshoot+* are typical parameter measurements on pulses.
- **Horizontal:** Lists the Scalar Functions used to measure horizontal values. These Scalars are typically timing parameters of a trace and often require a baseline level to be set.
- **Periodic:** Lists the periodic Scalar Functions. These Scalars require to be applied on cyclic signals.
- **Cursor:** Lists the Scalar Functions related to the cursor readout and cursor position.
- **Power:** Lists the Scalar Functions typically used in electrical power analysis applications.
- **Misc:** Lists some special Scalar Functions such as *Area* or *Number of Triggers* that don't belong to a category above.

Start with selecting a Scalar Function. Then trace as the source of the calculation is chosen under **Primary Trace**. Some Scalar Functions require a Reference Trace and/or a Base line with Hysteresis to configure as shown in the Scalar Function Examples.

Finally, the pair of cursor needs to be defined, between which the scalar will be calculated and the display format. Then all configuration settings will be accepted and the window closed by clicking "Ok".

The table looks now similar to the table shown below.

Function	
AbsMax(I)	+
0.87 A	

Function
Delete
Move Right
Copy
Insert Empty Col/Row
<input checked="" type="checkbox"/> Show Cursors
<input checked="" type="checkbox"/> Show BNC Connector
<input checked="" type="checkbox"/> Show FileName
<input checked="" type="checkbox"/> Show Unit
Font
Conditional Backcolor
Vertical

A right-click on the Scalar Function field in the table opens a dialog window. This allows to select a different function, delete, copy or to insert an empty column or row. It is also possible in the trace field to show the selected cursor, the channel input source and the unit. The selection Vertical will modify the orientation of the table from horizontal to vertical.

Font
Back Color
Font Color
Vertical

A right-click on the table element of the measurement value brings up a dialog box with parameters to modify the way the result is displayed.

Function
Vertical

Finally, a right-click on the + opens a dialog box with the items to select a Scalar Function for this column/line or to change the orientation of the table from horizontal to vertical.



The Scalar Function Table is updated in Single Shot mode only or during the run of a loop in an Auto sequence.

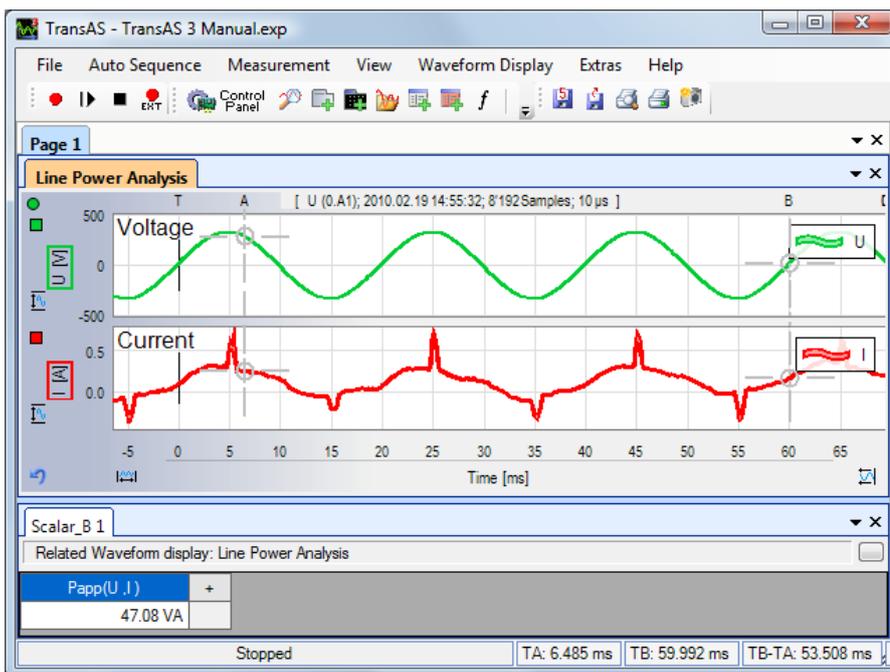
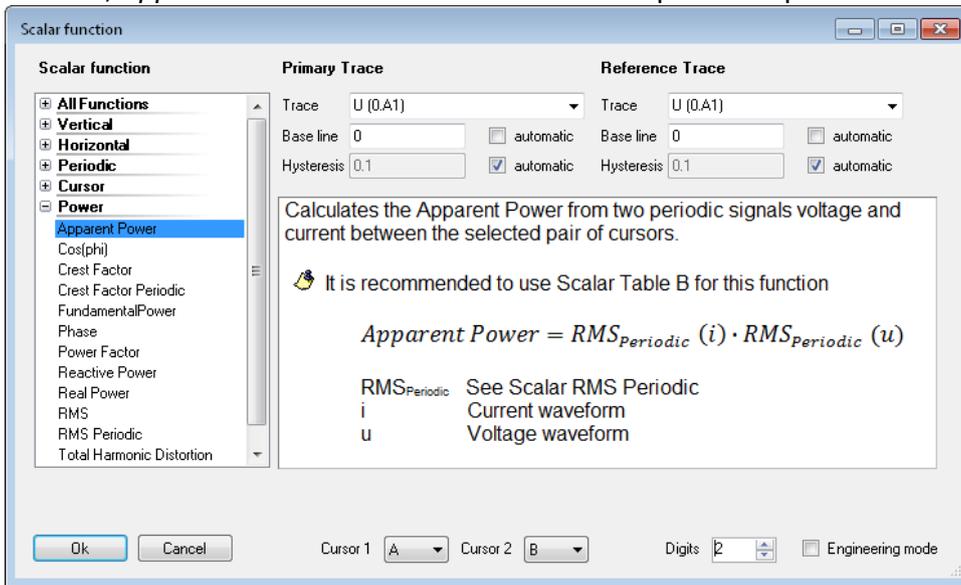
In case enabled [Auto-Refresh](#) , the table will be updated also if Single Trigger Mode is disabled.

25.2.2 Example 1: Apparent Power

- Apparent Power - Power Scalar Function; with Reference Channel; with Baseline and Hysteresis

Double-click on the Scalar Function "AbsMax" from above and open the list of the Power Scalar Functions. Select *Apparent Power*. The equation is shown in the right hand side of the list. Chose the Primary Trace (let's assume this is a line voltage signal) with Base line 0V and Hysteresis automatically, which is 10% of peak to peak value of the trace. Chose the Reference Trace (let's assume this is a line current signal) with Base line 0V and Hysteresis automatically, which is 10% of peak to peak value of the reference trace. By default, results are displayed in non-Engineering mode with 3 Digits. Change it to Engineering mode with 4 Digits.

Click OK to accept the modifications to the Scalar Function table. At the time of closing the window, *Apparent Power* will be calculated and replace the previous scalar.



The Apparent Power is now calculated and shown in the first column. To the right of the scalar *Apparent Power* a new empty column was automatically inserted.

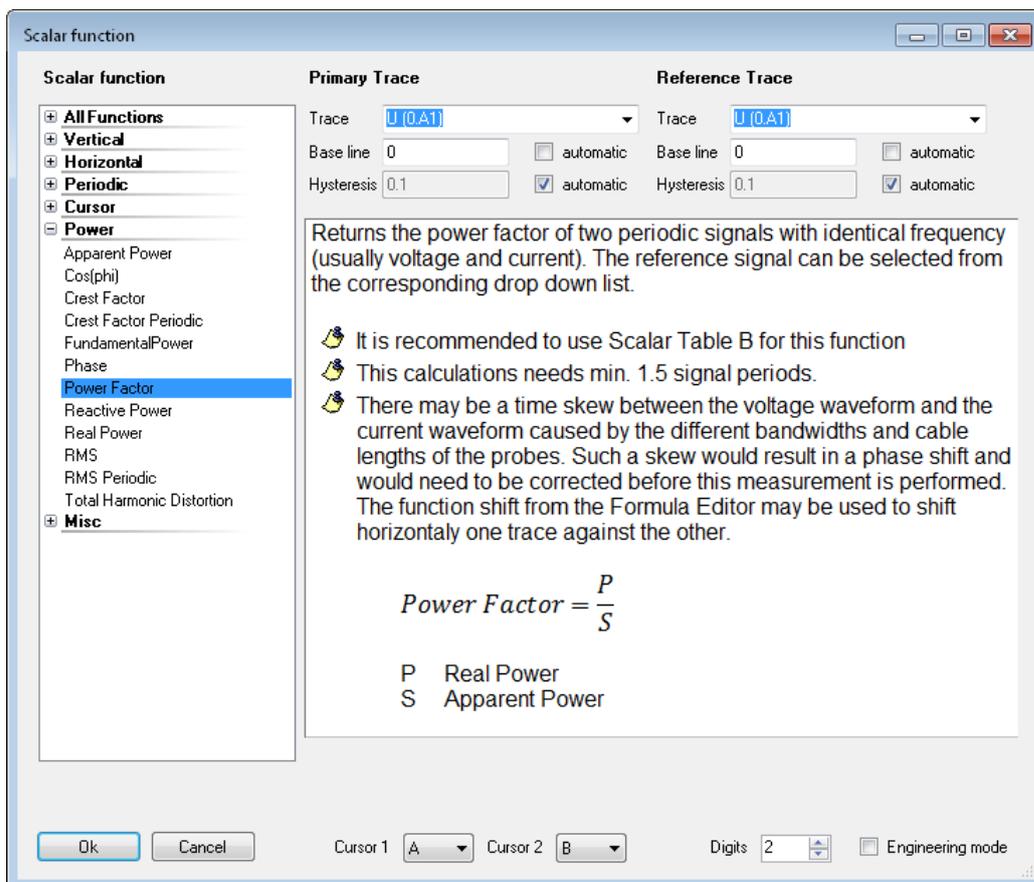


The Scalar Function will just be calculated between the two selected cursors (see [Scalar Functions Description Table](#)). Move to the associated Waveform tab and place the active cursor in order to cover the desired waveform region.

25.2.3 Example 2: Power Factor

- Power Factor - Power Scalar Function; with Reference Channel; with Baseline and Hysteresis

In this example a Scalar Function will be appended to the table. Double-click on the +, the next free column-header. - Open the list of Power Scalar Functions. Select *Power Factor*. The equation, description and boundary conditions are shown on the right side of the list. Engineering display mode is selected with 4 Digits, because the Scalar Function window remembers the last selection. Click Ok to close the window and to add the function *Power Factor* to the table.



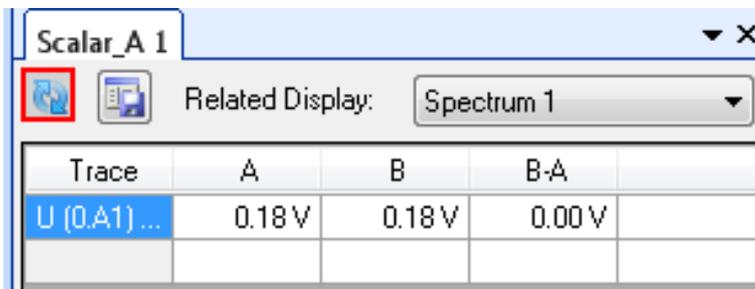
The power factor is now measured at the baseline level 0.0 V at a hysteresis of 10% of peak to peak value of the respective trace between the cursor A and B and the channels U(0.A1) and I(0.A3).



Some Scalar Functions are measured at a baseline level with a hysteresis. The [Scalar Functions Description Table](#) describes all Scalar Functions and gives information about the requirement to set the baseline level and the hysteresis.

25.3 FFT Function (Table A/B)

Open a new Scalar Table (A or B) with focus on an **FFT waveform**, the "**Related Display**" for this scalar table will be the frequency domain - spectrum waveform. Existing spectrum traces will automatically be added to the scalar table.



Trace	A	B	B-A
U (0.A1) ...	0.18V	0.18V	0.00V

Generally, the use of scalar tables with spectrum waveforms is the same as for waveforms with time domain signals. **The list of available functions is optimized for analyzing spectrums.**

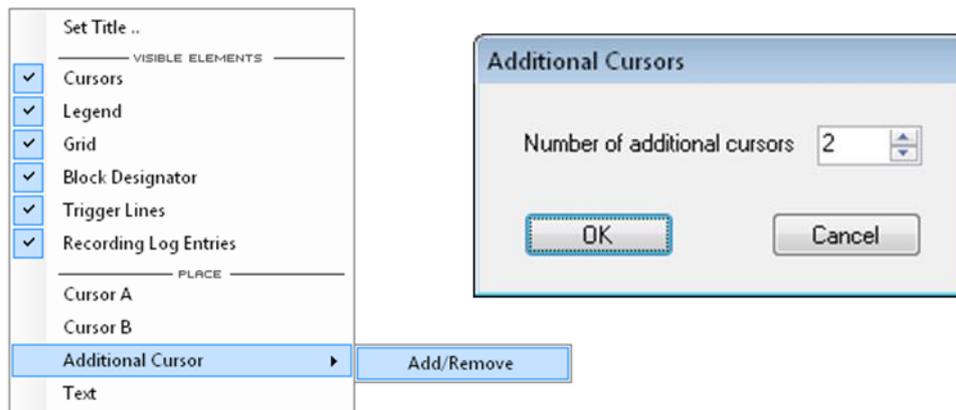
The following functions can be used and calculated:

- [Cursor Amplitude](#)
- [Cursor Delta Amplitude](#)
- [Cursor Delta Position](#)
- [Cursor Position](#)
- [Cursor Ratio Amplitude \(dB\)](#)
- [Frequency at Maximum](#)
- [Maximum](#) (between two cursors)

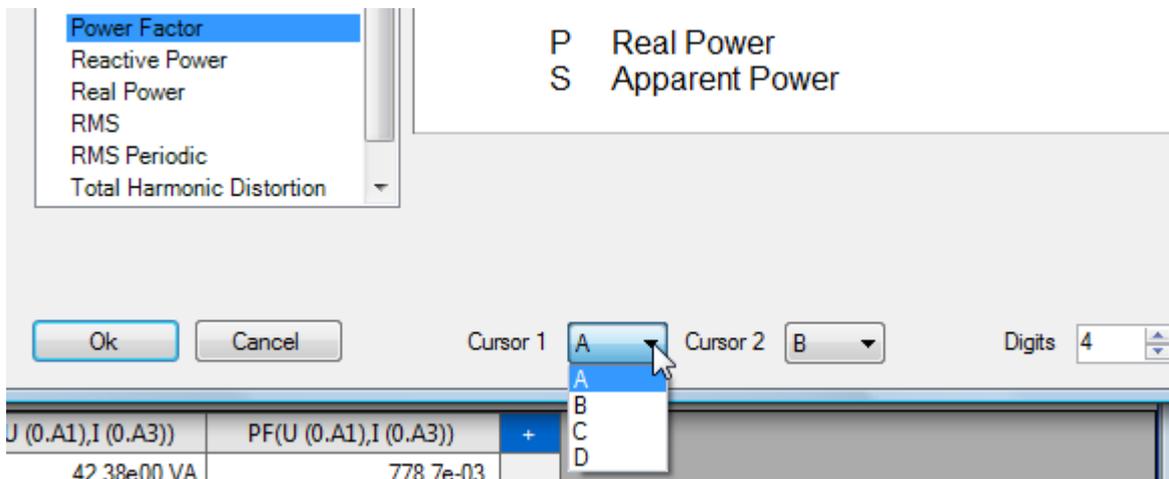
The "Related Display" can be changed for every scalar table from time based waveforms to a spectrum waveform or vice versa. After changing, the traces need to be dragged into the table again (**Drag & Drop** from Control Panel or Waveform). Also, the functions of the table may have to be **adjusted**. For example, the function "Mean" can't be calculated for a spectrum waveform.

25.4 Additional Cursors

If more cursors are required to define multiple measure gates, **right-click** on the waveform display or go to menu **"Waveform Display" / "Additional Cursor" / "Add/Remove"** and then specify the number of additional cursor (additional to the cursor A and B).



If Numbers of additional cursors is 2, then there will be in total 4 cursors in the Waveform Display, cursors C and D in addition to cursors A and B. Once the additional cursors are activated, they can also be selected in the **Scalar Function** dialog window.



25.5 Auto-Refresh of the Scalar Function Table

In **Single Shot mode** the Scalar Function table is refreshed with every acquisition. In case Single Shot is set to "OFF" the table is only updated with the Auto-Refresh button in the ON state (green frame).

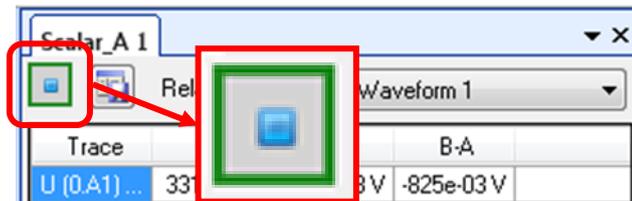
In the image below, **Auto-Refresh is turned off (red frame)**.



Since auto-refreshing of the Scalar Function table has a big **impact** on system **performance** it is recommended to turn it on only if required.

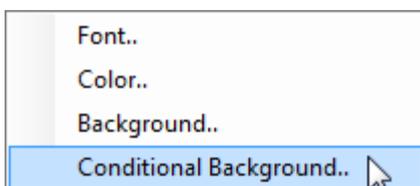
If the Waveform Display refresh-rate is more important than the calculation of Scalar Functions it is recommended to turn off the auto-refresh of the Scalar Function table.

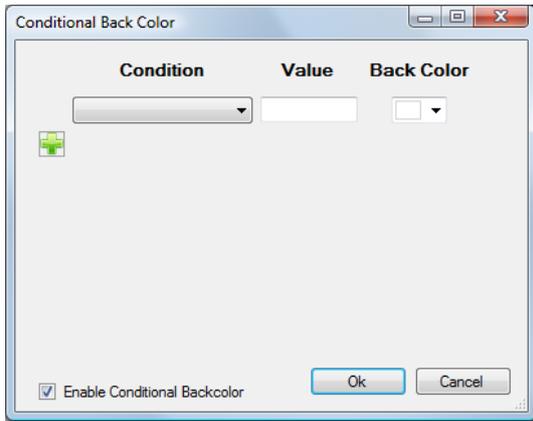
In the image below, **Auto-Refresh is turned on (green frame)**.



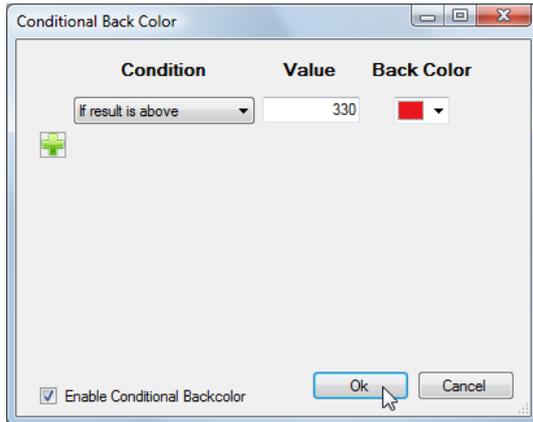
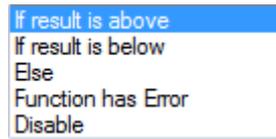
25.6 Conditional Background Color

Conditions can be defined to visually inform the user about certain **conditions** of a **Scalar Function**. With a right-click on a Scalar Function readout and the selection "Conditional Background.." a window opens to define the condition(s).

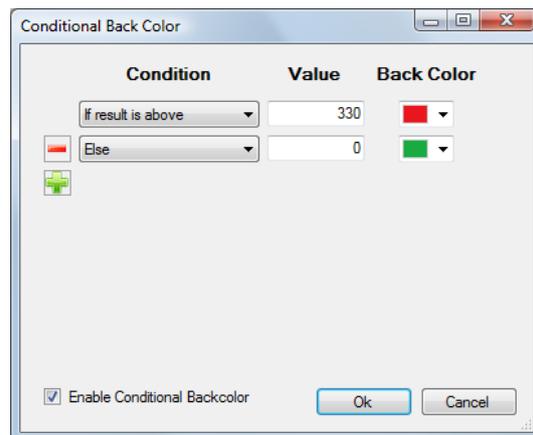




One or several conditions can be set. The Condition selection lists shows all available options



Below the condition is set to display a red back color in case the measured value is above 330V.



Another condition can be added with a click on the  button.

The second condition line is simply saying that the color should always be green in case the first condition is not met.



It is recommended to list to the values in increasing or decreasing order.

Finally, the background color needs to be activated and the window closed with Ok. The chosen Scalar Function result is now displayed in the conditional background color.

A(A1 (0.A1))	A(I (0.A3))	PP(A1 (0.A1)) [A,B]	+
23.1766 V	-0.0575 V	340e00 V	
A(A1 (0.A1))	A(I (0.A3))	PP(A1 (0.A1)) [A,B]	+
23.1766 V	-0.0575 V	215e00 V	

If the voltage is **higher than 300V**, the background color of the cell turns **red**.

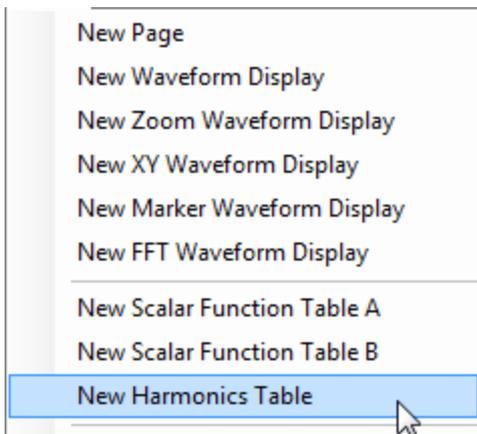
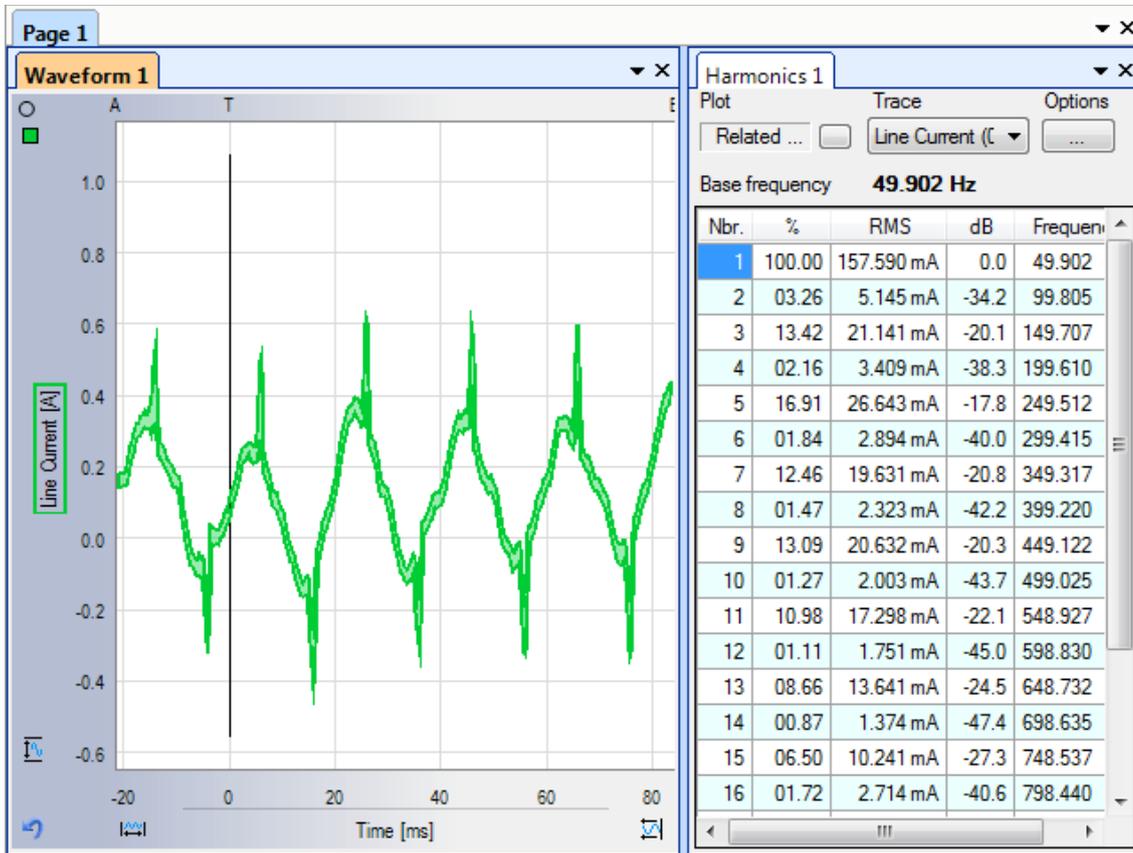
Then background color stays **green** for voltages **below 300V**.

26 Harmonics Table

A **periodic time domain signal** can be converted to the frequency domain with a **Fourier analysis** (FFT) and separated into the fundamental frequency and its harmonics along with its magnitudes.

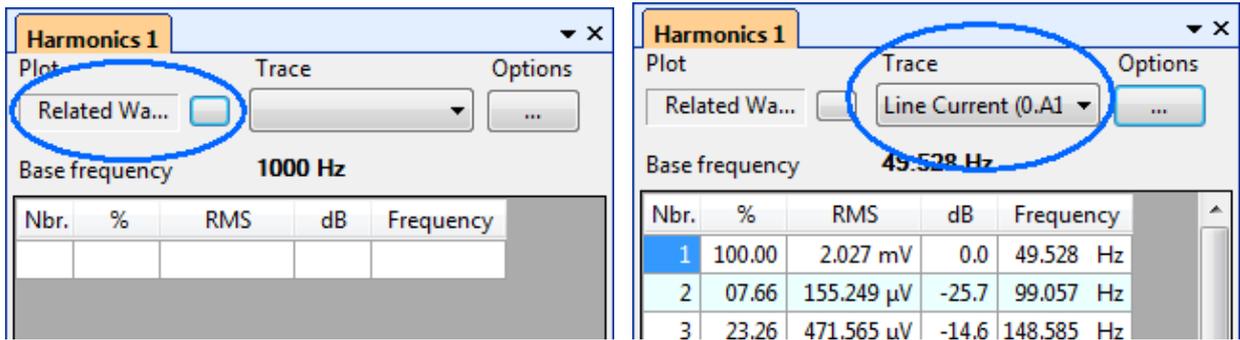


The Harmonics Table window can be positioned either on any side of the current waveform display or it can be kept as a tab. See the section [TranAX overview](#) for further details on the window docking technique.

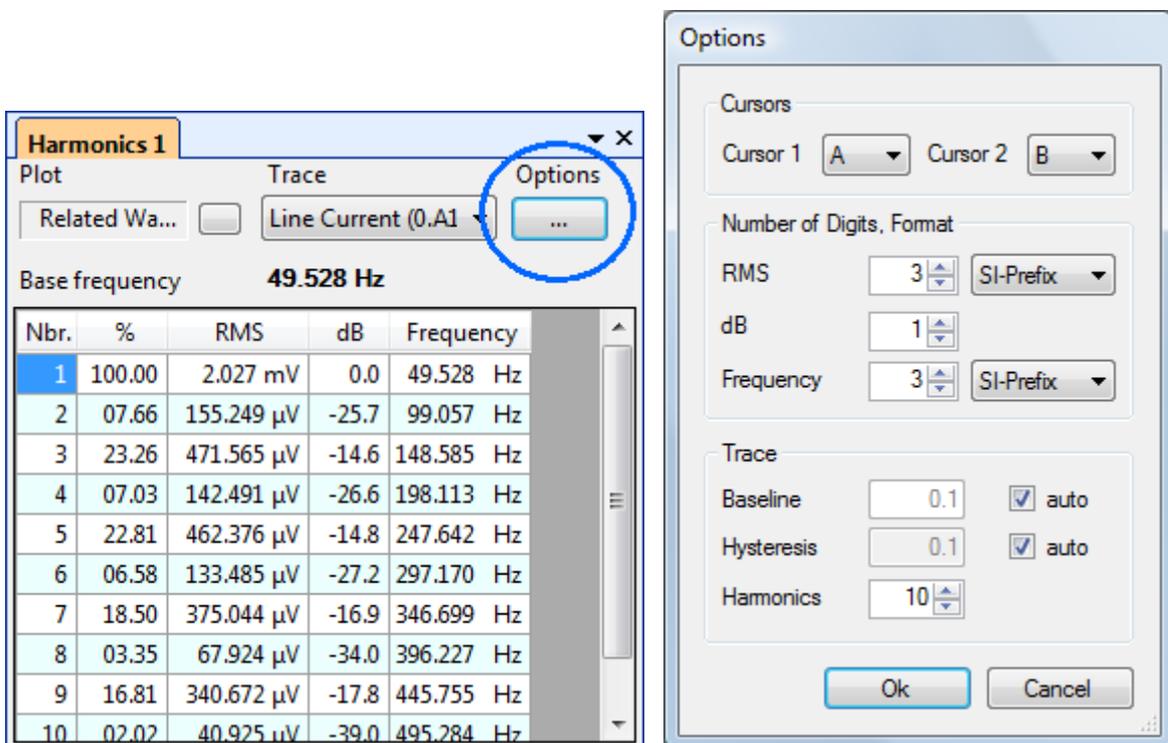


A new Harmonics Table can be opened from menu **"View"** / **"New Harmonics Table"**.

The new table needs to be **related first to a Waveform display** and then to an available trace within the selected Waveform display.



In Options it can be configured between what cursor, at what level and hysteresis the trace will be analyzed.

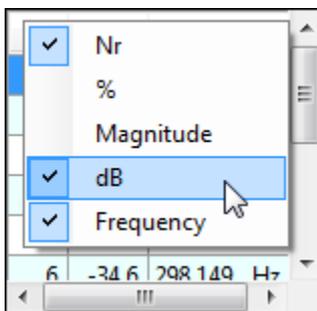


- **Cursor:** The harmonics will be calculated between the two selected cursors. A minimum of two signal periods are required.
- **Number of Digits, Format:** The format is chosen in which the result will be displayed
- **Trace:** The Baseline and Hysteresis are either set **automatically** or if auto is not selected a value will be set. In the control **HARMONICS** the number of harmonics to calculate can be in the range of [1 ... 100].

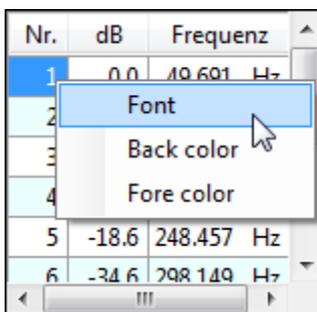
The table can be customized in such a way to display the columns in the desired order and to display only the columns of interest.

Nr.	%	RMS	dB	Frequenz
1	100.00	143.826 mA	0.0	49.691 Hz
2	01.02	1.469 mA	-45.8	99.383 Hz
3	17.46	25.106 mA	-17.5	149.074 Hz
4	03.70	5.315 mA	-33.0	198.766 Hz
5	15.62	22.469 mA	-18.6	248.457 Hz
6	03.14	4.518 mA	-34.6	298.149 Hz
7	12.16	17.483 mA	-21.1	347.840 Hz
8	02.22	3.194 mA	-38.1	397.532 Hz
9	11.37	16.347 mA	-21.7	447.223 Hz
10	01.74	2.597 mA	-40.5	496.914 Hz

Drag & Drop to change **column order**.



Right click on header to **turn on or off** columns, so you can see only the important information.

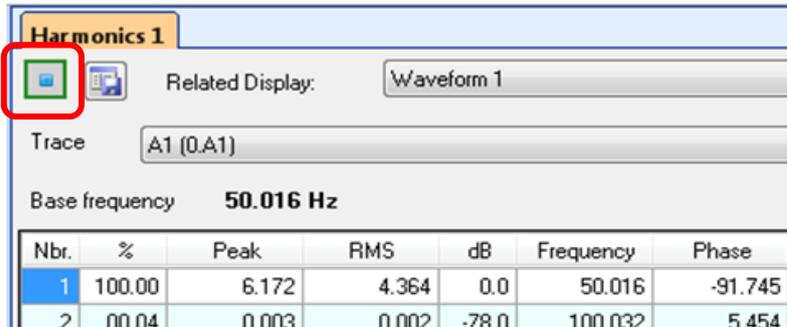


Right click on any cell in order to change font, back- or fore **color**.

26.1 Enhanced Harmonics Table

Export function: By clicking the button  the entire table will be exported into a text file. This ASCII text file can be imported afterwards into other programs like Excel. This will be useful for further use and analysis of the measured data.

Update during measurement: With the button in the upper left corner, updating e of the table values can be enabled or disabled.



Nbr.	%	Peak	RMS	dB	Frequency	Phase
1	100.00	6.172	4.364	0.0	50.016	-91.745
2	00.04	0.003	0.002	-78.0	100.032	5.454

In this picture (green frame of the icon), **updating during measurements is enabled.**

Peak	RMS	dB	Frequenz
2.992e-01	<input checked="" type="checkbox"/>	Nr	50.000
6.318e-05	<input checked="" type="checkbox"/>	%	100.014
4.429e-03	<input checked="" type="checkbox"/>	Peak	150.021
3.247e-05	<input checked="" type="checkbox"/>	RMS	200.021
9.294e-03	<input checked="" type="checkbox"/>	dB	250.031
6.902e-05	<input checked="" type="checkbox"/>	Frequency	300.041
1.541e-02	<input checked="" type="checkbox"/>	Phase	350.041
7.730e-05	<input type="checkbox"/>	0.000	400.051

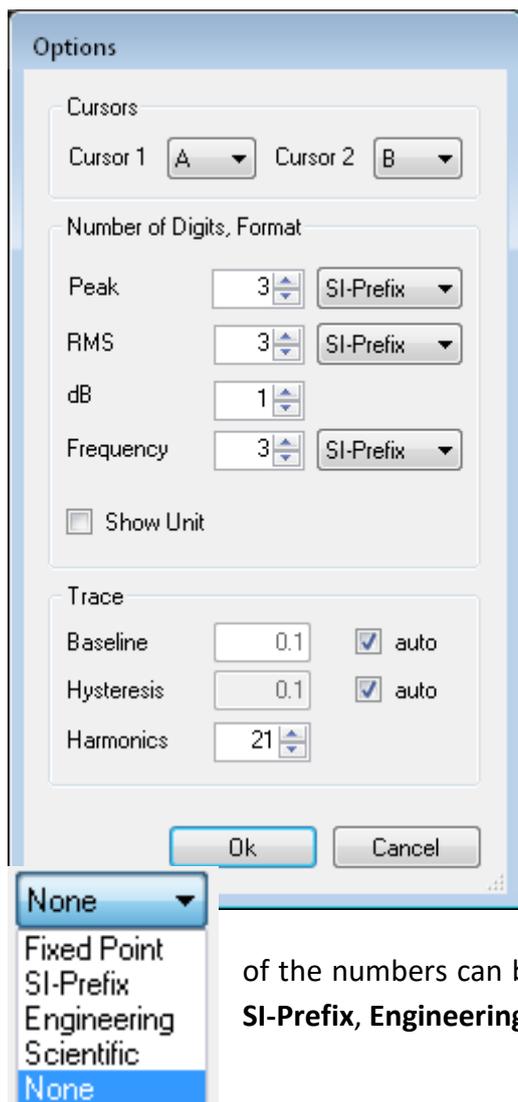
With a **right click** on the title bar on the top of the table, each column can be **enabled or disabled**. Only the checked columns will be displayed.

The calculated values for "**Peak**" and "**Phase**" of the Harmonics are now also available.



Please make sure, that the **cursors are placed correctly**. For calculating the Harmonics, **at least two whole periods** of the fundamental waveform are required!

In the menu "**Options**", the following settings can be made:



Cursors: A and B are the default cursors.

Number of digits and format: For each single column the number of digits and format can be selected.

Show Unit: If this box is checked, the units of the columns are visible. It is recommended to uncheck this option when exporting the Harmonics table to external programs. For further calculations, e.g. in Excel, it is required to have the pure numbers.

Trace: Baseline and Hysteresis can be either set manually or (as recommended) automatically. For very noisy signals, especially the Hysteresis may have to be set manually. The number of calculated Harmonics can be defined.

For each column in the harmonics table the formatting of the numbers can be defined. It's possible to choose between **Fixed Point**, **SI-Prefix**, **Engineering**, **Scientific** and **None**.

Example: Value for "Peak" is 23'070.20897 V

Format	Digits	Visible Value
Fixed Point	3	23'070.209 V
SI-Prefix	3	23.070 kV
Engineering	3	23.1e03 V
Scientific	3	2.307e04 V
None	3	2.31E+04 V

27 Formula Editor

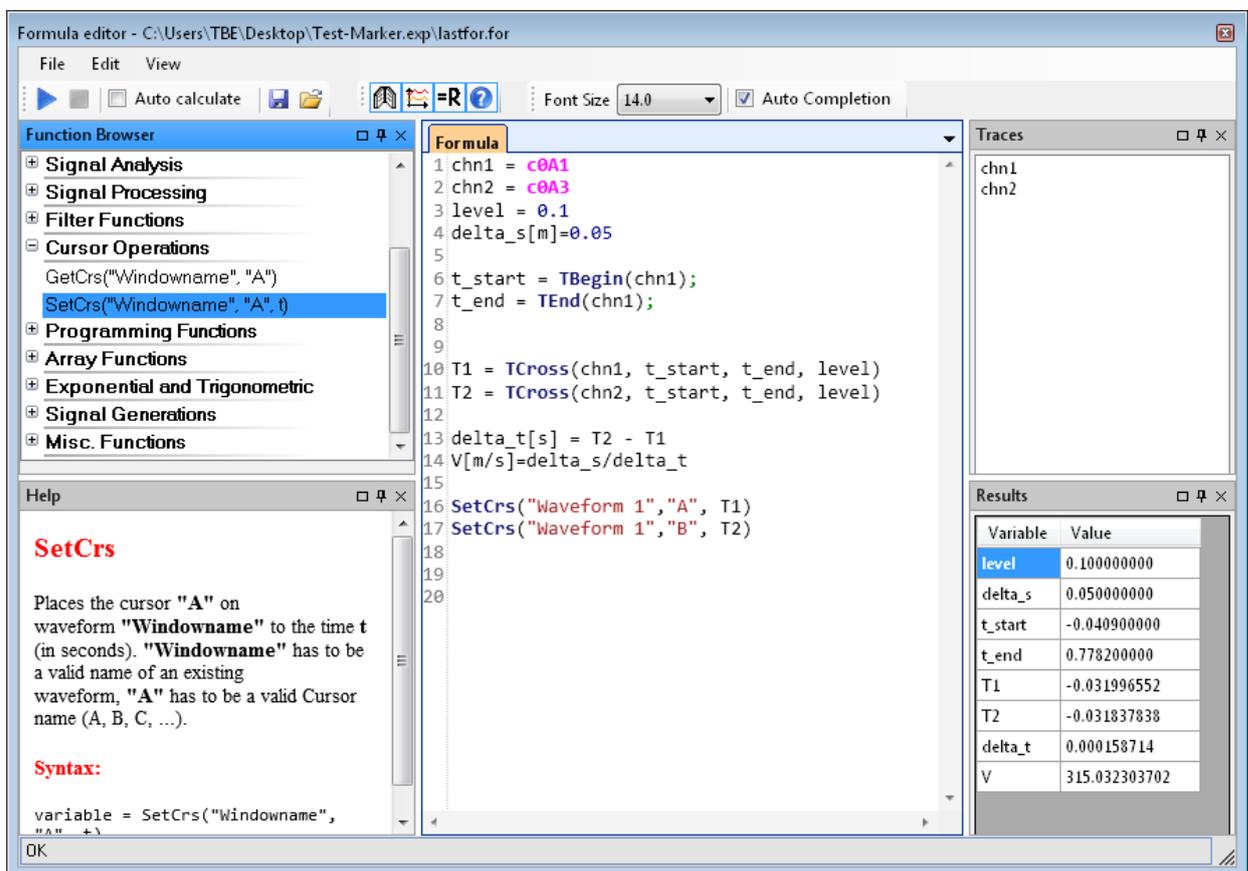
TranAX includes a powerful formula editor with a large variety of useful functions for analyzing measured signal curves and for further processing, out of which new signal curves and waveform measurements can be calculated easily.

With more than **100 mathematical functions** and commands, almost any practical calculation can be performed.

A complete list of functions can be found in the [Appendix](#).

The Formula Editor offers the convenience of current programming environments: syntax highlighting, auto completion, scalable font size, etc.

Next to the Main-Formula-Text field more tabs with other formula and function files can be added. They are used primarily to inspect, compare, change, etc., existing formulas or functions (in addition to Main-Formulas).



The formula editor consists of the following components:

- The **formulas for the calculations** are entered in the text box in the middle labelled "**Main Formula**". Each line represents a function and normally each function is separated by a line break.
- To the **right** is the column for **Results**, with scalar values at the bottom and the calculated signal curves at the top. The signal curves can be drawn using drag & drop into the [waveform display](#).
- Scalar-values in "Results" also can be made visible in the [Text-Boxes](#) of the curve display.
- Top **left** is a list of all **available channels and instructions**. They can be added by double clicking on the text box. The help section for each function is at the bottom of the left hand column.

27.1 Using the Formula Editor

To document formulas, comments can be added. They are marked by a semi-colon ";". All that is written after the semi-colon will be ignored and interpreted as commentary.

In addition, in a given comment a word (or part thereof) can be marked and linked to a file. This file will then be stored (invisible) in the formula storage file. It can contain extensive descriptions or instructions on the formulas and can be opened with a right-click on a designated screen position. PDF, Text and Picture files can be attached.

The formulas for calculations are entered in the text box "Formula" like the following:

```
name[Unit] = Expression
```

name is the name of the variable, which identifies the result. The result can be used in subsequent calculations for further calculations. **unit (in square brackets)** is used to assign a unit to a calculated signal curve. This is determined by the units of the signal curves that flow into the calculation, as well as any scaling.

Example: COA1 measures a voltage in volts, COA2 measure a current in amperes. Then the following calculation would be correct:

```
power[mW] = c0A1 * c0A2 * 1000
```

The variable "name" can be chosen arbitrarily. Keywords (e.g. cos, sine, if, for, to, as, Pi, etc.) are not allowed and will generate a syntax error.

On the **right side** of the **equal sign** can be **any expressions** which may contain the **basic operations** of addition, subtraction, multiplication, division and power function. The minus sign is also accepted (e.g. -4 or -c0A1). Multiplication and division have higher precedence than addition and subtraction. Parentheses can be used if required. The operands can be **numbers, channels, signal curves from TPC5 or TDP files, strings, arrays** and previously calculated **results**. In general, the internal calculations are performed in double precision (64, 15 decimal places, exponent > 300). The values of stored signal curves are saved as a 32Bit number.

Channels are indicated with a "c", the **device number** and the **input name** (e.g.: COA1, COB3). For recordings with more than one block (Multi Block or ECR), the **block number** has to be given (e.g. COA1.2), otherwise the signal from block 0 is used for calculation. With an **apostrophe (')** and a number between 1 and 2, a marker (digital input) of the channel can be selected:

```
Marker 1 = c0A1'1
```

```
Marker 2 = c0B3'2
```



Results of Markers are represented by analog values 0 and 1.

Signal curves from files: A file can be selected with its name:

```
File("filename.tpc5", 1)
```



With the right mouse button over the filename of a File Function the file can be replaced via the Windows Explorer.

The index represents the signal curve number in the file (0 corresponds to the first signal curve. not to be mixed with the channel number!). In the example above, the second signal curve of the file is used for the calculation.



With the right mouse button over the Index of a File Function all signals in the file will be listed by its name.

Should marker data be used instead of analog data, this can be specified by a third parameter:

`File("FileName", Index) .Block 'Marker`

Marker is a value between 1 und 2. Example:

`Signal = File("crash.tpc5",0) '1`

If no path is specified with the filename, the program searches the signal curve in the **DATA** directory of the **current Experiment**. For a file in another directory within the current Experiment (e.g. "C:\User\USERNAME\Documents\TranAX\EXPERIMENT.exp\ref\name.TPC5" it does not need to specify the full path name. The following term is sufficient: "..\ref\name.TPC5".

In each row, text after a **semicolon (;)** will be ignored by the formula interpreter until the end of the line. This can be used for explanatory **comments** to be entered.

27.2 Place Cursors

The function **SetCrs** ("Waveform Name", "A", t) **places the cursor** inside the waveform display. The resulting position of the cursor in the waveform window refers to the X-axis and depends on the set X-axis scaling (relative to the start, relative to the trigger time and samples).

For example, this feature is useful for calculating the area of a pulse with a scalar table. For this purpose, the time for rise and fall of the pulse can be found by using TCross. Afterwards, two cursors are set to the **detected pulse limits** using SetCrs ("...", "A") and SetCrs ("...", "B").

Another example:

Determinate start and end time of the signal curves:

```
t_start = TBegin(c0A1)
```

```
t_end = TEnd(c0A1)
```

Looking at two signal curves for the first increase over 1V on each signal curve:

```
T1 = TCross(c0A1, t_start, t_end, 1)
```

```
T2 = TCross(c0A2, t_start, t_end, 1)
```

Place the cursors at the two crossings found:

```
SetCrs("Waveform 1", "A", T1)
```

```
SetCrs("Waveform 1", "B", T2)
```

27.3 String Variables

String variables e.g. "Waveform 1" or "A" can be declared using the appropriate functions. Example:

```
value$ = "test.tpc5"
```



The \$ character in a string variable name is used only for better differentiation. A string variable can be specified without a \$ sign.

Multiple strings can be linked together (concatenated):

```
name$ = "test"
```

```
extension$ = ".tpc"
```

```
version = 5
```

```
value$ = name$ + extension$ + version
```

For string operations, given numbers are automatically interpreted as a string. For example, version=5 has not to be written as version="5" (the function version= 2*2.5 would provide the same result).

27.4 Assigning of Sub-Functions

Frequently used function blocks can be assigned into separate sub-functions. This is for ease and clarity of the main program.

```
Function Name (parameters*)
```

```
Function BLOCK
```

```
Name = value ; return value
```

```
EndFunction
```

The return value is determined as a formula with the variable name that corresponds to the function name.

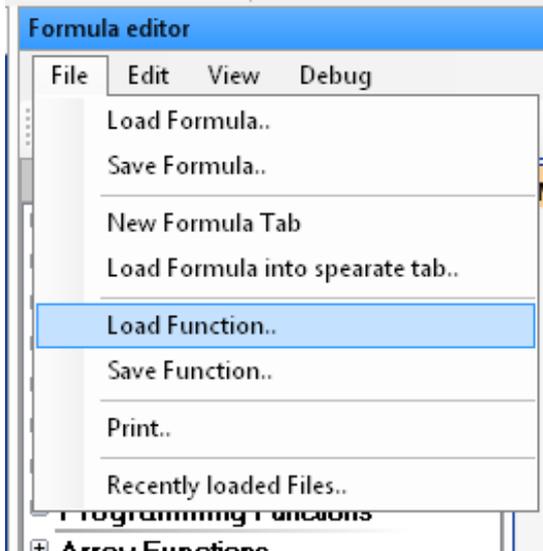
Also, several sub-functions can be defined in a single file.

Example, a file named "MathOperators.fnc"

```
Function Addition(value0, value1)
Addition = value0 + value1
EndFunction
```

Normally sub-functions are stored in dedicated files. The name extension of these files is *.fnc.

By the menu "File" / "Load Function" resp. "Save Function" these files can be administrated:



To use a sub-function, the file containing the sub-function must be placed on top in the main formula section. This is done using the keyword "using":

```
using "MathOperators.fnc"
result = Addition(2, 3)
```



It is also possible to place sub-functions below the main formulas after the instruction **EndFormula**. This feature helps developing sub-functions. In such a case the command **using ()** in the main formulas may be omitted.

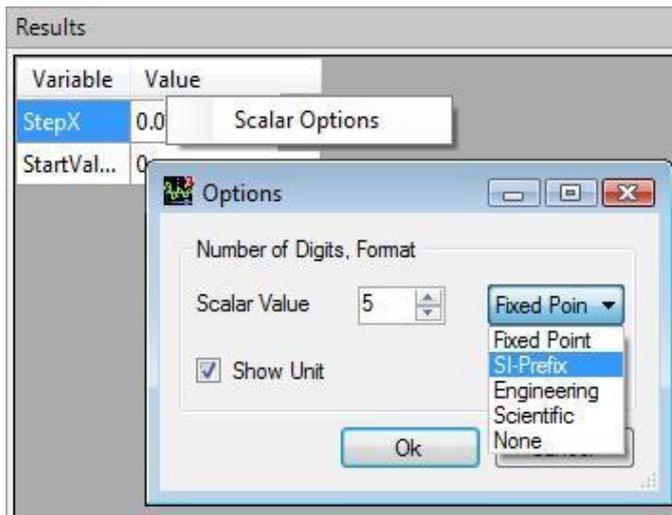
Sub-function-calls also can be found in other sub-functions.



A collection of standard sub-functions is included with a TranAX delivery. With that there are virtually no limitations to the number of calculations. The user also has the opportunity to set up a proprietary collection of functions.

27.5 Number format for scalar results

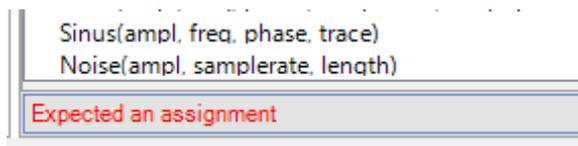
The representation of resulting scalar values can be adjusted by right-clicking on the cell "value".



27.6 Error Messages

Formula errors detected by the parser are displayed at bottom left.

Double clicking on a possible error message at the bottom of the formula editor jumps directly to the location of the error inside the formula. This shortens the search for the corresponding error source. Often the fault is located in the immediately preceding formula.



27.7 Groups of Functions, Overview

A complete list of functions can be found in the [Appendix](#). The available mathematical functions are separated into the following groups:

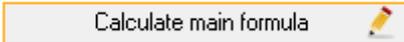
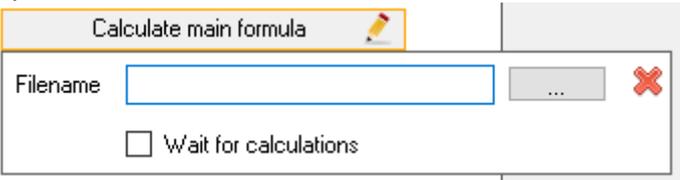
Name	Description
Channels	Hardware Signal Sources
All Functions	All available functions in alphabetical order
Base Functions	Base functions for calculations
File Functions	Operations related to files
Signal Analysis	In general, individual values as a result
Signal Processing	In general, curves as a result
Filter Functions	Various filter functions
Programming Functions	If Then, Loop etc.
Array Functions	Functions for working with arrays
Exponential and Trigonometric	Trigonometric functions
Spectrum (FFT)	Calculation of spectra
Report Generator	Building of reports
Recording Parameters	Hardware settings
Layout Waveforms	Property of waveform displays
Auto Sequence Functions	Auto sequence macros
Signal Generations	Functions that generate curves
Misc. Functions	Several additional functions

28 Measurement Flow Control (MFC)

The MFC replaces the well-known Auto Sequences from the previous versions but with many new functionalities. The MFC is a powerful tool for measurement automation. Define once your measurement flow and never lose any important data again.

In TranAX there is a new ribbon bar labelled "Measurement Flow Control"

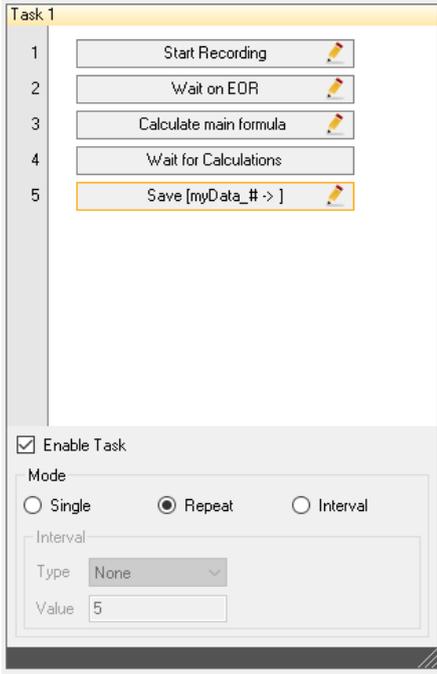
28.1 MFC Functions

Abort Sequence	Stops the Measurement Flow, same function as pressing the Stop button or button "F8".
Autocalibration	Starts an Autocalibration of the connected devices.
Beep	Activates the Windows system sound.
Calculate	<p>Calculates the Main formula in the Formula Editor.</p>  <p>Optional a formula file with the extension *.for can be selected.</p>  <p>If "Wait on calculation" is enabled, this command will wait until the calculation of the formula is done. This has the same effect as "Calculate" in combination with "Wait for Calculations".</p>  
Call	<p>Calls an external application or batch file.</p>  <p>Filename defines the name of the application, Arguments the console parameters or arguments. In case of the application doesn't close, there is a possibility to define a timeout.</p>
Comment	<p>As it says, just a comment field for some additional information for the sequence or Task. Documentation inside a Task or information about date and author of this Measurement Flow Task.</p> 
Delay	Waits the defined time in seconds. Allowed are integer and float number. Long delays will be visualized with a countdown in remaining seconds.

Disable External Start	TraNET devise have an External Start input on its 25pin D-sub connector of the Starhub. This command disables the External Start function. See also "Enable External Start"
Enable External Start	Enables the External Start input. See also "Disable External Start"
Event	Creates an event, which can be used in another Task to start
Exit Loop	Quits a running loop, mostly used in combination with an If Else statement, e.g. to check the status or value of a formula
Formula	Formula calculates the formula in its sliding window. See also Calculate, which calculates the main formula tab of the Formula Editor
If-Else	If-Else statement, check a variable value form a formula and decide how to proceed. Please note that each Task runs independent, formula values, traces etc. have to be transferred with WriteforNext and ReadofPrevious command.
Load Formula	Loads a Formula file to the main formula Tab
Load Layout	Load a saved Layout file
Load Settings	Loads a hardware settings file
Pause	Shows an information dialog, which has to be confirmed.
Print	Printout a Page or Waveform to the defined default printer (the curves will be updated automatically after each measurement). The layout can be defined with the print preview option. Please use the name of the Page or Waveform which will be printed.
Quit Application	Closes TranAX. In case of changes and activated write protection, a dialog will appear first. Without write protection, settings will be saved and TranAX closed.
Repeat	Used in Combination with Next. Number of repetitions can be defined, Repeat Forever is possible to. Please note that every Task has the possibility to for Repeat Forever in its mode settings in the bottom section of the Task window. Also available options are Singe and Interval.
Save	Save recorded traces to a file, *.tpc5
Save spectrum	Save a calculated FFT traces to a file, *.tps5
Start Recording	Starts a new recording, often used in combination with Wait on EOR, this can also be activated directly in this function. It also allows to save recorded traces directly to a reference.
Stop Recording	Stops a running recording
Store Page	Saves a whole page to a *.tpd file
Store Readout	Stores the Scalar tabled data to a text file.

Store Snapshots	Save a screenshot of the defined Page or Waveform. File format and resolution can be defined in the TranAX settings.
Trigger	Raises a Trigger Event
Wait for Calculations	Waits until the calculation of the main formula, Scalar table A and B and Harmonics Table is done. Used in combination with Calculate.
Wait on Armed	Wait until the System is armed
Wait on Device	Waits until the device is online, in case of a disconnected TraNET device
Wait on EOR	Waits until the Recording is done (End Of Record, EOR), used in combination with Start Recording
Wait on Event	Wait on an Event. An Event can be raised in another task
Wait on Trigger	Wait until the system has triggered. Manual or a trigger event.

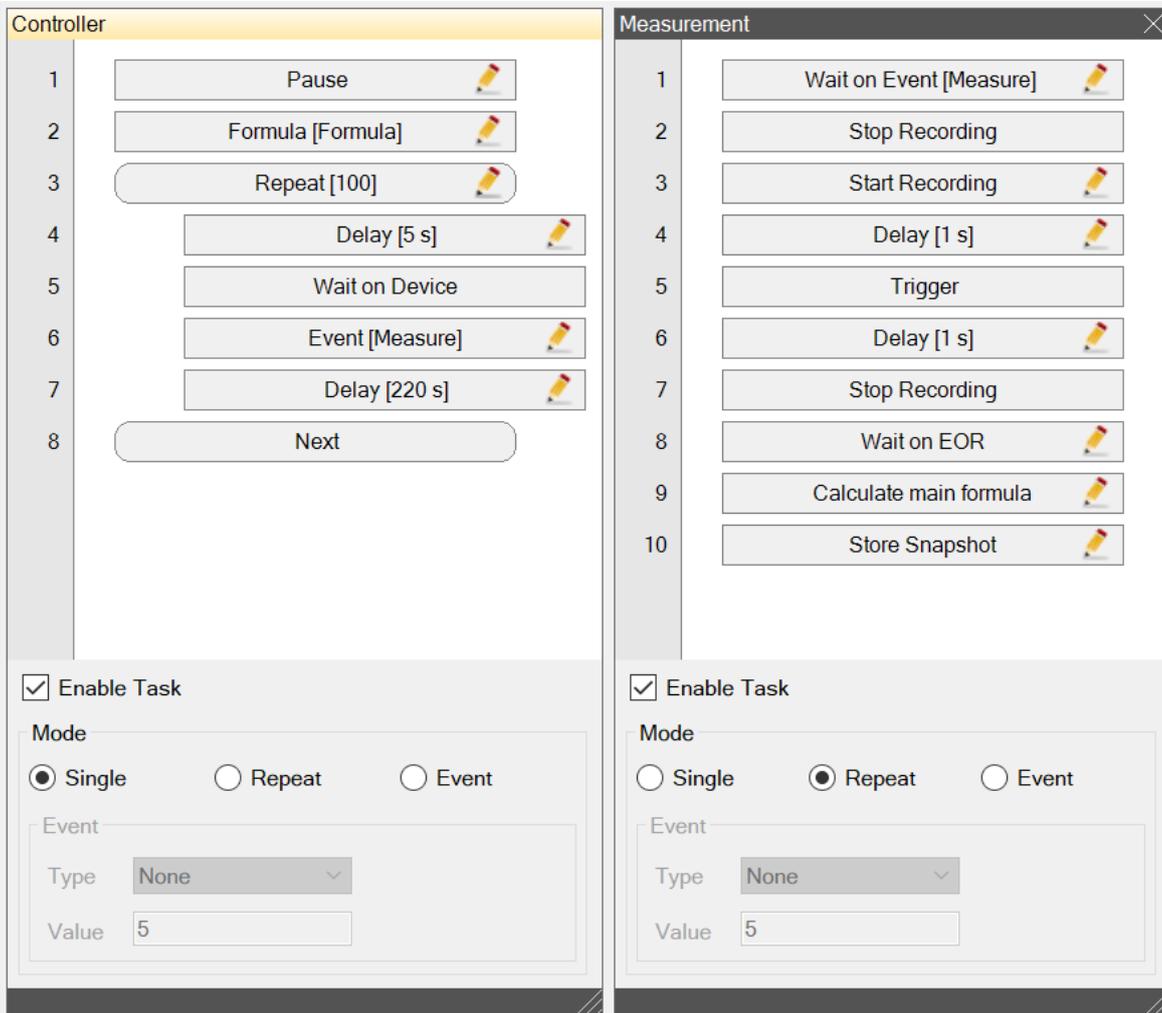
28.2 MFC Examples



Simple Example:
Measuring of data, calculation of main formula in Formula Editor and finally save some traces to a file.

Example with two MFC Tasks:

Controller checks for the TraNET device to be ready (powered on and connected), then starts recording, does some calculation with the Formula Editor and finally saves a snapshot of the waveform.



29 Averaging over multiple recordings

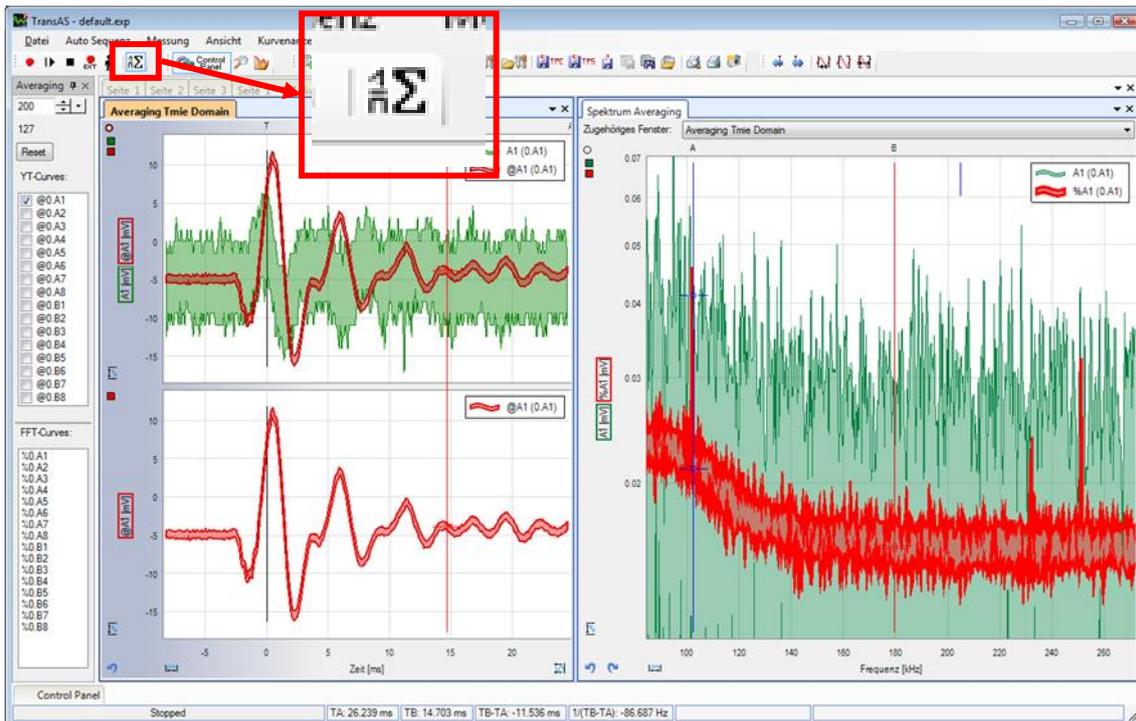
The averaging function reduces noise on periodically measured signals. Each recording will be added up to the previous signal and the calculated summation average will be shown as a new trace.



Averaging can only be used in [Scope-Mode](#).

The button  has to be pressed to enable the averaging functionality. The averaging window will be opened during the measurement. We recommend to position this window on the left side.

29.1 Averaging in the Time Domain



The lower list (FFT-curves) is used for [Averaging in spectrum](#) range

In the upper part of the window is a textbox for entering the number of records for averaging. The counter below shows the number of acquired records. If the pre-set limit is reached, the averaged result will be deleted and a new series starts. The reset button will delete the averaged signals before reaching the limit.



Pressing the "Reset" button will flag internally, that a next series will restart with record one. The averaged trace will still be available until the next record starts.

The List below shows all available channels in the system. Each channel individually can be selected whether or not it should be in averaging mode.

This list can also be used to drag & drop the averaged curves to a waveform. Left click with the mouse pointer on one trace, drag it to the waveform display area and release the mouse button. To be able to distinguish a normal trace from an averaged trace, the averaged traces are marked with an "@" at the beginning of the channel name.

After each record, the traces will be added to the averaged curve and redrawn in the display. The actual averaged curve will always be visible. After reaching the pre-set limit, the averaged final result will be deleted and a new series starts.

If Single Shot is disabled in the Control Panel and Auto-Start is active, continuous measuring will be ongoing and every trace will be added to the averaged curve until the limit of records is reached. Further recording will be inhibited. By pressing the Start button a new series will be recorded and averaged.

The actual calculated averaged curve can also be applied for new calculation with the formula editor, just add an "@" to the channel name, e.g. @c0A1 instead of c0A1 for Channel 1.

Example:

Save `xx.tpc5`, `@0A1`, `@0A3-4`



For saving averaged Traces in an Auto sequence, the installed software may need to be upgraded. The following **versions** are **prerequisites**:

TranAX:

3.2.1.702

(Menu "**Help**" / "**About**")

29.2 Averaging in the Frequency Domain

The same way as averaging can be used for time domain processing, it can be used for Spectrum traces. Only the amplitude values will be averaged, not the phases. Click the button  to enable the averaging function. In the lower part of the Average window all the channels are listed which can be used for spectrum averaging. A "%" symbol at the beginning of a channel name marks an averaged spectrum curve.

Averaged spectrum curves will only be calculated from traces that are currently active in a spectrum display.

The upper example shows the result of an averaged spectrum signal. There are now frequencies visible, which normally can't be seen because of the noise in the signal.



If there are changes in the parameters during FFT averaging (e.g. FFT weighting factors or changes in the timing of the corresponding Y-T-Waveform) the counter for averaging will be reset automatically.



Averaging uses CPU resources for calculation and also disk space. It is recommended to enable only the channels which are needed and to use the smallest block size possible (especially for averaged spectra).

30 Experiments und Experiment Sets

In TranAX the term "experiment" actually means a measurement project. Such an Experiment includes settings such as measuring range, sample rate, channel name, the arrangement of the windows, formulas, Auto Sequences etc.



Experiments from TranAX 3 and 4 are mostly compatible. Existing experiments can still be used with TranAX 4.

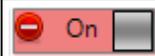
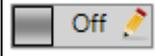
If a TranAX 3 Experiment will be loaded in TranAX 4, it creates a write protected Experiment Set.

30.1 Data structure of an Experiment

 example.exp	For each experiment, a subdirectory will be created. This usually carries its name plus the extension ".exp". This will be handled this way in TranAX 4 for backward compatibility.
 data  expr  example.exp  example.zip	Inside of the Experiment, there are the sub directories <i>data</i> and <i>expr</i> , within are the measured traces and curves, respectively the calculated traces in <i>expr</i> stored.
 example.aut  example.ctf  example.lay  example.tps.xml  Settings.xml  Snapshot.bmp	An Experiment Set is a Zip file, which includes all the settings: Layout, hardware settings, Auto Sequences, formulas, etc..
 example.exp	The file with the extension *.exp, with the same name as the Experiment, includes the information of the last used experiment Set for TranAX
<input checked="" type="checkbox"/> Load last experiment after startup	With activated checkbox "Load last Experiment after start up", TranAX 4 will run in "Legacy Mode" and will open the last used Experiment Set and settings. This can be enabled or disabled on the Startup Page of TranAX.

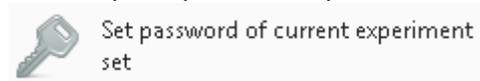
30.2 Write protection for Experiment Sets

In general, when closing TranAX, all changes in the experiment are stored into the current Experiment Set. For test applications, it may be advantageous that these settings are not changed and no changes in the original Experiment Sets are made.

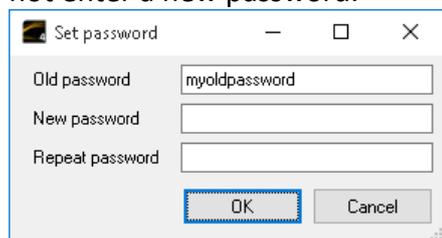
	<p>Write protection is activated. Changes to the layout and settings can be done, but they are discarded when closing. A corresponding dialog warns against the loss, so that they can still be saved in another experiment set.</p>
	<p>Write protection is turned off. Upon closing of TranAX any changes in the current experiment set is stored.</p>

30.3 Password protection for Experiment Sets

There is also the possibility to assign a password to the experiment set. If this is set, the write protection can still be further activated and turned off. When turning off the write protection, TranAX prompts for the password.



To disable the set password, the dialogue can be reopened, enter the existing password and do not enter a new password.




In case the password is no longer known, use a text editor to open in the Settings.xml file and delete the hash code of the password between `<Password>` and `</ Password>`. This will remove the password protection.

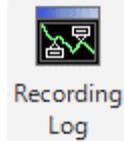


The functionality of the write protection and password protection is used purely to prevent unwanted tampering, either out of ignorance or carelessness. Both functions do not provide protection against malicious and Deliberate tampering!

31 Misc. Controls

31.1 Recording Log

Click the icon "Recording Log" in the Ribbon Tab "Layout", group "Misc. Controls" to open the Recording Log window.



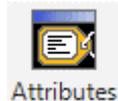
The record log lets you add comments to the signal data at any time during the [continuous](#) or [ECR mode](#). The entries will automatically be marked with a time stamp and can be displayed at the according spot on the time axis. The record log entries are also saved with signal curves to a TPC5 file and can be modified later on if required.



By enabling the Show Trigger option all trigger events will be listed including the timestamp. You can also add your own event comments by pressing the Add button.

31.2 Attributes

Click the icon "Attributes" in the Ribbon Tab "Layout", group "Misc. Controls" to open the Attributes window.

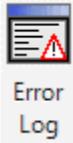


Name	Text
ATTENT...	Server restarts @ 6.30 AM
Supervisor	Dr Monen (TEL 987)

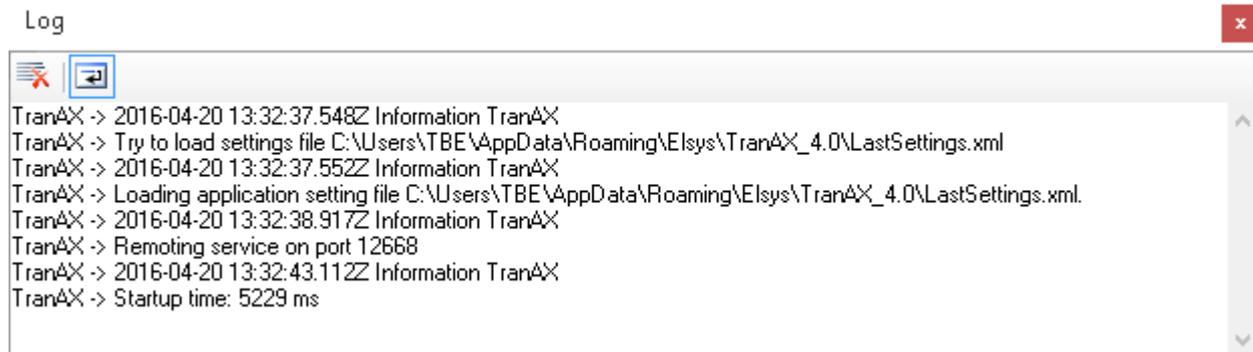
As with the [Record Log](#), text entries can be made and will be saved in the TPC5-signal files. Contrary to the record log no time stamps can be added. You may use the "**Attributes**" Window for Project names and descriptions, Participants or Measurement setup information.

31.3 Error Log

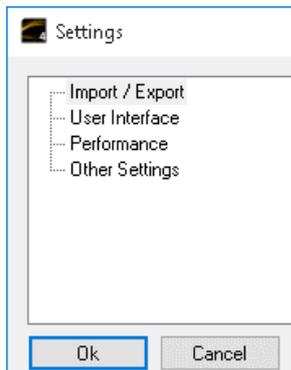
Click the icon "Error Log" in the Ribbon Tab "Layout", group "Misc. Controls" to open the Error Log window.



To open the Error log window, click menu "**View**" / "**Error Log**". In case TranAX is not working properly, the error log may give helpful information.

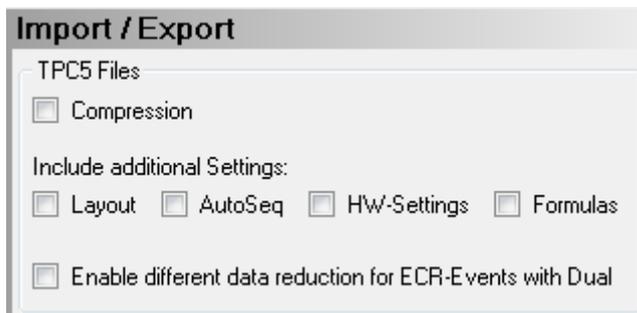


32 Settings



On the Startup Page and in the Ribbon Tab "Settings", group "Settings", Icon "Settings" you can open the dialog for general settings of TranAX.

32.1 Import/Export



These settings will be used as **default settings for other dialogs**.

The settings of the checkbox "compression" will also be used in the Save dialog.

The following settings can be done in the TPC5 Files section:

Compression.	This input control specifies the setting of the window Save Traces.
Include additional Settings	The TPC5-format allows saving besides the waveform data additional settings and configurations in the same file.
Layout	contains the information of the user interface
AutoSeq	contains the information of the Auto Sequence
HW-Settings	contains the hardware settings of the Control Panel
Formulas	contains the information of the Formula Editor
Enable different data reduction for ECR-Events with Dual	If the data acquisition mode was ECR with Dual then it can be enabled with this control to reduce the Dual data in a different way than the fast sampled events



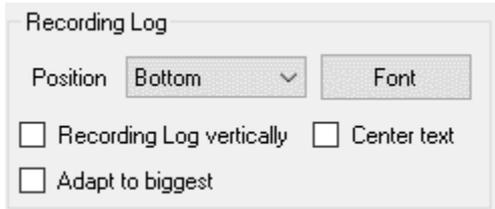
In ECR Mode the slower Dual trace can be captured along with the fast sampled events. The input control "**Different reduction for ECR-Events if Dual**" in the save TPC5 dialog is only visible if it was enabled in the settings under the Ribbon bar "**Settings**" / "**Settings**" / "**Import / Export**".

For more information about saving traces to a TPC5 file, see section [Saving](#).

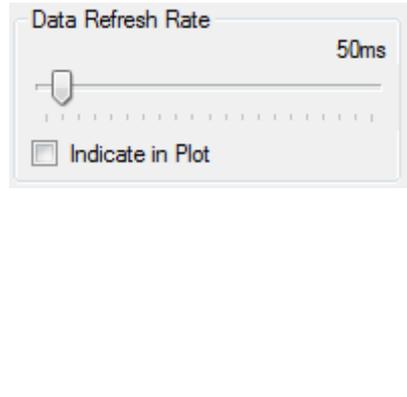
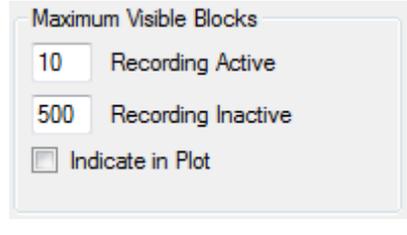
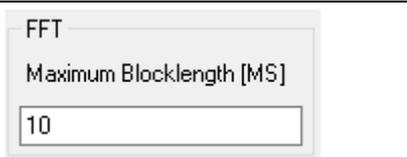
32.2 User Interface

In this dialog, the defaults for the user interface can be defined.

<p>Language</p> <p><input checked="" type="radio"/> English <input type="radio"/> Deutsch</p>	<p>The language of the GUI can be set to English or German. A restart of the application software is required in order to change the language.</p>
<p>Toolbar</p> <p><input type="checkbox"/> Use bigger Toolbar icons</p>	<p>The icons related to data acquisition can be displayed larger.</p>
<p>Mouse-Wheel</p> <p><input type="checkbox"/> Inverted Zooming</p> <p><input type="checkbox"/> Inverted Scrolling</p>	<p>The direction to zoom in or move a trace can be inverted.</p>
<p>Snapshot</p> <p><input type="radio"/> Bitmap <input type="checkbox"/> Enable Constant Size (Pixel)</p> <p><input checked="" type="radio"/> Vectorized (.emf) Width <input type="text" value="1280"/></p> <p><input type="checkbox"/> White Background Height <input type="text" value="500"/></p>	<p>The Waveform Display is stored to clipboard with a click on the Snapshot-icon  in the format as Bitmap or vectorized. Additionally, the size the display section should have then, can be chosen . The size of multiple windows in the page are proportionally adjusted. All text entries are stored in-tact. Attention should be given to the position of user specific text entries. Otherwise it may be placed awkwardly or entirely cut off.</p>
<p>Scalar Function Tables</p> <p><input type="checkbox"/> Colorful Channel Cells <input checked="" type="checkbox"/> Show Readout Markers On Trace</p>	<p>If "Colorful Channel Cell" is selected, the background of a Scalar Function Table cell has the same color as its Channel. Marker symbols can be attached to the traces, referencing particular scalar calculations. They are small circles for amplitude values (e.g. Maximum, Peak-Peak, etc.) or small squares for time values (e.g. RMS, Period, etc.).</p>
<p>Default Tab Window Placement</p> <p>Zoom <input type="text" value="Bottom"/></p> <p>XY <input type="text" value="Right"/></p> <p>Marker <input type="text" value="Bottom"/></p> <p>FFT <input type="text" value="Bottom"/></p> <p>Scalar Function Table A <input type="text" value="Right"/></p> <p>Scalar Function Table B <input type="text" value="Right"/></p> <p>Harmonics Table <input type="text" value="Right"/></p>	<p>Defines the position of the newly opened window.</p>
<p>Misc</p> <p>Grid Color <input type="color" value="black"/> Default Color <input type="checkbox"/> Minor Grid</p> <p>Back Color <input checked="" type="radio"/> Black <input type="radio"/> White</p> <p><input checked="" type="checkbox"/> Show cursor and trigger letters in statusbar</p> <p>Text / Plot Label / Indicators <input type="text" value="Font"/></p>	<p>Defines the grid Color and background color. The checkbox "Minor Grid" displays a finer grid. The checkbox "Show cursor..." shows cursor characters (A, B, C ...) at the top of the curve display or on the overhead status line.</p>

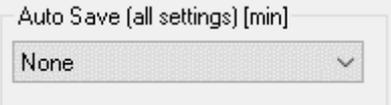
 <p>Recording Log</p> <p>Position <input type="text" value="Bottom"/> <input type="button" value="Font"/></p> <p><input type="checkbox"/> Recording Log vertically <input type="checkbox"/> Center text</p> <p><input type="checkbox"/> Adapt to biggest</p>	<p>Settings for the Recording Log.</p>
---	--

32.3 Performance

 <p>Data Refresh Rate</p> <p><input type="text" value="50ms"/></p> <p><input type="checkbox"/> Indicate in Plot</p>	<p>Defines with what time interval the traces in the waveform displays will be refreshed. TranAX uses this information to determine how the system performance should be shared between data visualization and data processing and what value fits the application optimally.</p> <p>If the system responds is slow, for example in Continuous Mode with many channels, the Data Refresh Rate should be set to a higher value.</p> <p>With the control Indicate in Plot displaying the Data Refresh Rate in the <i>Waveform Display</i> can be selected.</p>
 <p>Maximum Visible Blocks</p> <p><input type="text" value="10"/> Recording Active</p> <p><input type="text" value="500"/> Recording Inactive</p> <p><input type="checkbox"/> Indicate in Plot</p>	<p>Defines how many data blocks may be shown in the waveform display in active or inactive recording mode. This setting is related to the ECR and Block data acquisition modes.</p> <p>With the control Indicate in Plot, displaying the Maximum Visible Blocks in the waveform display can be selected.</p>
 <p>Crosshair position for Cursors</p> <p><input type="radio"/> Screen Data (Pixel)</p> <p><input checked="" type="radio"/> Raw Data (Hardware / File)</p>	<p>The cursor positions are either taken from the processed screen data or from the raw trace data. The selection Raw Data forces TranAX to request new data and may therefore slow down the system.</p>
 <p>FFT</p> <p>Maximum Blocklength [MS]</p> <p><input type="text" value="10"/></p>	<p>Limits the maximum block length for FFT analysis in order not to affect system performance unnecessarily.</p>

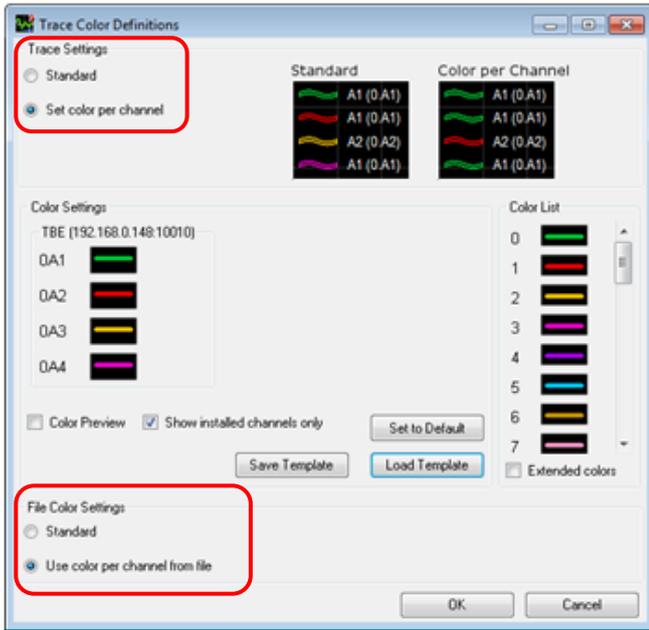
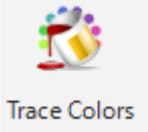
32.4 Other Settings

If a disruption of TranAX (power failure or otherwise) takes place, the system settings are not stored in Files. All earlier planned changes since the last active storing of settings are lost.

 <p>Auto Save (all settings) [min]</p> <p><input type="text" value="None"/></p>	<p>Ribbon tab "Settings", icon "Settings" under "Auto Save" a time interval can be set, after which the settings are stored automatically. The current Experiment Set will be overwritten.</p>
--	---

32.5 Trace Color Definitions (Menu "Extras")

"Trace Color Definitions" defines a **specific color for each hardware channel**. In a given Experiment, the channels have the same corresponding colors for every waveform. Click the icon "Trace Colors" in the Ribbon tab "Settings", group "Settings" to open this dialog:



To set your **own colors** for each channel, the option **"Trace Settings"** has to be set to "Set color per channel."

The group box "Color settings" allows you now to define the color for each channel. Either by clicking with the left mouse button on the channel color box and select from the Color Chooser or select a color by Drag & Drop from the **"Color List" on the right hand side**. If you are looking for more colors, check the item "Extended colors".

Once your definition is done, check the item **"Color preview"** to see how the colors will look like in TranAX. It is also possible to set colors for not installed channels, for example if you prepare a

default template which will be used on several systems. Uncheck item **"Show installed channels only"** to see all the other channels. Now you can define the channel colors for up to 8 Devices with up to 8 Modules.

The color definition can be saved as a **template** with a user specific filename. Click "Save Template" and enter a filename. Click "OK" to close the dialog and apply the settings. By closing the actual [Experiment](#), TranAX creates automatically a file called **"LastColor.ctf"**. When opening a new Experiment, the currently used color definitions will be **imported** to the new Experiment.

Trace Settings:

Standard	Every trace gets the color from TranAX. The color will be assigned in the order they are placed onto the screen (default).
Set color per channel	Every channel trace will have a user defined color.

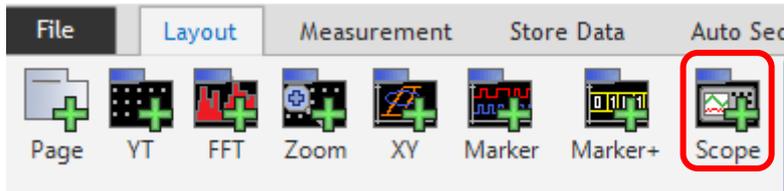
File Color Settings:

Standard	Every file gets its color from TranAX. The color will be assigned in the order they are placed onto the screen
Use color per channel from file	Traces from a TPC5 file will have their original colors as when the traces were saved to file.

33 SCOPE (Oscilloscope)

With SCOPE, instrument handling in straight forward applications has become much easier, as TranAX behaves like an oscilloscope that way. Although still without rotary knobs, the elements of seldom used operating modes are moved significantly to the background.

All manipulations, whether it concerns waveform curves and their windows, axes, annotations, etc. all behave as in a normal TranAX Y/T Waveform window.

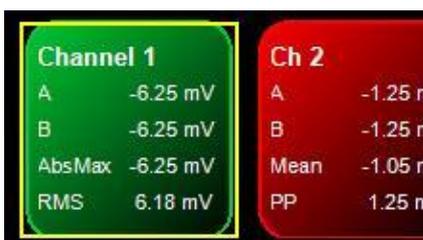


33.1 Channel settings

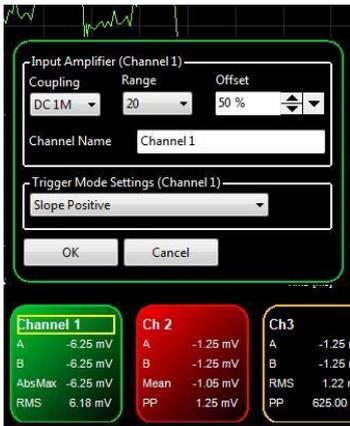
In the lower part of the display each hardware channel has its **operating box**.

Depending on the available hardware (channels) 4 to 8 such boxes are available (in systems with more than 8 channels only the first 8 will be shown).

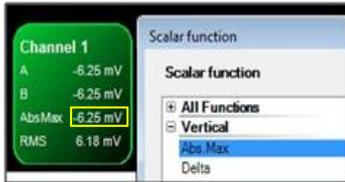
The operating box for the first channel (BNC A1) is at left. These boxes are typically labeled with the name of the channel.



A yellow square is drawn around the entire operating box when the mouse cursor is on the area. Left clicking the mouse activates or deactivates the channel. By activating the channel also, the curve will be shown.



When the mouse cursor is over a channel name, a yellow square is coming up. Left clicking opens a menu for setting up the most important channel parameters.



A yellow square also will come up, when the mouse cursor is placed over a scalar value. By left clicking the mouse a menu is coming up for selection and specifying of a scalar function.

33.2 Buttons for recording commands

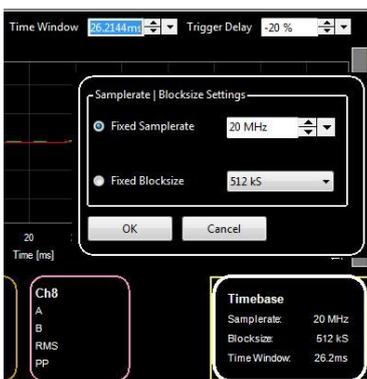
In the upper part of the display are the recording command buttons. As mentioned these emulate oscilloscope operations. More advanced recording modes (e.g. Multi-Block, ECR, etc.) are still possible but via alternative set-up procedures.



- Auto:** Multiple record mode with “Auto Trigger“ (free running oscilloscope).
- Normal:** Multiple record mode by waiting for a trigger, then finish record and start again for waiting on next trigger, etc.
- Single:** Starts record and waits for triggering, finishes record then stops. Needs re-arming for next record.
- Trigger :** A software trigger is sent to the hardware. A running recording can be finished orderly that way e.g. in the case of a missed signal trigger.
- Stop:** A running recording will be stopped. Then waiting. Normally such recorded signal cannot be used for further processing.

33.3 Time settings

Time range can be set with the **Time Window** parameter. After a recording its value usually will be equal to the full range of the X-axis (**button  down left**).



The time range is calculated as follows:

$$T = \text{Blocksize} * 1/\text{Samplerate}$$

The user has the choice which of the two values should be set as a constant.

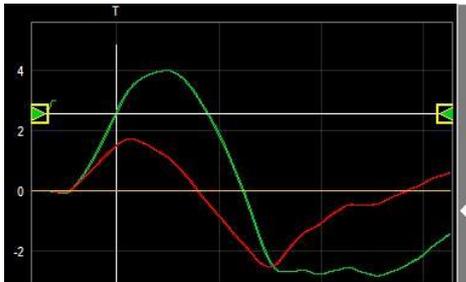
By clicking on the **Timebase** box (below right) a menu will come up. There a fixed sample rate or fixed block length can be chosen within their proprietary ranges. Then the other value will automatically be calculated in relation to the set **Time Window** parameter.

With **Trigger Delay** actual pre- or post-trigger values (-100% to +200%) can be set. Those settings however influence the range of the time axis. In any case also via  the full range of the X-axis can be set.

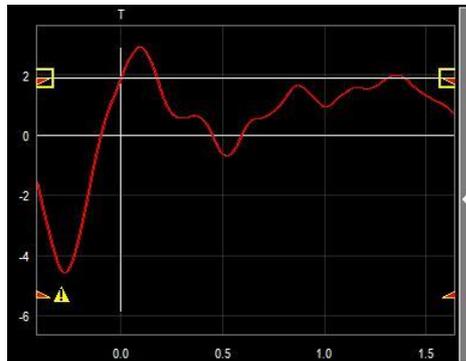
33.4 Trigger conditions, trigger level

Trigger conditions are also set via the channel settings menu. They are deliberately kept simple, i.e. only edge and window triggers can be set.

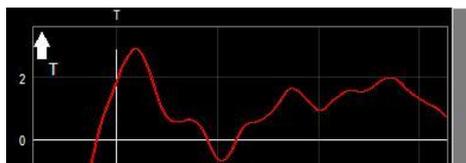
Other adjustments (incl. Trigger-option modes) if need can be set directly in the control panel.



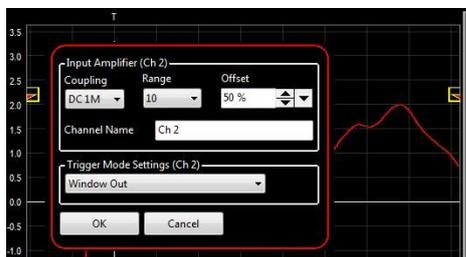
Trigger level is being set directly in the waveform window. For every active channel which trigger mode is not set to OFF, at the left and right side of the waveform window triangle symbols appear. They can be grabbed with the mouse and moved vertically. These symbols also show whether triggering will be on a positive or negative edge.



Window triggering is shown through two (four) half-triangles. Those also can be picked up by the mouse and shifted up or down. In case the level is set outside the vertical range of the signal, a warning triangle will appear. Its meaning is that on this channel no trigger can be generated.



In case the level is set above or below the display window (as a result of Y-zoom) white arrows appear, pointing to where the trigger point is. They can be picked up by the mouse and trigger point dragged into the display.



When these symbols are left clicked instead of grabbing, the menu for channel settings comes up (similar to clicking on channel names in the channel operating box).

33.5 Digital Readout Boxes

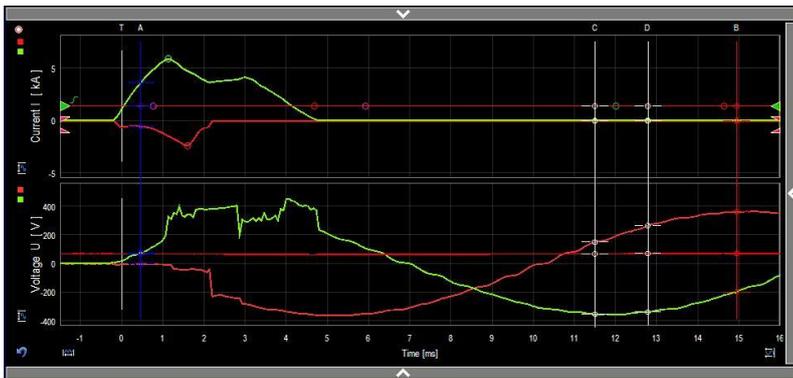


On the right hand side of the display several Readout Boxes may be shown. By clicking on the vertical bar at right, they will be switched on or off. The digital measurement values are obtained through scalar calculations. In principle all calculations as per the Scalar -Table B are possible. The values are labeled with abbreviated measurement/calculation results as well as channel names. If a calculation is carried out on a curve in a file, then also the name of the file will be blended in. Right clicking on the label overhead, opens the menu for selecting scalar calculations.

33.6 Maximum curve display



Left clicking on the top or bottom horizontal bar, suppresses operating tabs at the top as well as the channel operating fields below.



With that additional space for curve display is created.

Appendix

34 Group of functions in Formula Editor

34.1 Array Functions

ArrayMultidimensional(number*)	<p>Creates a multi-dimensional array, in most practical application a 2D or 3D Array.</p> <p>Example: <code>arr2D = ArrayMultidimensional(2, 2)</code> <code>arr2D(0, 0) = 0</code> <code>arr2D(0, 1) = 1</code> <code>arr2D(1, 0) = 1</code> <code>arr2D(1, 1) = 0</code></p>
Array(min to max) as dataType	<p>Creates an array of type dataType. **dataType can be double, string or complex. min and max define the size of the array. 0 to 3 results in four fields, 7 to 12 results in six fields.</p> <div data-bbox="646 734 1345 875" style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  An array can contain only the same type of data! </div> <p>Each individual value from the array can be accessed with brackets array(n). n must reside within the definition of the array of min and max.</p> <p>Example: <code>arr1 = Array(0 to 3) as double</code> <code>arr2 = Array(0 to 3) as string</code> <code>arr3 = Array(0 to 3) as complex</code> <code>arr4 = Array(5, 6, 7, 12, 56)</code> <code>arr5 = Array("U1", "I1", "U2", "I2", "°C")</code> <code>arr6 = Array(5+0j, 3+45j, 12+23j, 78+45j, 34+12j)</code></p> <pre> ; return first item number = arr1(0) ; set second item to "D" arr2(1) = "D" ; set first item to complex number "5+2j" arr3(0) = 5+2j </pre>
Array(min : max)	<p>Creates an Array with the length max - min + 1. All items in this array will be initialised with the value min + index + 1</p> <p>Example: <code>; arrExample = 1, 2, 3, 4, 5</code> <code>arrExample = Array(1:5)</code></p>
Complex	<p>Keyword for a complex number format a+bj. Normally used for assigning the data type to an Array</p> <div data-bbox="646 1928 1345 2069" style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  To define a complex number j or i is allowed </div> <p>Example:</p>

	<pre>arrComplex = Array(0 to 3) as complex arrComplex(0) = 2 + 2j arrComplex(1) = -1 + 3j arrComplex(2) = -10.6 - 4.55j arrComplex(3) = 4j ; = 1 + 5j complexRes1 = arrComplex(0) + arrComplex(1) ; 18.2 - 42.4j complexRes2 = arrComplex(2) * arrComplex(3)</pre>
ConvToArray(a)	<p>The function ConvToArray converts a to an array. If a is a curve time and trigger information gets lost.</p> <p>Example: sig = c0A1 arr = ConvToArray(sig)</p>
ConvToList(a)	<p>Converts a into a List.</p> <p>Example: ar\$ = Array("A", "B", "C") CrsList = ConvToList(ar\$)</p>
ConvToTrace(a, fs)	<p>The function ConvToTrace converts a into a curve. fs is the sampling rate of the new curve. 1/fs is the interval between each sample in the new curve. Trigger point is set at 0 s, the time stamp is the actual time the computer at the time of conversion.</p> <p>Example: sf = 1E3 ;1kHz arr1 = array(1,2,3,4,5,6,7,8,9,10) trace = ConvToTrace(arr1, sf)</p>
Copy(a)	<p>Formula Editor with Runtime 1.0 have always made copies of List, Array or Dictionary. With Runtime 2.0, these are passed as references. Changes in the copy also have an effect on the original. This is also the case for function calls with an array, list or dictionary as parameters. If you would like to create a copy explicitly, this can be done with copy. Parameter a is from type List, Array or Dictionary.</p> <p>Example: ; Select Runtime 2.0 arrOrig = Array(0:9) ; Creates an Array with entries 0 ... 9 arrRef = arrOrig ; Reference to arrCopy = Copy(arrOrig) ; Copy, no reference to arrOrig arrRef(4) = -5 ; Changes arrRef and arrOrig arrCopy(4) = 99 ; No impact to arrOrig</p>
Dictionary()	<p>Creates a dictionary which contains pair of values (key and value). Key can be used to read a specific value of a dictionary without knowing its exact position, like an array or a common list. Please see also function HasKey.</p> <p>Example:</p>

	<pre>dictT = Dictionary() dictT("voltage") = 230 dictT("current") = 10 voltage = NotDefined current = NotDefined key = "voltage" if HasKey(dictT, key) = 1 then voltage = dictT(key) endif key = "current" if HasKey(dictT, key) = 1 then current = dictT(key) endif</pre>
Double	<p>Keyword for a variable number format (double precision, 64-bit). Normally used at declaration of an Array.</p> <p>Example: arr1 = Array(0 to 3) as double</p>
Flip(a)	<p>Reverses a from the first value to the last one. Return value is the same as the input in a reversed order. This function can be used to compensate the time difference after filtering with the LowPass function.</p> <p>Example: trace = c0A1</p> <pre>; creates a time offset trace = LowPass(trace, Bessel, 6, 10E3) trace = Flip(trace) ; removes existing time offset trace = LowPass(trace, Bessel, 6, 10E3) trace = Flip(trace)</pre>
GetArrayDimensions(array)	<p>Returns the dimensions of an array. The return value variable is a number.</p> <p>Example:</p> <pre>; create two arrays arr = Array(0 to 9) as double arr3D = ArrayMultidimensional(4, 5, 7) ; read dimensions of these arrays dimArr = GetArrayDimensions(arr) ; 1 dimensional dimArr3D = GetArrayDimensions(arr3D) ; 3 dimensional</pre>
HasKey(dictionary, key)	<p>Checks for an existing entry key in dictionary. Return value is an integer: Exists = 1, not found = 0 Please see also the function Dictionary</p> <p>Example:</p> <pre>dictT = Dictionary() dictT("voltage") = 230 dictT("current") = 10 voltage = NotDefined current = NotDefined key = "voltage" if HasKey(dictT, key) = 1 then</pre>

	<pre> voltage = dictT(key) endif key = "current" if HasKey(dictT, key) = 1 then current = dictT(key) endif </pre>
Insert(a, index, value)	<p>Inserts a variable value at position index into the String or List a.</p> <p>Please note that index has to be \leq the length of a + 1.</p> <p>Example:</p> <pre> text = Insert("Voltage", 7, "s") ; text= "Voltages" </pre>
Length(a)	<p>Returns the size of parameter a.</p> <p>Example:</p> <pre> ; String str = "Test" LenStr = Length(str) ; 4 ; Array arr = Array(0 : 7) LenArr = Length(arr) ; 8 ; Multi dimensional array arr3D = ArrayMultidimensional(4, 5, 7) x = Length(arr3D, 0) ; 4 y = Length(arr3D, 1) ; 5 z = Length(arr3D, 2) ; 7 </pre>
List()	<p>List is a dynamic data structure. All types of variables are supported. Variable types may be mixed. List may be extended or shortened at run time.</p> <p>Example:</p> <pre> mylist = List() ; List declaration mylist() = 134 ; Append a new elements mylist() = "Text" mylist() = c0A1 ; some calculation on trace element. Result is ; appended to list mylist() = Smooth(mylist(2),20) tr1 = mylist(2) ; get original trace </pre>
Merge(a*)	<p>Merge combines multiple a into one. Every single a will sequentially be concatenated. The values remain unchanged. a must be of the same data type. Please see also the function MergeTraces.</p> <p>Example:</p> <pre> arr1 = array(1, 2, 3, 4) arr2 = array(5, 6, 7, 8) arr3 = array(9, 10, 11, 12) arr_new = Merge(arr1, arr2, arr3) ; arr_new contains now 12 elements. </pre>
Remove(a [, index [, length]]) Remove(a, key)	<p>Removes an element from the List, Dictionary or string a at the position index. The optional parameter length can be used to remove multiple entries.</p>

	<p>Removes the element key from the List or Dictionary</p> <p>a.</p> <p>Example:</p> <pre>; create an example dictT = Dictionary() dictT("test") = 10 dictT("voltage") = 20 ; remove one item Remove(dictT, "test")</pre>
String	<p>Keyword for a variable with text. Normally used at declaration of an Array There are no mathematical operations possible with strings. However, strings can be concatenated ("123" + "abc", results in "123abc").</p> <p>Example:</p> <pre>arr2 = Array(0 to 3) as string</pre>
ZeroPadding(arr, NrOfZeros)	<p>ZeroPadding adds additional fields with the value 0 to the end of an array.</p> <p>arr is an array, NrOfZeros a number for the additional fields.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;">  The existing fields will not changed, new fields are set to 0. </div> <p>Example:</p> <pre>; Array with 4 elements arr = array(1, 2, 3, 4) arr = ZeroPadding(arr, 10) ; Now there are 14 elements in arr</pre>

34.2 Auto Sequence Functions

AutoCalibration()	<p>Starts an autocalibration of all input channels.</p> <p>Example:</p> <pre>AutoCalibration() Pause("Autocalibration done!")</pre>
Beep()	<p>The PC gives a beep tone.</p> <p>Example:</p> <pre>for a = 0 to 10 step 1.0 delay(1) Beep() next</pre>
Call(cmd, args [, timeoutMs [, showWindow [, attachToProcess]])	<p>This command starts another program. That can be a *.exe or a *.bat file.</p> <p>timeoutMs indicates how long should be waited for the called-up program to finish. When timeoutMs is missing, processing immediately is continued with the next formula-command. The called-up program then runs as a new process parallel to the application. When timeoutMs larger than zero is specified, then Auto-sequence waits maximum so many milliseconds for the program to finish. If the program is being stopped on</p>

	<p>beforehand, immediately processing will resume with the next formula command. Is timeoutMs negative e.g. -1, the program will be finishing without any time restrictions.</p> <p>cmd: The command which will be executed by the system.</p> <p>args: Additional arguments for cmd</p> <p>timeoutMs: Time to wait for the called process (>0)</p> <p>showWindow: Shows or hides the window</p> <p>attachToProcess: Attaches itself to the application process. If application will be closed this process will end as well.</p> <p>Example:</p> <pre>; Example executes a Python Script and waits until finished p = GetPath("data") ; Reference to "data" folder in Experiment pythonpath = "C:\Python\Python36\python.exe" arg = p + "\scan_sga.py" Call(pythonpath, arg, -1)</pre>
DisableExternalStart()	Prevents data acquisition via an external signal at the Start Record input (Pin 3 on the 25 pole connector).
EnableExternalStart()	Initiates data acquisition via an external signal at the Start Record input (Pin 3 on the 25 pole connector).
GetTriggerStatus([clusterNr])	Returns a number which represents the trigger status of the measurement system. Optional parameter clusterNr can be used to specify an individual cluster, if multiples are defined. Return values are: Disarmed = 0, Armed = 1, Triggered = 2
IsRecording()	<p>Checks if a measurement is active. The return value variable is of type Boolean, Measurement is running = True, otherwise = False.</p> <p>Example:</p> <pre>if IsRecording()=1 then ; do some debug stuff else pause("Not recording...") endif</pre>
LoadLayout(filepath)	Uploads the layout file filepath .
LoadSettings(filepath)	Uploads the settings file filepath in the control panel.
Pause([text [, variables])	<p>Pauses the formula, the user must confirm the continuation.</p> <p>This command can be used to send messages such as "Please Enable DUT" or allows the user to view a completed measurement before proceeding with the auto sequence.</p> <p>Parameter text is a text, with variables* several variables can be passed and then adjusted in the Pause</p>

	<p>dialog. Return value variable is an Integer with the following values: 1 = Abort 3 = Continue</p> <p>Example: items = 20 text = "This is a text." temperature = 23.5</p> <pre>Pause("Please check!") val = Pause("Parameters", items, temperature, text)</pre>
Print(name)	<p>Prints the Waveform Display name (the curves displayed, are automatically updated after each measurement). Printing is carried out as specified in Print preview.</p> <p>Example: ; print "Waveform 1" to the default printer Print("Page 1") ; print "Page 1" to the default printer Print("Page 1")</p>
QuitApplication()	<p>The main application (TranAX or BallAX) will be stopped and terminated.</p>
SaveSpectrum(filepath, name, channels)	<p>Stores the spectrum of a FFT-Waveform in file filepath (usually in the directory data of the current experiment. name designates the FFT-Spectrum-Windows. All channels from which spectra must be stored in the file. Channels from which no spectrum is calculated (i.e. could not be calculated) are not stored away. Calculated spectra are stored precisely in the same way as shown in the spectrum window, corresponding to time window limitations of the original time domain signal. Also the FFT weighting window cannot be changed afterwards. Only spectra from hardware channels (e.g. 0A1) are accepted.</p> <p>With regards to filepath the same is valid as described under File. When there is a "#" at the end of a file name, it will be replaced by an ascending number.</p> <p>Example: SaveSpectrum("spectrum.tps5", "Spectrum 1", "0A1-4, 0B1, 0B3")</p>
StartRecording()	<p>Starts a recording. The necessary settings must first be made in the Control Panel.</p> <p>Please see also the functions Trigger, WaitonEOR and StopRecording.</p> <p>Example: ; stop active measurements StopRecording()</p>

	<pre> ; start, trigger and wait until recording has stopped StartRecording() Delay(0.5) Trigger() WaitOnEOR() ; save traces Save("myFilename#.tpc5", c0A1-c0A4) </pre>
StopRecording()	<p>Cancels an ongoing recording.</p> <p>Example:</p> <pre> ; stop active measurements StopRecording() ; start, trigger and wait until recording has stopped StartRecording() Delay(0.5) Trigger() WaitOnEOR() ; save traces Save("myFilename#.tpc5", c0A1-c0A4) </pre>
StorePage(filepath, name)	<p>Stores the Page name in file filepath (usually in the directory data of the current experiment. A page represents a whole tab which can include waveform displays, scalar tables, etc.</p> <p>With regards to filepath the same is valid as described under File. When there is a "#" at the end of a file name, it will be replaced by an ascending number.</p> <p>Example:</p> <pre> StorePage("mypage#.tdp", "Page 1") </pre>
StoreSnapshot(filepath, name)	<p>Stores the Waveform Window name in the filepath file. Considers the settings under Extras / Settings / User space.</p> <p>With regards to filepath the same is valid as described under File. When there is a "#" at the end of a file name, it will be replaced by an ascending number.</p>
Trigger()	<p>Generates a manual trigger (in case that not already happened in hardware by the signal itself).</p> <p>Example:</p> <pre> ; stop active measurements StopRecording() ; start, trigger and wait until recording has stopped StartRecording() Delay(0.5) Trigger() WaitOnEOR() ; save traces Save("myFilename#.tpc5", c0A1-c0A4) </pre>
WaitForCalculations()	<p>The operation WaitForCalculations waits until the calculations of Scalar and/or Harmonics Tables are completed.</p>
WaitForData([timeoutSeconds])	<p>Wait until minimal one valid block is recorded. With the Parameter timeoutSeconds a maximum</p>

	<p>waiting time can be given (in seconds). Is the parameter negative or in case of failure, waiting time is indefinite. In Continuous-Mode there is no waiting. The same is true when in ECR-Mode the Dual-Recording mode is activated. Then the function WaitOnEOR should be used.</p> <p>Example: <code>val = WaitForData()</code></p>
WaitOnEOR()	<p>Waits with further processing until recording has ended (End Of Record). If Single Shot is not selected on the control panel, the recording must be stopped manually (or by the software command Stop Recording) in order to fulfill the EOR-Status.</p> <p>Example:</p> <pre> ; stop active measurements StopRecording() ; start, trigger and wait until recording has stopped StartRecording() Delay(1) Trigger() WaitOnEOR() ; Data available? if Length(c0A1) > 0 then Save("myFilename#.tpc5", c0A1, c0A2, c0A3, c0A4) endif </pre>
WatOnEOR()	<p>Waits with further processing until recording has ended (End Of Record). If Single Shot is not selected on the control panel, the recording must be stopped manually (or by the software command Stop Recording) in order to fulfill the EOR-Status.</p>

34.3 Base Functions

a - b	<p>Subtraction of the parameters a and b. The returned value variable is generally of the same type as the paramter a.</p> <p>Please note:</p> <ul style="list-style-type: none"> • Strings cannot be subtracted. • The subtraction of a curve with a number returns a corresponding offset of the curve. <p>Example:</p> <pre> ; Numbers n1 = 5-3 ; n1=2 ; Complex numbers n2 = -3+5j n3 = 5-10j n4 = n2 - n3 ; n2=-8+10j </pre>
--------------	---

	<pre> n5 = n2 - n1 ; n2=-5+5j ; Strings cannot be subtracted! ; Curves tr1[V] = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude (20Vpp), 50Hz tr2[V] = Sinus(8, 30, 0, tr1) ; 8V amplitude (16Vpp), 30Hz tr3[V] = tr1 - tr2 ; tr3 is a superposition of tr1 and tr2 tr4[V] = tr3 - 5 ; tr4 with an offset of 5. The unit of the offset is the same as the physical unit of curves ; Arrays ; arr1 is an array of type double. arr1 = array(0, 2, 5, 7) arr2 = array(8, 7, -6, -5) arr3 = arr1 - 1 ; arr3=(-1, 1, 4, 6) arr4 = arr1 - arr2 ; arr4=(-8, -5, 11, 12) </pre>
a % b	<p>Calculates the Modulo operation (Euclidean division), the division of the two parameters a and b. which produces a quotient and the integer remainder as result of variable.</p> <p>The returned value variable is generally of the same type as the paramter a.</p> <p>Example:</p> <pre> ; Numbers n1 = 5 % 3 ; n1=2 n2 = 10.3 % 2.1 ; n2=1.9 ; Curves tr1[V] = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude (20Vpp), 50Hz tr2[V] = Sinus(8, 30, 0, tr1) ; 8V amplitude (16Vpp), 30Hz tr3 = tr1 % tr2 ; sample for sample modulo of the two traces tr4 = tr1 % 6 ; cuts off all values higher than 6 ; Strings cannot be used with modulo! ; Arrays ; arr1 is an array of type double. arr1 = array(0, 2, 5, 7) arr2 = array(8, 7, -6, -5) arr3 = arr1 % 4 ; arr3=(0, 2, 1, 3) arr4 = arr1 % arr2 ; arr4=(0, 2, 5, 2) ; Modulo by 2 can also be used to determinate a even or odd number for a = 0 to 4 step 1.0 str = "The number " + a + " is " if a % 2 = 0 then str += "even" else str += "odd" endif Pause(str) next </pre>
a * b	<p>Multiplication of the parameters a and b. The returned value variable is generally of the same type as the paramter a.</p>

	<p>Please note:</p> <ul style="list-style-type: none"> • Strings cannot be multiplied. • The multiplication of a curve with a number returns a curve with corresponding scaling. <p>Example:</p> <pre> ; Numbers n1 = 5*3 ; n1=15 ; Complex numbers n2 = -3+5j n3 = 5-10j n4 = n2 * n3 ; n2=35+55j n5 = n2 * n1 ; n2=-45+75j ; Strings cannot be multiplied! ; Curves tr1[V] = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude (20Vpp), 50Hz tr2[V] = Sinus(8, 30, 0, tr1) ; 8V amplitude (16Vpp), 30Hz tr3[V] = tr1 * tr2 ; tr3 is a modulation of tr1 and tr2 tr4[V] = tr3 * 5 ; tr4 with scaling of factor 5 ; Arrays ; arr1 is an array of type double. arr1 = array(0, 2, 5, 7) arr2 = array(8, 7, -6, -5) arr3 = arr1 * 3 ; arr3=(0, 6, 15, 21) arr4 = arr1 * arr2 ; arr4=(0, 14, -30, -35) </pre>
<p>a / b</p>	<p>Division of the parameters divisor a and dividend b. quotient = divisor / dividend The returned value quotient variable is generally of the same type (not the same unit) as the paramter a.</p> <p>Please note:</p> <ul style="list-style-type: none"> • Strings cannot be divided • The division of a curve with a number returns a curve with corresponding scaling. <p>Example:</p> <pre> ; Numbers n1 = 5/3 ; n1=1.666... ; Complex numbers n2 = -3+5j n3 = 5-10j n4 = n2 / n3 ; n2=-0.52-0.04j n5 = n2 / 5 ; n2=-0.6+1j ; Strings cannot be divided! ; Curves tr1[V] = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude (20Vpp), 50Hz tr2[V] = Sinus(8, 30, 0, tr1) ; 8V amplitude (16Vpp), 30Hz tr3[V] = tr1 / tr2 ; tr3 is a division of tr1 with tr2 </pre>

	<pre>tr4[V] = tr1 / 5 ; tr1 with scaling of factor 1/5 ; Arrays ; arr1 is an array of type double. arr1 = array(0, 2, 5, 7) arr2 = array(8, 7, -6, -5) arr3 = arr1 / 3 ; arr3=(0, 0.66667, 1.6667, 2.3333) arr4 = arr1 / arr2 ; arr4=(0, 0.28571, -0.83333, -1.4)</pre>
<p>a^b</p>	<p>Exponentiation of the parameters base a and exponent b. exponentiation = base ^ exponent The returned value exponentiation variable is generally of the same type as the paramter a.</p> <p>Please note:</p> <ul style="list-style-type: none"> • Strings cannot be exponentiated • Comlex numbers cannot be used as an exponent • The exponentiattion of a curve with a number returns a curve with exponential scaling. • This function returns a useful result, if the parameters comply with the rules of exponentiation. E.g. $-3 ^ 0.5$ is not defined for real numbers. <p>Example:</p> <pre>; Numbers n1 = 5^3 ; n1 = 125 n2 = (-3) ^ 0.5 ; n2 = 0+1.7321j n3 = 64 ^ (1/2) ; n3 = 8 (scuare root) n4 = 64 ^ (1/3) ; n3 = 4 (cube root) ; Complex numbers n5 = (-3+5j)^2 ; n5 = -16 -30j ; Strings cannot be exponentiated! ; Curves tr1[V] = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude (20Vpp), 50Hz tr2[V] = Sinus(8, 30, 0, tr1) ; 8V amplitude (16Vpp), 30Hz tr3[V] = tr1 ^ tr2 ; tr3 ist tr1 exponentiated with tr2 tr4[V] = tr1 ^ 5 ; tr1 ist tr2 with exponential scaling ; Arrays ; arr1 is an array of type double. arr1 = array(0, 2, 5, 7) arr2 = array(8, 7, -6, -5) arr3 = arr1 ^ 3 ; arr3=(0, 8, 125, 343) arr4 = arr1 ^ arr2 ; arr4=(0, 128, 6.4E-05, 5.9499E-05)</pre>
<p>a + b</p>	<p>Addition of the parameters a and b. The returned value variable is generally of the same type as the paramter a.</p> <p>Please note:</p> <ul style="list-style-type: none"> • Strings can be concatenated. • The addition of a curve with a number returns a curve with corresponding offset.

	<ul style="list-style-type: none"> Concatenation of a string and a number returns a string, wherein the number is interpreted as a string. <p>Example:</p> <pre> ; Numbers n1 = 5+3 ; n1=8 ; Complex numbers n2 = -3+5j n3 = 5-10j n4 = n2 + n3 ; n2=5-5j n5 = n2 + n1 ; n2=2+5j ; Strings s1 = "c0A"+1 ; s1="c0A1" of type string s2 = "c0A"+(1+1) ; s2="c0A2" of type string s3 = "c0A"+1+1 ; s3="c0A11" of type string ; Curves tr1[V] = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude (20Vpp), 50Hz tr2[V] = Sinus(8, 30, 0, tr1) ; 8V amplitude (16Vpp), 30Hz tr3[V] = tr1 + tr2 ; tr3 is a superposition of tr1 and tr2 tr4[V] = tr3 + 5 ; tr4 with an offset of 5. The unit of the offset is the same as the physical unit of curves ; Arrays ; arr1 is an array of type double. arr1 = array(0, 2, 5, 7) arr2 = array(8, 7, -6, -5) arr3 = arr1 + 1 ; arr3=(1, 3, 6, 8) arr4 = arr1 + arr2 ; arr4=(8, 9, -1, 2) </pre>
Abs(a)	<p>Returns the absolute (positive) value of a. The result remains as the same type of variable a.</p> <p>Can also be used in combinatin with complex numbers, see also the functions Real, Imag and Angle.</p> <p>Example:</p> <pre> ; Numbers n = -5 nAbs = Abs(n) ; nAbs = 5 ; Complex numbers c = -1j cAbs = Abs(c) ; cAbs = 1 ; Signals: Sine wave, 50Hz, 10Vpp tr = Sinus(10,50,0,10E3,1E3) trAbs = Abs(tr) ; Pulsating DC voltage, 100Hz </pre>
Diff(a)	<p>Calculates the Derivative the trace a. The result variable stays as a measurement curve after calculation.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  <p>The usage of function Diff in combination with high frequency noise could lead to unsatisfied results. Using filtering and smoothing reduces this effect. Please see also the functions Lowpass and Smooth.</p> </div> <p>Example:</p>

	<pre> ; create a signal v[m/s] with 10[m/s^2] acceleration v[m/s] = Ramp(10,1E3,10E3) a[m/s^2] = diff(v) </pre>
<p>GetCrs(name [, arealdx], cursor)</p>	<p>Returns the time position (X-axis) in seconds or the level in the scaled physical unit (Y-axis) of cursor.</p> <p>name: Valid Waveform (window name) as String (for example, "Waveform 1"). cursor: Active and valid cursor as a string (for example, "A" or "AH").</p> <p>For multiple areas in one waveform, the optional parameter arealdx from type integer can be used. This starts at 0, so the second area = 1.</p> <div data-bbox="678 651 1460 846" style="border: 1px solid gray; padding: 5px;">  <p>Horizontal cursors representing the level and have the extension "H". Therefore cursor "A" reads the time information, with cursor "AH" the level. Horizontal cursors must be explicitly activated in the waveform.</p> </div> <p>Example:</p> <pre> ; read from a waveform or first area in a waveform t1 = GetCrs("Waveform 1","A") ; position of Cursor A l1 = GetCrs("Waveform 1", "AH") ; level of Cursor A Horizontal ; read from second area in a Waveform t2 = GetCrs("Waveform 1", 1, "B") ; position of Cursor B l2 = GetCrs("Waveform 1", 1, "BH") ; level of Cursor B Horizontal </pre>
<p>GetFFTTimeWindow(name)</p>	<p>Returns the positions of the left and right time markers in the YT Waveform. The parameter name is of type String, the return value variable is a two-dimensional array of type Double. This contains the time information of the left and right time marker.</p> <div data-bbox="678 1377 1460 1536" style="border: 1px solid gray; padding: 5px;">  <p>Depending on how the time markers are moved, the first or the second number may represent the left or the right position. If necessary, these two times must be sorted for further processing.</p> </div> <p>Please see also function SetFFTTimeWindow.</p> <p>Example:</p> <pre> timeMarker = GetFFTTimeWindow("Waveform 1") t1 = timeMarker(0) t2 = timeMarker(1) ; make sure t1 is always smaller than t2 if t1 > t2 then tmp = t1 t1 = t2 t2 = tmp endif Pause ("Position of Time Window Indicator", t1, t2) </pre>

Int(a)	<p>Calculates the integral of the curve a. The result variable remains as a measurement curve after calculation.</p> <p>Please see also the function Area for numeric results.</p> <p>Example:</p> <pre> ; create a signal v[m/s] with 10[m/s^2] acceleration v[m/s] = Ramp(10,1E3,10E3) a[m/s^2] = diff(v) s_trace = Int(v) ; creates a trace s_num = Area(v) ; numeric result </pre>
Integer(a)	<p>Rounds the value of the parameter a to the nearest lower integer (up to 15 decimal places).</p> <p>The returned value variable is of the same type as the paramter a. This function is needed especially for rounds of single values.</p> <p>Example:</p> <pre> ; Previous calculation ; several periods of a signal Phas[°]=682.5 ; Phase should be +/-180°; Phas[°]=360*(Phas/360 - Integer(Phas/360+0.5)) ; Phas = -37.5°; n1 = Integer(0.5) ; n1=0 n2 = Integer(0.9) ; n2=0 ; round according to rules for a = 0.4 to 1.6 step 0.1 valRound = Integer(a + 0.5) s = "Value "+a+" is rounded "+valRound Pause(s) next </pre>
Limit(a, min, max)	<p>Limits a within a min to max range.</p> <p>The returned value variable is of the same type as the paramter a.</p> <p>This function can be used to avoid a possible division by zero (or almost zero, in curves with zero crossings).</p> <p>Example:</p> <pre> ; trace operation tr = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude -> 20Vpp trLimit = Limit(tr,-5, 8) ; cut off signal ; array operation arr = Array(0:20) ;create an array with the entries from 0 tp 20 len = Length(arr) ; len = 21 for a = 0 to len-1 step 1.0 arr(a) = Limit(arr(a), 2, 10) ; limit values of the array next </pre>

MaxOf(a, b)	<p>This function returns the larger value of the two parameters a and b.</p> <p>Example: val1 = -12.3 val2 = 20</p> <pre>nlarger = MaxOf(val1, val2) ; val = 20 nsmaller = MinOf(val1, val2) ; val = -12.3</pre>
MinOf(a, b)	<p>This function returns the smaller value of the two parameters a and b.</p> <p>Example: val1 = -12.3 val2 = 20</p> <pre>nlarger = MaxOf(val1, val2) ; val = 20 nsmaller = MinOf(val1, val2) ; val = -12.3</pre>
SetCrs(name [, arealdx], cursor, position)	<p>Places the cursor cursor on the position position, on a time value in seconds, or on the level in the scaled physical unit.</p> <p>name is a valid Waveform (window name) as string (for example, "Waveform 1"). cursor is an active and valid cursor as a string (for example, "A" or "AH").</p> <p>For multiple areas in one waveform, the parameter areaIndex can be used as an integer. This starts with 0, means the second area would be =1.\</p> <p>Horizontal cursors representing the level have the extension "H". Therefore cursor "A" sets the time information, with cursor "AH" the level. Horizontal cursors must be explicitly activated in the waveform.</p> <p>Example:</p> <pre>; set in a waveform or first area in a waveform SetCrs("Waveform 1", "A", 0.1) ; position of Cursor A SetCrs("Waveform 1", "AH", 4.5) ; level of Cursor A Horizontal ; set in the second area in a Waveform SetCrs("Waveform 1", 1, "B", -0.1) ; position of Cursor B SetCrs("Waveform 1", 1, "BH", -2.1) ; level of Cursor B Horizontal</pre>
SetFFTTimeWindow(name, start, end)	<p>Sets the positions of the left and right time markers in the YT Waveform.</p> <p>The parameter name is of type String, start and stop are of type Double. These are each the time positions of the two time markers.</p> <p>Please see also the function GetFFTTimeWindow.</p> <p>Example: tLeft = -1E-3 ; -1ms tRight = 1E-3 ; 1ms</p> <pre>SetFFTTimeWindow("Waveform 1", tLeft, tRight)</pre>

<p>SetRange(a, b) SetRange(a, min, max)</p>	<p>Sets the y-axis range of a with the axis settings from b. This can be used to create curve diagrams with the same y-axis range.</p> <div data-bbox="563 286 1345 488" style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  <p>Every calculation which includes a curve changes the y-axis range. That's why SetRange is a good function if you want to have the same y-axis range in the waveform to compare curves. It does not affect any samples of the curve.</p> </div> <p>Return value variable is the modified parameter a with the new y-axis range.</p> <p>Sets the y-axis range of a with the parameters min and max. This can be used to create curve diagrams with the same y-axis range.</p> <div data-bbox="563 790 1345 992" style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  <p>Every calculation which includes a curve changes the y-axis range. That's why SetRange is a good function if you want to have the same y-axis range in the waveform to compare curves. It does not affect any samples of the curve.</p> </div> <p>Return value variable is the modified parameter a with the new y-axis range.</p> <p>Example:</p> <pre>trOrig1 = Sinus(10,50,0,1E3,1E3) ; sine wave signal, 20Vpp trOrig2 = trOrig1 * 0.75 ; less amplitude trNew1 = SetRange(trOrig1, -15, 15) ; set range to +/-15V trNew2 = SetRange(trOrig2, trNew1) ; copy range from previous trace</pre>
<p>Sqrt(a)</p>	<p>Calculates the square root of a.</p> <p>Example:</p> <pre>n1 = Sqrt(9) ; n1= 3 n2 = Sqrt(-9) ; n2= 0+3j ; alternatively, Sqrt(a) = a^(1/2) n3 = 9 ^ 0.5</pre>

34.4 Channels

<p>EnChannels()</p>	<p>In combination with for each loop, enChannels can be used the read-out and analyse data form each channel from the selected devices. It will get all the available channels in the measurement system.</p> <div data-bbox="592 1944 1345 2085" style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  <p>Please note disabled channels will be counted too.</p> </div>
----------------------------	---

	<p>Example:</p> <pre>for each input in EnChannels() maxVal = Max(input) minVal = Min(input) next</pre>
<p>GetChannel(deviceIdx, boardIdx, inputIdx, blockNr) GetChannel(name, blockNr)</p>	<p>Returns the measured block blockNr of the corresponding channel which is specified with the parameters deviceIdx, boardIdx and inputIdx.</p> <div style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p> The parameter inputIdx starts at 1, deviceIdx and boardIdx start at 0.</p> </div> <div style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p> Check with the function NBlks() how many blocks are available.</p> </div> <p>Returns the measured block blockNr of the corresponding channel which is specified with the parameter name. name is the actual input name which can be user defined.</p> <div style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p> Check with the function NBlks() how many blocks are available.</p> </div> <p>Example:</p> <pre>Blk = 0 value = GetChannel(0, 0, 1, Blk) ; value = Device 0, c0A1, Blocks 0</pre>
<p>GetChannelDesc(trace)</p>	<p>Returns an Array with three entries (length = 3) which contains, device, board and channel. The parameter trace defines a hardware channel, not a trace from a file!</p> <div style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p> Please note Device and Board starts at 0 Channel starts at 1.</p> </div> <p>Example:</p> <pre>for each input in enChannels() desc = GetChannelDesc(input) range = GetRecordingPar(desc(0), desc(1), desc(2), 23) next</pre>
<p>GetChannelRanges(deviceIdx, boardIdx)</p>	<p>Returns the available measurement ranges of the board boardIdx in the device deviceIdx.</p> <p>Example:</p> <pre>ranges = GetChannelRanges(0, 0); 1. Device, 1. Board, ranges= 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100</pre>
<p>GetChName(a)</p>	<p>Provides the name of a. This can be a hardware channel or a calculated trace.</p> <p>Example:</p> <pre>trHw = c0A1 name1 = GetChName(c0A1) ; hardware channel name name2 = GetChName(trHw) ; hardware channel name</pre>

	<pre>trCalc = Sinus(10,50,0,1E3,1E3) name3 = GetChName(trCalc) ; variable name</pre>
<p>GetTrace (a[, index[, blockNr[, markerNr]]])</p>	<p>Returns a trace from any available source in TranAX. The parameter a can be one of the following types:</p> <ul style="list-style-type: none"> • Hardware channel (e.g. c0A1) • Reference (e.g. ref1) • File (e.g. "myFile.tpc5") • Averaged time signal (e.g. "@0A1") • Averaged FFT signal (e.g. "%0A1") <p>The optional parameter index defines the channel for reading from a file, in any other case set this to 0.</p> <p>The optional parameter blockNr defines the block number beginning with 0. Please note that in ECR recordings the dual recording part (lower sampling rate) is block 0 when active.\</p> <p>The final optional parameter markerNr is either 1 or 2, in case of 14 bit recording mode the upper 2 bits can be used for external digital signals, called marker.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Please note Marker is a hardware option which has to be installed on your TraNET device. </div> <p>Example:</p> <pre>;Pause("Start Multi Block recording with 5 blocks. Click continue when done.") ; - Analyse hardware channel ; - Data from reference ; - Data from saved file ; Save channel A1 to a file Save("myTrace.tpc5", "0A1") ; save all blocks ; Set channel A1 to Reference 1 SetReference(ref1,true, c0A1) ; true means references to the hardware channel ; 1. example ; read one block from the Multi Block recorded trace ; read 1st trace, 3rd block (begins with 0) tr1 = GetTrace(c0A1, 0, 2) tr2 = GetTrace("myTrace.tpc5", 0, 2) tr3 = GetTrace(ref1, 0, 2) ; 2. example ; use for each loop n1 = NBlks(c0A1) n2 = NBlks(ref1) n3 = NBlks(File("myTrace.tpc5",0)) cnt1 = 0 cnt2 = 0 cnt3 = 0 minVal1 = List() minVal2 = List() minVal3 = List() get ; Hardware channel</pre>

	<pre> for a = 0 to n1-1 step 1.0 tr1 = GetTrace(c0A1, 0, a) minVal1() = Min(tr1) next ; Reference for a = 0 to n1-1 step 1.0 tr2 = GetTrace(ref1, 0, a) minVal2() = Min(tr2) next ; From File for a = 0 to n1-1 step 1.0 tr3 = GetTrace("myTrace.tpc5", 0, a) minVal3() = Min(tr3) next </pre>
hardwarechannel	Represents a designator of a hardware channel .

34.5 Enumerable Functions

<p>EnBlocks(a, waitForBlocks) EnBlocks(a, start, end) EnBlocks(deviceldx, boardIdx, inputIdx, waitForBlocks) EnBlocks(deviceldx, boardIdx, inputIdx, start, end)</p>	<p>Function enBlocks returns in combination with the for each loop function all recorded blocks of a. The parameter waitForBlocks defines if the formula waits until new blocks are recorded.</p> <p>Function enBlocks returns in combination with the for each loop function all recorded blocks of a. The parameters start and end define which blocks to read from a.</p> <p>Function enBlocks returns in combination with the for each loop function all recorded blocks of the specified input which is defined with the parameters deviceldx, boardIdx and inputIdx. The parameter waitForBlocks defines if the formula waits until new blocks are recorded.</p> <div data-bbox="667 1310 1460 1400" style="background-color: #f0f0f0; padding: 5px;">  Note: deviceldx and boardIdx start at 0, inputIdx at 1. </div> <p>Function enBlocks returns in combination with the for each loop function all recorded blocks of the specified input which is defined with the parameters deviceldx, boardIdx and inputIdx. The parameters start and end define which blocks to read from a.</p> <div data-bbox="667 1713 1460 1803" style="background-color: #f0f0f0; padding: 5px;">  Note: deviceldx and boardIdx start at 0, inputIdx at 1. </div> <p>Example:</p> <pre> ;multi block recording trace = c0A1 minVal = List() for each item in enBlocks(trace,false) ; write min of each block into a list minVal() = Min(item) next </pre>
---	--

<p>EnChannels()</p>	<p>In combination with for each loop, enChannels can be used the read-out and analyse data form each channel from the selected devices. It will get all the available channels in the measurement system.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  Please note disabled channels will be counted too. </div> <p>Example:</p> <pre>for each input in EnChannels() maxVal = Max(input) minVal = Min(input) next</pre>
<p>EnEvents(a, type, start, end, level, hysteresis)</p>	<p>Function enEvents is comparable the function FindEvent, but it is used in combination with a for each loop. All events in a can be found between start and end, both are time marks. The parameter type defines the type of slope. Other parameters are level and hysteresis. These are used to define and find the type of slope and level. Return value variable is then the position as time value in seconds.</p> <p>Example:</p> <pre>trace = c0A1 ; Slopes: 0=positive, 1=negative, 2=both type = 0 t1 = TBegin(trace) t2 = tEnd(Trace) tEvents=list() for each item in enEvents(trace, type, t1, t2, 0, 0.1) ; list with time stamps tEvents() = item next</pre>
<p>EnSlices(a, size [, doUntilEOR]) EnSlices(a, size, start, end)</p>	<p>The function enSlices provides in combination with the function for each loop single slices of a. The parameter size defines the number of samples per slice. The optional parameter doUntilEOR defines if the function is waiting until the entire recording is done. This can be useful during a continuous measurement.</p> <p>The function enSlices provides in combination with the function for each loop single slices of a. The parameter size defines the number of samples per slice. The parameters Start and End are time marks. Between this two, the calculation will be done.</p> <p>Example:</p> <pre>trace = c0A1 for each item in enSlices(trace, 1E3) ; your code here tr = diff(item) next ; continous calculation until measurement is done cntr = 0</pre>

	<pre>for each item in EnSlices(c0A1,1E3, true) cntr += 1 next Pause("Numer of slices:"+cntr)</pre>
--	--

34.6 Exponential and Trigonometric

<p>ACos(a)</p>	<p>Calculates the arccosine function of a. The result variable matches the source of a.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Values that are located outside +/-1 are marked as undefined. </div> <p>The result for numbers is shown in degrees (0° to +180°). The result for curves is shown in radiant (0 to +Pi)</p> <p>Example:</p> <pre>a1 = 1 s1 = ACos(a1) ; 0° a2 = 0.5 ; s2 = ACos(a2) ; 60° ; convert rad <-> deg ; deg = 360 / (2*Pi) * rad ; rad = (2*Pi) /360 * deg</pre>
<p>ASin(a)</p>	<p>Calculates the arcsine function of a. The result matches the source of a.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Values that are located outside +/-1 are marked as undefined. </div> <p>The result for numbers is shown in degrees (-90° to +90°). The result for curves is shown in radiant (-Pi/2 to +Pi/2).</p> <p>Example:</p> <pre>a1 = 1 s1 = ASin(a1) ; 90° a2 = 0.5 s2 = ASin(a2) ; 30° ; convert rad <-> deg ; deg = 360 / (2*Pi) * rad ; rad = (2*Pi) /360 * deg</pre>
<p>Atan(a) Atan(y, x)</p>	<p>Calculates the arctangent function. The result matches the source of a. The result for numbers is shown in degrees (-90° to +90°). The result for curves is shown in radiant (-Pi/2 to +Pi/2).</p> <p>Calculates the arctangent function of y : x. corresponding to the four-quadrant polar diagram. x and y can be either numbers or curves (both y and x must be in the same form). The result matches the source. The result for numbers is shown</p>

	<p>in degrees (-180° to +180°). The result for curves is shown in radian (-Pi to +Pi).</p> <p>Example: a1 = 1 s1 = ATan(a1) ; 45°</p> <p>a2 = 0 s2 = ATan(a2) ; 0°</p> <pre>; convert rad <-> deg ; deg = 360 / (2*Pi) * rad ; rad = (2*Pi) /360 * deg ; alternatively to function angle() x = -3+5j ; complex number deg = atan(imag(x), real(x)) ; deg = 120.96°</pre>
Cos(a)	<p>Calculates the cosine function of a. The result matches the source of a. The value for a has to be in degrees (360° = full circle).</p> <p>Example: a1 = 0 ; deg s1 = Cos(a1) ; 1</p> <p>a2 = 60 ; deg s2 = Cos(a2) ; 0.5</p> <pre>; convert rad <-> deg ; deg = 360 / (2*Pi) * rad ; rad = (2*Pi) /360 * deg</pre>
Exp(a)	<p>Exp calculates the exponential function of a. The result matches the source of a.</p> <p>Example: ; Euler's number e = exp(1) ; 2.7183 val = ln(e) ; 1</p> <pre>;Discharge function of a capacitor R = 1E3 ; 1 kilo Ohm C = 100E-6 ; 100 micro Farad U = 10 ; Start voltage 10 V tsamp = 0.01 ; 100Hz sample interval ; create a list with calucated voltages vlist = List() for t = 0 to 1 step tsamp Uc = U * exp(-t/(R*C)) vlist()=Uc next ; convert the list into a trace tr[V] = ConvToTrace(vlist, 1/tsamp) ; calculate time for a defined voltage U1 = 4 t1 = ln((U1/U)) * (-R*C) ; place cursort on position of t1 SetCrs("Waveform 1", "A", t1)</pre>
Ln(a)	<p>Computes the natural logarithm of a. The result matches the source of a.</p>

	<p>Example:</p> <pre> ; Euler's number e = exp(1) ; 2.7183 val = ln(e) ; 1 ; Discharge function of a capacitor R = 1E3 ; 1 kilo Ohm C = 100E-6 ; 100 micro Farad U = 10 ; Start voltage 10 V tsamp = 0.01 ; 100Hz sample interval ; create a list with calucated voltages vlist = List() for t = 0 to 1 step tsamp Uc = U * exp(-t/(R*C)) vlist()=Uc next ; convert the list into a trace tr[V] = ConvToTrace(vlist, 1/tsamp) ; calculate time for a defined voltage U1 = 4 t1 = ln((U1/U))* (-R*C) ; place cursort on position of t1 SetCrs("Waveform 1", "A", t1) </pre>
Log(a [, base])	<p>Calculates the logarithm (base 10) of a. Define base to calculate the logarithm with a different base. The result matches the source of a.</p> <p>Example:</p> <pre> ; base = 10 without additional parameter l1 = log(100) ; l1 = 2 v1 = 10^l1 ; v1 = 100 ; base = 2 l2 = log(64, 2) ; l2 = 6 v2 = 2^l2 ; v2 = 64 ; amplifier calculation of gain in dB Uin = 1 Uout = 20 gain = 20*log(Uout/Uin) ; 26.021dB </pre>
Sin(a)	<p>Calculates the sine function of a. The result matches the source of a. The value for a has to be in degrees (360° = full circle).</p> <p>Example:</p> <pre> a1 = 90 ; deg s1 = Sin(a1) ; 1 a2 = 30 ; deg s2 = Sin(a2) ; 0.5 ; convert rad <-> deg ; deg = 360 / (2*Pi) * rad ; rad = (2*Pi) /360 * deg </pre>
Tan(a)	<p>Calculates the tangent function of a. The result matches the source of a. The value for a has to be in degrees (360° = full circle).</p> <p>Example:</p> <pre> a1 = 45 ; deg s1 = Tan(a1) ; 1 </pre>

```

a2 = 0 ; deg
s2 = Tan(a2) ; 0

; convert rad <-> deg
; deg = 360 / (2*Pi) * rad
; rad = (2*Pi) /360 * deg

```

34.7 File Functions

<p>CloseFile(a)</p>	<p>Closes the file (handle) with the name a. Example: Use CloseFile with WriteLine to start a new file number when you use the hashtag '#' inside of a.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  Opened files will be closed automatically when the formula is finished. </div> <p>Example:</p> <pre> filename = "meanMaxMean#.txt" for blockIdx = 0 to NBlks(c0A1) - 1 step 1.0 minVal = Min(c0A1.blockIdx) maxVal = Max(c0A1.blockIdx) meanVal = Mean(c0A1.blockIdx) WriteLine(filename, Tabulator, "Min:", "Max:", "Mean:") WriteLine(filename, Tabulator, minVal, maxVal, meanVal) ; We close here the file to start a new file CloseFile(filename) next </pre>
<p>Col</p>	<p>Keyword used for StoreReadouts function. With Col the corresponding column of the scalar table is saved.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  Please see also the keyword Line. </div> <p>Example:</p> <pre> StoreReadouts("readouts.txt", "Skalar_A 1", Col, 1) </pre>
<p>Comma</p>	<p>Key words for the Parameter separator of the functions WriteLine and WriteColumn. They determine the delimiter of the single values.</p>
<p>CopyFile(srcFilepath, destFilepath, overwrite [, cancellable])</p>	<p>Copy the file srcFilepath to the destination file destFilepath. Parameter overwrite as a Boolean allows overwriting an existing file if defined as True. The optional parameter cancellable as a Boolean defines if a copy process can be aborted or not. Return value will be 1 if file was successfully created and 0 if copying has failed.</p>
<p>CreateDirectory(path)</p>	<p>Creates a new directory with the name path. Existing directories will not be changed. There is no return value for this function.</p>
<p>DeleteDirectory(path)</p>	<p>Deletes an existing directory path with all subdirectories and files.</p>

	<p>Return value will be 1 if the path was successfully deleted and 0 if the directory was not found.</p> <div style="border: 1px solid gray; padding: 5px; background-color: #f0f0f0;">  Note that all files inside this directory will be deleted too without any confirmation! </div> <p>Example:</p> <pre>; Delete data folder in Experiment path variable = DeleteDirectory("data")</pre>
DeleteFile(a)	<p>Deletes one or more files which are defined by a. The Function responds with False (=0) when at least one file could not be deleted (the file is perhaps opened in the Signal Source Browser and/or its curves are being displayed on screen). True (=1) is the response if the files have been deleted or when no files are available.</p> <p>Example:</p> <pre>; Delete single file in data folder of Experiment DeleteFile("123.txt") ; Delete single file in a specific location test = DeleteFile("C:\xxx\xxx\TranAX_4.1\xxx.exp\123.txt") ; Delete multiple files from a list mylist = List() mylist() = "123.txt" mylist() = "456.txt" mylist() = "789.txt" test = DeleteFile(mylist)</pre>
DirectoryExists(path)	<p>Checks if path exists. Returns True if it does and False otherwise.</p> <p>Example:</p> <pre>path = GetPath("data") if DirectoryExists(path) = true then ; do something pause("Exists!") endif</pre>
File(filepath, index)	<p>The function File returns a curve from a file (Usually *.tpc5). The analog signals and the digital (marker) signals can be read out. In Multi block recordings, each individual block can be accessed. There could be only read one single block at once.</p> <p>filepath is a string containing the name of the file to open. There are only files with the extension *.tpc5 or *.tdp accepted. If no path is specified with the filename, the program searches the file in the DATA directory of the current experiment. For a file in another directory within the current experiment, it does not need to be specified the full path name.</p>

	<p>(E.g. instead of "C:\User\USERNAME\Documents\TransAS\EXPERIMENT.exp \Ref\NAME.tpc5") the term "..\Ref\NAME.tpc5" is sufficient.</p> <p>index is the number of the curve in the file (not to be mistaken with the channel number). 0 corresponds to the first curve. Example: File("filename.tpc5", 1) In this example, the second signal curve of the file is used for the calculation.</p> <p>With .Block an individual block of a multi-block shot can be accessed. The block number is optional. If it is missing always block 0 will be read. There can always be read only one block at once. Example: File("Filename", Index).Block</p> <p>With 'Marker, one of the two digital traces (markers) instead of the analog-curve (corresponding to the parameter index) is read out. This parameter can take the values 1 or 2. The result is a curve with the amplitude values 0 or 1. Markers are only present when for the original recording in the Control Panel "Averaging" was not set to 16 bits Example: File("crash.tpc5",0)'1</p> <p>Examples: signal_a = File("measure.tpc5", 1) Returns the analog signal of the second channel of the file "measure.tpc5".</p> <p>signal_m = File("measureblock.tpc5", 0).3'1 Returns the marker signal in the fourth block (starting with 0) of the first curve in the file "measureblock.tpc5".</p> <p>Example: <pre>filepath = "signal.tpc5" variable = File(filepath, 0)</pre></p>
FileExist(a)	<p>Checks if a file a exists. Returns True if it does and False otherwise.</p> <p>Example: <pre>if FileExist("xy.tpc5") = True then ;do something endif</pre></p>
FileIndexExists(a, index)	<p>Checks if an index (0...) exists in a. a can be a String with the name of a *.tpc5 file or also a Reference (e.g. Ref1). Return value will be a number. 1 means channel or trace index exists, 0 means unavailable.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> Also see NTracesInFile. For Reference-Pointers also the function Length can be used. Both functions return the number or curves.</p> </div> <p>Example: <pre>fname\$ = "test.tpc5"; Signale generieren</pre></p>

	<pre>tr1 = Noise(1, 1E3, 1E3) tr2 = Noise(2, 1E3, 1E3) Save(fname\$, tr1, tr2) ; Kurven speichern for a = 0 to 10 step 1 ; hier bis zu 11 Kurven val = FileIndexExists (fname\$, a) if val = True then ; ist Kurve vorhanden ? tr = File(fname\$, a) ; Kurve kann geholt werden ; weitere Berechnungen endif next</pre>
FileNameInc(filepath [, nrOfDigits])	<p>Increments the counter of filepath which contains a hashtag '#'. The optional parameter nrOfDigits can define how many digits the new filename should use.</p> <p>Example:</p> <pre>; Sine wave signal tr = Sinus(10,50,0,1E3,1E3) ; File name template fnameTemplate = "myTrace_#.tpc5" ; create 11 files for a = 0 to 10 step 1.0 fname = FileNameInc(fnameTemplate) Save(fname, tr) tr = tr*1.01 next</pre>
GetDirectoryName(filepath)	<p>Returns the directory name of filepath.</p> <p>Example:</p> <pre>dirName = GetDirectoryName("C:\Test\data.tpc5") ; dirName = "C:\Test"</pre>
GetFileExtension(filepath [, knownExtensions])	<p>Returns the file extension from filepath. The optional Parameter knownExtensions can be used to recognize some special file extension which have two or more dots in the extension, like Elsys *.tps.xml hardware settings.</p> <p>Example:</p> <pre>extension = GetFileExtension("C:\Test\data.tpc5") ; extension = ".tpc5"</pre>
GetFileInfo(filepath)	<p>Returns all relevant file parameters from filepath as a dictionary. The keys are as follow:</p> <p>Time Information "creationTime", "creationTimeUtc", "lastAccessTime", "lastAccessTimeUtc", "lastWriteTime", "lastWriteTimeUtc"</p> <p>Attributes "isReadOnly", "isHidden", "isDirectory", "isOffline", "isCompressed", "isEncrypted", "isNormal"</p> <p>Size, File Name and Path "sizeInBytes", "extension", "name", "directoryName", "fullName"</p> <p>Example:</p> <pre>; Create a file with one trace fname = "mytrace.tpc5" tr = Sinus(10,50,0, 1E3, 1E3) Save(fname, tr)</pre>

	<pre> ; Read file info info = GetFileInfo(fname) key = "sizeInBytes" if HasKey(info, key) = 1 then size = info(key) else size = notdefined endif </pre>
GetFilename(filepath, [, withoutExtension])	<p>Returns the just the file name, without path. The optional parameter withoutExtension defines if the filename should include the extension.</p> <p>Example:</p> <pre> fname1 = GetFilename("C:\Test\data.tpc5", false) ; fname1 = "data.tpc5" fname2 = GetFilename("C:\Test\data.tpc5", true) ; fname2 = "data" </pre>
GetFiles(srcPath [, pattern])	<p>Returns an array with all files available in srcPath. The optional parameter pattern can define if GetFiles should only return certain extensions (example: "*.tpc5").</p> <div data-bbox="571 887 1345 1025" style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> The function will not return files from subdirectories.</p> </div> <p>Example:</p> <pre> ; analyze and add all *.tpc5 files from a directory fileArray = GetFiles(GetPath("Data"), "*.tpc5") nItems = Length(fileArray) ; number of files found SetMessage("Found "+ nItems + " items(s)", True) sizeByte = 0 for each item in fileArray ;do some analysis info = GetFileInfo(item) key = "sizeInBytes" if HasKey(info, key) = 1 then val = info(key) sizeByte += val endif next Pause("File info:", nItems, sizeByte) </pre>
GetHarmonics(a, nrOfHarmonics)	<p>Computes the harmonics of a. The parameter nrOfHarmonics defines the number of harmonics to be calculated. Return value variable is of type Dictionary. The keys are defined as following:</p> <p>percent, peak, rms, decibel, frequency, phase</p> <p>Example:</p> <pre> tr = Sinus(10,50,0,10E3,10E3) ; create a noisy 50Hz signal tr = tr + Noise(2,tr) dictHarm = GetHarmonics(tr, 6) arr = dictHarm("peak") peak1st = arr(0) </pre>

	<pre>arr = dictHarm("frequency") Freq1st = arr(0)</pre>
GetNewestFile(srcPath [, pattern])	<p>Returns the newest file inside of directory sourcePath. The optional parameter pattern can restrict the files to certain extensions.</p> <p>Example:</p> <pre>sourcePath = GetPath("data") fnewest = GetNewestFile(sourcePath, "*.tpc5") tr = File(fnewest,0) ; do something with this trace</pre>
GetPath(a)	<p>Returns experiment specific directories. Possible values for parameter a are:</p> <p>experiment, experimentset, data, expr, commonexperimentpath, settingspath.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Furthermore, the standard Windows environment variables such as %appdata% or %programdata% are also supported. </div> <p>Example:</p> <pre>dataPath = GetPath("data")</pre>
Line	<p>Keyword, which is used for the function StoreReadouts. Line is the line/row of the Scalar table to store.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  See also keyword Col </div> <p>Example:</p> <pre>StoreReadouts("readouts.txt", "Skalar_A 1", Line, 1)</pre>
NTracesInFile(filepath)	<p>Returns a number indicating the amount of contained curves in file filepath. If the file filepath does not exist, the formula editor reports an error. There are only files with the extension *.tpc5 or *.tdp accepted. The rules for file paths are the same as for function File.</p> <p>Example:</p> <pre>; simple example if FileExist("myfile.tpc5") = True then Ntrace1=NTracesInFile("xy.tpc5") else Ntrace=0 endif ; advanced example, read all blocks from all traces in a file fname = "myfile.tpc5" ; valid filename and path? if FileExist(fname) = false then Pause("File not found") EndFormula endif minVal = List() ; create a list</pre>

	<pre> nTr = NTracesInFile(fname) ; number of traces in file for a = 0 to nTr-1 step 1.0 ; go through each trace minVal = List() for each item in enBlocks(file(fname, 0),false) ; go through each block ; do something with each block minVal() = Min(item) ; write min of each block into a list next next </pre>
PathCombine(path*)	<p>Combines multiple strings path to a valid path string (Adding backslashes between the paths).</p> <p>Example: Mainpath = "C:\Users" User = "Default" Directory = "Desktop"</p> <p>NewPath = PathCombine(Mainpath, User, Directory) ;NewPath = C:\Users\Default\Desktop</p>
ReadLine(filepath, separator)	<p>Read a complete line form a text file filepath. The values from the line will be separated by the delimiter separator. Return value is an array with all values.</p> <p>Example: ; create an example file filepath = "text.txt" WriteLine(filepath, Tabulator, "Voltage1", 10) WriteLine(filepath, Tabulator, "Voltage2", 10.5) CloseFile(filepath) lineArray1 = ReadLine(filepath, Tabulator) ; lineArray1 = [Voltage1, 10] lineArray2 = ReadLine(filepath, Tabulator) ; lineArray2 = [Voltage2, 10.5] CloseFile(filepath)</p>
RenameFile(srcFilepath, destFilepath, overwrite)	<p>Renames an existing file with the name srcFilepath to the new name destFilepath. If parameter overwrite is True, an existing file will be replaced.</p>
Save(filepath, [options,] trace*) Save(filepath, spectrum*) Save(filepath, filename*) Save(filepath, array) Save(filepath, [options,] list) Save(filepath, [options,] list, nameList nameArray)	<p>filepath is a string containing the name of the file to be created. The file name must have the extension *.tpc5 if the stored signals are to be reanalyzed with TranAX.</p> <p>A "#" (hash/pound symbol) at the end of the file name, is replaced by a sequential number. (e.g. "Check#.tpc5" results "Check001.tpc5", "Check002.tpc5", ...). The rules for file paths are the same as for the function "File (...)".</p> <p>trace is the variable for the curve to be saved. It can also be several comma-separated curves.</p> <p>The optional parameter options allows to define additional parameters, like the separator character for ASCII export.</p> <p>Ascii export "Conversion": "Marker", "Both" "Separator": "", ";", "," "DecimalPoint": ". (Point)", ", (Comma)"</p> <p>Tpc export (TransAS 2)</p>

"FilePerChannel": "true", "false"

SEG-Y export (geophysical data)

"Endianness": "LittleEndian", "BigEndian"

If the file has been **stored successfully**, the function **returns 1 (True)**, if an error occurs, the return is 0 (False).



An existing file with the same name is overwritten without warning.

filepath is a string containing the name of the file to be created. The file name must have the extension *.tpc5s if the stored signals are to be reanalyzed with TranAX.

A **"#" (hash/pound symbol)** at the end of the file name, is replaced by a **sequential number**.

(e.g. "Check#.tpc5s" results "Check001.tpc5s", "Check002.tpc5s", ...). The rules for file paths are the same as for the function **"File (...)"**.

spectrum is the variable for the **FFT curve** to be saved. It can also be several comma-separated curves.

If the file has been **stored successfully**, the function **returns 1 (True)**, if an error occurs, the return is 0 (False).



An existing file with the same name is overwritten without warning.

filepath is a string containing the name of the file to be created. The file name must have the extension *.tpc5 if the stored signals are to be reanalyzed with TranAX.

A **"#" (hash/pound symbol)** at the end of the file name, is replaced by a **sequential number**.

(e.g. "Check#.tpc5" results "Check001.tpc5", "Check002.tpc5", ...). The rules for file paths are the same as for the function **"File (...)"**.

filename designates the file names as a string, which should be saved in the new curve file. Multiple files separated by commas can also be specified.

trace is the variable for the curve to be saved. It can also be several comma-separated curves.

If the file has been **stored successfully**, the function **returns 1 (True)**, if an error occurs, the return is 0 (False).



An existing file with the same name is overwritten without warning.

filepath is a string containing the name of the file to be created. The file name must have the extension *.tpc5 if the stored signals are to be reanalyzed with TranAX.

A **"#" (hash/pound symbol)** at the end of the file name, is replaced by a **sequential number**.

(e.g. "Check#.tpc5" results "Check001.tpc5", "Check002.tpc5", ...). The rules for file paths are the same as for the function **"File (...)"**.

array is the variable for the curves to be saved.

If the file has been **stored successfully**, the function **returns 1 (True)**, if an error occurs, the return is 0 (False).



An existing file with the same name is overwritten without warning.

filepath is a string containing the name of the file to be created. The file name must have the extension *.tpc5 if the stored signals are to be reanalyzed with TranAX.

A **"#" (hash/pound symbol)** at the end of the file name, is replaced by a **sequential number**.

(e.g. "Check#.tpc5" results "Check001.tpc5", "Check002.tpc5", ...). The rules for file paths are the same as for the function **"File (...)"**.

list is the variable for the curves to be saved.

The optional parameter **options** allows to define additional parameters, like the separator character for ASCII export.

Ascii export

"Conversion": "Marker", "Both"

"Separator": "", ";", ",", "

"DecimalPoint": ". (Point)", ", (Comma)"

Tpc export (TransAS 2)

"FilePerChannel": "true", "false"

SEG-Y export (geophysical data)

"Endianness": "LittleEndian", "BigEndian"

If the file has been **stored successfully**, the function **returns 1 (True)**, if an error occurs, the return is 0 (False).



An existing file with the same name is overwritten without warning.

filepath is a string containing the name of the file to be created. The file name must have the extension *.tpc5 if the stored signals are to be reanalyzed with TranAX.

A **"#" (hash/pound symbol)** at the end of the file name, is replaced by a **sequential number**.

(e.g. "Check#.tpc5" results "Check001.tpc5", "Check002.tpc5", ...). The rules for file paths are the same as for the function **"File (....)"**.

nameList of type list or **nameArray** of type array are the **channel names** to be saved.

The optional parameter **options** allows to define additional parameters, like the separator character for ASCII export.

Ascii export

"Conversion": "Marker", "Both"

"Separator": "", ";", ",", "

"DecimalPoint": ". (Point)", ", (Comma)"

Tpc export (TransAS 2)

"FilePerChannel": "true", "false"

SEG-Y export (geophysical data)

"Endianness": "LittleEndian", "BigEndian"

If the file has been **stored successfully**, the function **returns 1 (True)**, if an error occurs, the return is 0 (False).



An existing file with the same name is overwritten without warning.

Example:

```
; 1. simple example
v[m/s] = Smooth(c0A1,200)
Save_OK = Save("Velocity.tpc5", v)
; File may be open
if Save_OK = False then
    ; Use a different name
    Save("Velocity-2.tpc5", v)
endif

; 2. simple example
; the # will be replaced with a number for each calculation
save("myTrace#.tpc5", c0A1) ; myTrace001.tpc5, etc.

; 3. simple example
; saves just 1. block
; TPC5 file
Save("save1_Block1.tpc5", c0A1, c0A2, c0A3)
; ASCII file
Save("save1_Block1.asd", c0A1, c0A2, c0A3)

; 4. simple example
; save all blocks
```

	<pre> ; TPC5 file Save("save1_AllBlocks.tpc5", "0A1", "0A2", "0A3") ; ASCII file Save("save1_AllBlocks.asd", "0A1", "0A2", "0A3") ; ### Advanced exmaples ### ; get 1. block from hardware channels trace1 = c0A1.0 trace2 = c0A2.0 trace3 = c0A3.0 ; define separator for ASCII export options = Array("Separator", ",") ; export ASCII with customiced sperator Save("save2.asd", options, trace1, trace2, trace3) ; save some specific parts from hardware channels ; the values withing [] defines block number ; and data reduction like Skip, Average or MinMax Save("save3a.tpc5", "0A1-2, 0A4", "0A3[1-5]", "0B1[0-5:Skip5]") Save("save3b.tpc5", "0A1-2, 0A4", "0A3[1-5]", "0B1[0-5:Average10]") Save("save3c.tpc5", "0A1-2, 0A4", "0A3[1-5]", "0B1[0-5:MinMax12]") ; save multiple filies into one file Save("save4.tpc5", "save3a.tpc5", "save3b.tpc5", "save3c.tpc5") ; use lists and strings mixed examples listTraces = List(trace1, trace2, trace3) Save("save5.tpc5", listTraces) Save("save5.asd", options, listTraces) listNames = List(GetChName(trace1), GetChName(trace2), GetChName(trace3)) val1 = Save("save6.tpc5", listTraces, listNames) val2 = Save("save6.asd", options, listTraces, listNames) arrChanDesc = Array("0A1-2", "0B1[0-5:Skip5]") Save("save7.asd", arrChanDesc) listNameMisc = List(GetChName(c0B2), GetChName(c0B3), GetChName(c0B1), GetChName(trace1), GetChName(trace2), GetChName(trace3)) listMisc = List("0B2", "0B3", "0B1[0-5:Skip5]", trace1, trace2, trace3) Save("save8.tpc5", listMisc, listNameMisc) ; exmaples for FFT traces traceFFT1 = FFT(trace1) traceSpec1 = ConvToSpectrumTrace(traceFFT1) traceFFT2 = FFT(trace2) traceSpec2 = ConvToSpectrumTrace(traceFFT2) traceFFT3 = FFT(trace3) traceSpec3 = ConvToSpectrumTrace(traceFFT3) Save("saveFFT1.tps5", traceSpec1, traceSpec2) ; Channels have to be opened in a FFT window. ; Make sure that you only save one block with the time window otherwise it will not work. Save("saveFFT2.tps5", "0A1-3") </pre>
Semicolon	Key words for the Parameter separator of the functions WriteLine and WriteColumn , they determine the delimiter of the single values.

Space	Key words for the Parameter separator of the functions WriteLine and WriteColumn . They determine the delimiter of the single values.
StoreReadouts(filepath, name, type, number [, results*])	<p>Saves single values Results determined by formula in a text file named FileName.</p> <p>Saves the results from the scalar table name in a text file named filepath. With the parameter type (Col/Line) and the corresponding number (LineNr or ColNr) the cells in the table are determined. Optionally, by formulas calculated single values Results can be appended for saving. The rules for file paths are the same as for File function. A "#" (hash/pound symbol) at the end of the file name, will be replaced by a sequential number.</p> <p>Example: <code>StoreReadouts(Filename, Results*)</code> <code>StoreReadouts(Filename, ScalarTableName, Line Column, LineNr ColNr[, Results]*)</code></p>
Tabulator	Key words for the Parameter separator of the functions WriteLine and WriteColumn . They determine the delimiter of the single values.
Write(filepath, value*)	Writes the parameters value into the file filepath . The function will append new values to the file. To create a new line you have to write "\r"C and "\n"C.
WriteColumn(filepath, separator, value*)	<p>In essence this function operates the same as WriteLine. The elements of an array and/or list are written to the same column. Thus, for every single parameter value a column is created. By repeating commands from WriteColumn new values are added to the bottom of these columns. As it often leads to confusion the command should only be given once. This function is also not very well suited when there are only single values as it takes the same file content as with WriteLine.</p> <p>Example: <code>WriteColumn(fname ,separator, value*)</code></p>
WriteLine(filepath, separator, value*)	<p>Writes value data in the text file filepath (usually in the data directory of the actual experiment).</p> <p>A # after the file name will be replaced by successive numbers. This way overwriting of existing files is prevented. separator can be any string (also just a single character such as a tab indicator). This way the separation of values is fixed. value* are single values or strings, including those that have been calculated earlier (names under Results). value can also be declared Array or List *).</p> <p>If the Text file already exists, all values are erased at first operation of the function. Following successive WriteLine commands (running the actual Formula-Commands) each iteration generates a new line.</p>

	<p>In case the Text file does not exist, it will be generated as filepath.</p> <p>Example: <code>WriteLine(fname ,separator, value*)</code></p>
--	--

34.8 Filter Functions

<p>BandPass(a, type, [, ripple] order, freq, bandwidth)</p>	<p>A Bandpass blocks lower and higher frequency ranges, lets through the middle range. a is a curve which is filtered. type specifies the filter type:</p> <ul style="list-style-type: none"> • Bessel • Butter (Butterworth) • Cheby (Chebyshev). For Chebyshev, the parameter ripple (ripple in dB) must be specified <p>order specifies the order (1 .. 9) of the filter. freq is the center frequency in Hz of the bandpass or the bandstop filter, bandwidth is the bandwidth in Hz.</p> <p>The return value variable again a curve.</p> <p>Example: <code>tr = c0A1</code> <code>trfilter = BandPass(tr, Bessel, 4, 10E3, 1E3)</code></p>
<p>BandStop(a, type [, ripple], order, freq, bandwidth)</p>	<p>A band-stop filter allows deeper and higher frequency ranges, blocks the middle range. a is a curve which is filtered. type specifies the filter type:</p> <ul style="list-style-type: none"> • Bessel • Butter (Butterworth) • Cheby (Chebyshev). For Chebyshev, the parameter ripple (ripple in dB) must be specified <p>order specifies the order (1 .. 9) of the filter. freq is the center frequency in Hz of the bandpass or the bandstop filter, bandwidth is the bandwidth in Hz.</p> <p>The return value variable again a curve.</p> <p>Example: <code>tr = c0A1</code> <code>trfilter = BandStop(tr, Bessel, 4, 10E3, 1E3)</code></p>
<p>Bessel</p>	<p>Keyword to type in function LowPass, HighPass, BandPass and BandStop.</p>
<p>Butter</p>	<p>Keyword to type in function LowPass, HighPass, BandPass and BandStop.</p>
<p>Cheby</p>	<p>Keyword to type in function LowPass, HighPass, BandPass and BandStop.</p>

<p>HighPass(a, type [, ripple], order, freq)</p>	<p>A high-pass filter cuts off low frequency components. The parameter a is a curve which is filtered. Parameter type specifies the filter type:</p> <ul style="list-style-type: none"> • Bessel, • Butter (Butterworth) • Cheby (Chebyshev). For Chebyshev, the parameter ripple (ripple in dB) must be specified <p>The parameter order specifies the filter order (1 .. 9). freq is the cut-off frequency in Hz. The return value variable is again a curve</p> <p>Example:</p> <pre>tr = Sinus(10,50,0,1E3,1E3) tr = tr + Noise(2,tr) trLO = LowPass(tr, Bessel, 4, 80) trHP = HighPass(tr, Bessel, 4, 80) ; optional to prevent phase shift trLoNoShift = Flip(trLO) trLoNoShift = LowPass(trLoNoShift, Bessel, 4, 80) trLoNoShift = Flip(trLoNoShift)</pre>
<p>LowPass(a, type [, ripple], order, freq)</p>	<p>A low-pass filter cuts off high frequency components. The parameter a is a curve which is filtered. Parameter type specifies the filter type:</p> <ul style="list-style-type: none"> • Bessel, • Butter (Butterworth) • Cheby (Chebyshev). For Chebyshev, the parameter ripple (ripple in dB) must be specified <p>The parameter order specifies the filter order (1 .. 9). freq is the cut-off frequency in Hz. The return value variable is again a curve</p> <p>Example:</p> <pre>tr = Sinus(10,50,0,1E3,1E3) tr = tr + Noise(2,tr) trLO = LowPass(tr, Bessel, 4, 80) trHP = HighPass(tr, Bessel, 4, 80) ; optional to prevent phase shift trLoNoShift = Flip(trLO) trLoNoShift = LowPass(trLoNoShift, Bessel, 4, 80) trLoNoShift = Flip(trLoNoShift)</pre>
<p>Median(a, width)</p>	<p>This function calculates the median over width sliding samples in a. The result is a curve. This function enables the elimination of spikes from the measurement curve.</p> <p>Example:</p> <pre>tr = c0A1 trfilter = Median(tr, 50)</pre>

Smooth(a, width)	<p>The function calculates the moving average of a over width samples. The return value variable is again a curve.</p> <p>Example: tr = c0A1 trfilter = Smooth(tr, 50)</p>
-------------------------	---

34.9 Layout Waveform

BottomAxis	<p>Keyword for the Parameter orientation in the function SetXRange.</p> <p>Example: ; Sets the x-axis to +/- 10 seconds SetXRange("Waveform 1", BottomAxis, 0, -10, 10)</p>
GetWaveforms(name)	<p>Reads the existing curves from a Waveform. The parameter name is the name of an existing Waveform Display. Return value variable is a list of dictionaries. The available keys are "name" and "filepath" if the curve is from a file.</p> <p>Example: nameWaveForm = "Waveform 1" dictT = GetWaveforms(nameWaveForm)</p> <pre>for each item in dictT key = "filepath" if HasKey(item, key) = 1 then filepath = item(key) endif key = "name" if HasKey(item, key) = 1 then nameTrace = item(key) endif Pause("Traces in Waveform", nameWaveForm, filepath, nameTrace) next</pre>
GetXRange(name, orientation, axisIdx)	<p>Reads the X-axis interval of the curve display name Return Value variable is an array of two numbers which represent the left and right bounds. The two parameters orientation and axisIdx are reserved for future extensions and can be set to 0.</p> <p>Example: rangeX = GetXRange("Waveform 1", 1, 0) rangeY = GetYRange("Waveform 1", 0, LeftAxis, 0) ; 1st Area</p> <pre>xMin = rangeX(0) yMax = rangeY(1)</pre>
GetYRange(name, arealdx, orientation, axisIdx)	<p>Reads the Y-axis interval of the area arealdx of the curve display name. Return Value variable is an array of two numbers representing the left and right boundaries. For the parameter orientation one of the two keywords LeftAxis or RightAxis is used. arealdx denotes the corresponding area in the trend view window, starting with 0 from the top. axisIdx is in most cases 0. If there are several Y-axes, this index is incremented</p>

	<p>accordingly from the inside to the outside. The values correspond to the respective physical units.</p> <p>Example: <code>rangeX = GetXRange("Waveform 1", 1, 0)</code> <code>rangeY = GetYRange("Waveform 1", 0, LeftAxis, 0) ; 1st Area</code></p> <p><code>xMin = rangeX(0)</code> <code>yMax = rangeY(1)</code></p>
LeftAxis	<p>Keyword for the Parameter orientation in the function SetYRange.</p> <p>Example: <code>; Sets the y range of the first left axis to +/- 5</code> <code>SetYRange("Waveform 1", 0, LeftAxis, 0, -5, 5)</code></p>
RightAxis	<p>Keywords for the Parameter orientation in the function SetYRange.</p> <p>Example: <code>; Sets the y range of the first right axis to +/- 5</code> <code>SetYRange("Waveform 1", 0, RightAxis, 0, -5, 5)</code></p>
SetXRange(name, orientation, axisIdx, min, max)	<p>Sets the X-axis-interval of the curve display name on min and max values. min and max can be interchanged, i.e., the smaller value always corresponds with the left side of the display. The two parameters orientation and axisIdx are reserved for future expansions.</p> <p>Example: <code>; auto fit the waveform to a random trace</code> <code>amp = Random(10,20)</code> <code>samples = Random(1E3, 10E3)</code></p> <p><code>tr = Sinus(amp,50,0,1E3,samples)</code> <code>tr = tr + Noise(2,tr)</code></p> <p><code>t1 = TBegin(tr)</code> <code>t2 = TEnd(tr)</code></p> <p><code>maxVal = Max(tr)</code> <code>minVal = Min(tr)</code></p> <p><code>; change X-axis of waveform 1 form -10 to +5 seconds</code> <code>SetXRange("Waveform 1", BottomAxis, false, t1-0.1, t2+0.1)</code> <code>SetYRange("Waveform 1", LeftAxis, false, false, minVal-2, maxVal+2)</code></p>
SetYRange(name, arealIdx, orientation, axisIdx, min, max)	<p>Sets the Y-axis-range-interval arealIdx of the curve display name on min and max values. For the Parameter orientation the two key words LeftAxis or RightAxis are being used. arealIdx indicates the corresponding range in the curve display window starting with 0 from the top. axisIdx in most cases is 0. When more Y-axes are available, the index is incrementally expanded from the inside out. min and max can be interchanged, i.e., the smaller value always corresponds with the bottom of the display. The values reflect the respective physical units.</p> <p>Example: <code>; auto fit the waveform to a random trace</code></p>

	<pre> amp = Random(10,20) samples = Random(1E3, 10E3) tr = Sinus(amp,50,0,1E3,samples) tr = tr + Noise(2,tr) t1 = TBegin(tr) t2 = TEnd(tr) maxVal = Max(tr) minVal = Min(tr) ; change X-axis of waveform 1 from -10 to +5 seconds SetXRange("Waveform 1", BottomAxis, false, t1-0.1, t2+0.1) SetYRange("Waveform 1", LeftAxis, false, false, minVal-2, maxVal+2) </pre>
--	---

34.10 Measurement Flow Control

mfclsCalculating()	<p>Checks if Measurement Flow Control (MFC) is active.</p> <p>Return value variable is of type integer: 1 = MFC is active, is being executed 0 = MFC is inactive, has been stopped</p> <p>Example:</p> <pre> ; create an MFC with two tasks, both enbaled ; with an delay of 5s as executable function ; start Measurement Flow Control mfstart() ; do some code delay(2) ; stop the second task val = mfstopTask("Task 2") if val = 0 then Pause("Task not found!") endif ; check for aktive MFC status = mfclsCalculating() if status = 1 then ; stop Measurement Flow Control retvalStop = mfstop(false, true) endif ; ckeck again status of ative MFC status = mfclsCalculating() if status = 0 then Pause("MFC stopped!") endif </pre>
mfstart()	<p>Starts the Measurement Flow Control (MFC). First, the Initialize Tasks and then all other tasks are started according to the configuration. This function has no additional parameter or return value.</p> <p>Example:</p> <pre> ; create an MFC with two tasks, both enbaled ; with an delay of 5s as executable function ; start Measurement Flow Control </pre>

	<pre> mfcstart() ; do some code delay(2) ; stop the second task val = mfcStopTask("Task 2") if val = 0 then Pause("Task not found!") endif ; check for aktive MFC status = mfcIsCalculating() if status = 1 then ; stop Measurement Flow Control retvalStop = mfcStop(false, true) endif ; ckeck again status of ative MFC status = mfcIsCalculating() if status = 0 then Pause("MFC stopped!") endif </pre>
<p>mfcStop(stopImmediately, waitUntilStopped)</p>	<p>Stops the Measurement Flow Control (MFC). The two parameters of type boolean have the following meaning: stopImmediately: If true, the MFC will stop immediately. If false, the Terminate task will run properly like the stop button in the ribbon bar. waitUntiStopped: If true, the formula will stall until the MFC exits; if false, the next step in the formula will be executed immediately.</p> <p>Return value variable is of type integer, 0 means no errors</p> <p>Example:</p> <pre> ; create an MFC with two tasks, both enbaled ; with an delay of 5s as executable function ; start Measurement Flow Control mfcstart() ; do some code delay(2) ; stop the second task val = mfcStopTask("Task 2") if val = 0 then Pause("Task not found!") endif ; check for aktive MFC status = mfcIsCalculating() if status = 1 then ; stop Measurement Flow Control retvalStop = mfcStop(false, true) endif ; ckeck again status of ative MFC status = mfcIsCalculating() if status = 0 then Pause("MFC stopped!") </pre>

	<code>endif</code>
mfcStopTask(name)	<p>Stops a running MFC task. The parameter name as string corresponds to the title of the corresponding task. Return value is 1 if the task exists. If this Task name should not exist, 0 will be returned. In this case, the formula will not stop, nor an exception will be raised.</p> <p>Example:</p> <pre> ; create an MFC with two tasks, both enbaled ; with an delay of 5s as executable function ; start Measurement Flow Control mfcstart() ; do some code delay(2) ; stop the second task val = mfcStopTask("Task 2") if val = 0 then Pause("Task not found!") endif ; check for aktive MFC status = mfcIsCalculating() if status = 1 then ; stop Measurement Flow Control retvalStop = mfcStop(false, true) endif ; ckeck again status of ative MFC status = mfcIsCalculating() if status = 0 then Pause("MFC stopped!") endif </pre>

34.11 Misc. Functions

Name[Unit] = Expression	<p>Principle of a formula.</p> <p>Example:</p> <pre> ; set units to traces i[A] = c0A1 u[V] = c0A2 P[W] = i * u ; modify unit of a trace UnitP = GetUnit(P) P2 = P*1000 SetUnit(P, "m"+UnitP) </pre>
Audio_Sr_44100	<p>Returns the value 44100. This constant can be used in function PlaySound to indicate the standard sample rate.</p> <p>Example:</p> <pre> tr = Sinus(1,440,0,100E3,100E3) PlaySound(tr, Audio_Sr_44100) </pre>
BoardDriverVersion	Keyword for the function GetVersion .
BoardFirmwareVersion	Keyword for the function GetVersion .
BoardHardwareVersion	Keyword for the function GetVersion .

ClearReference(a)	<p>Deletes the entries and curves from the specified reference a.</p> <p>Example: ; Clears all existing references listRef = GetReferenceNames() for each item in listRef ClearReference(item) next ; Clear one specific reference ClearReference("ref1")</p>
ElapsedWatch()	<p>Reads the internal counter, started with the function StartWatch. Returned time will be in milliseconds. Please see also the functions StartWatch and StopWatch.</p> <p>Example: StartWatch() Delay(1.1) t1 = ElapsedWatch() ; approx. 1'100ms Delay(0.5) t2 = StopWatch() ; approx. 1'600ms</p>
EnableSaveHdf(bool)	<p>Prevents files from being created in the expr directory of the current experiment.</p> <p>The parameter bool of type Boolean has the following functions:</p> <p>True = * .tpc5 Data is generated, False = no * tpc5 files are stored. By default, this is set to True.</p> <p>Example: ; Read all files in the current data directory path = GetPath("data") files = GetFiles(path, "*.tpc5") ; Disable writing of *.tpc5 files EnableSaveHdf(False) for each item in files trace = File(item,0) minVal = Min(trace) maxVal = Max(trace) next ; Enable writing of *.tpc5 files EnableSaveHdf(True) Pause("values", minVal, maxVal)</p>
EnableScreenUpdate(bool)	<p>Enables or disable the screen update during calculation of formulas.</p> <p>It's recommended to disable screen update (bool = false) to increase the performance of formula calculations.</p> <p>Parameter bool is a Boolean, true means update enabled, false means disabled.</p> <p>Example: EnableScreenUpdate(false) ; your code here: ; calculation of multiple traces EnableScreenUpdate(True)</p>

GetActiveWindow()	<p>Returns the name of the active window as a text.</p> <p>Example: <code>s = GetActiveWindow()</code> <code>Pause(s)</code></p>
GetInfo(key, deviceIdx [, boardIdx [, inputIdx]])	<p>Returns detailed information from a device, board or channel according the parameter key. Following keys are implemented.</p> <p>Device: url, deviceName, deviceId, isSimulated, deviceDescription, deviceMacAddress, useLocalTime, autoStartMeasurement, autoStartAutoSequence, serverPort1, serverPort2, hasTwoInOne, hasTwoInOnePW, dataFilename, nrOfBackupFiles, hdFlushInterval, writeThroughCache, modelType, freeDiskSpace, totalDiskSpace</p> <p>Board: hasecr, hasAdvancedTrigger, maxBlockLength, maxSampleRate, nrOfInputs, has16Bit, hasSlewRateTrigger, serialNr, boardClass</p> <p>Input: hasIcp, hasFilterModule, adcResolution, maxMarkerMask, maxAdcSpeed, hasFullRange, hasDiffModule, has50Ohm, inputClass, isSimulated</p> <p>Example: <code>url = GetInfo("URL", 0); IP of device</code> <code>nrOfInputs = GetInfo("nrOfInputs", 0, 0); number of channels of board A</code></p>
GetNrOfBoards(deviceIdx)	<p>Returns the number of Boards (Modules) in the instrument device. deviceIdx = 0 corresponds to first device.</p> <p>Example: <code>; change settings for each channel</code> <code>nrDev = GetNrOfDevices()</code> <code>for dev = 0 to nrDev-1</code> <code> nrBoard = GetNrOfBoards(dev)</code> <code> for brd = 0 to nrBoard-1</code> <code> nrInp = GetNrOfInputs(dev, brd)</code> <code> for inp = 1 to nrInp</code> <code> ; do some code</code> <code> SetRecordingPar(dev,brd,inp,23, 10) ; range 10V</code> <code> next</code> <code> next</code> <code>next</code></p>
GetNrOfDevices()	<p>Returns the number of Devices in the configured measuring System.</p> <p>Usually only one device is in a system, multiple devices must be synchronized with an Elsys SyncLink box. Furthermore, they have to be connected collectively via Control Panel -> Devices.</p> <p>Example:</p>

	<pre> ; change settings for each channel nrDev = GetNrOfDevices() for dev = 0 to nrDev-1 nrBoard = GetNrOfBoards(dev) for brd = 0 to nrBoard-1 nrInp = GetNrOfInputs(dev, brd) for inp = 1 to nrInp ; do some code SetRecordingPar(dev,brd,inp,23, 10) ; range 10V next next next </pre>
<p>GetNrOfInputs(deviceIdx, boardIdx)</p>	<p>Returns the number of Channels on a Board in the instrument device. deviceIdx = 0 corresponds to first device. boardIdx = 0 corresponds to first board "A" in the device.</p> <p>Example:</p> <pre> ; change settings for each channel nrDev = GetNrOfDevices() for dev = 0 to nrDev-1 nrBoard = GetNrOfBoards(dev) for brd = 0 to nrBoard-1 nrInp = GetNrOfInputs(dev, brd) for inp = 1 to nrInp ; do some code SetRecordingPar(dev,brd,inp,23, 10) ; range 10V next next next </pre>
<p>GetNrOfReferences()</p>	<p>Gets the number of references defined in the Signal Source Browser.</p> <p>Example:</p> <pre> ; Get the number of references from the Signal Source Browser nRef = GetNrOfReferences() </pre>
<p>GetRecordingPar(deviceIdx, boardIdx, inputIdx, parameter)</p>	<p>Reads the current settings of a hardware channel.</p> <div data-bbox="710 1489 1460 1624" style="border: 1px solid gray; padding: 5px; background-color: #f0f0f0;">  The Parameters deviceIdx and boardIdx starts with 0, inputIdx starts with 1. </div> <p>Depending on the command, parameter is of type number or text. Depending on the command parameter, either one or the other type is used. A list of all available parameters can be found in the documentation of TpcAccess.</p> <p>http://wiki.elsys-instruments.com/index.php?title=TpcAccess</p> <p>Example:</p> <pre> ; read range and channel name of A1 Range = GetRecordingPar(0,0,1, 23) ChnName = GetRecordingPar(0,0,1, "ChName") </pre>

	<pre> ; read Operation Mode ; 0=Scope, 1=Multi Block, 2=continuous ; 3=ECR Single Channel, 67=with Dual Mode ; 131=ECR Multi Channel, 195=with Dual Mode OperationMode = GetRecordingPar(0,0,1, 1) </pre>
GetReference(a)	<p>Reads a reference, channel, block or marker from the references and returns it. Return value variable is a curve.</p> <p>Example:</p> <pre> ; Create a reference tr1 = Sinus(10,50,0, 1E3,1E3) tr2 = Noise(2,tr1) tr3 = tr1 +tr2 tr4 = tr1 -tr2 tr5 = Smooth(tr4, 50) SetReference("ref1", false, tr1, tr2, tr3, tr4, tr5) trRead = GetReference("ref1",1) ; tr2 </pre>
GetReferenceNames()	<p>Returns a list of active references. Return value variable is of type List with the names of the references as string. Please see also the function GetReference and NTracesInReference.</p> <p>Example:</p> <pre> ListRef = GetReferenceNames() ; get a list with all references ppValList = List() ; prepare a list for peak-peak values for each item in ListRef ; go through each reference ntr = NTracesInReference(item) ; read the number of traces in the reference for n = 0 to ntr-1 step 1.0 ; to through each trace tr = GetReference(item, n) ppValList() = PeakPeak(tr) ; Do something the the traces next next </pre>
GetSSBFiles()	<p>Returns a list with all files from the Signal Source Browser. Files can be placed into the Signal Source Browser by drag & drop from the Windows File Explorer. Afterwards, the selected files can be analysed with the Formula Editor. Allowed file types are *.tpc5, *.tps5, *.tdp and *.bdf for analysis in the Formula Editor.</p> <p>Example:</p> <pre> fileList = GetSSBFiles() for each item in fileList tr = File(item,0) ; do something with this trace next </pre>
GetVersion(key)	<p>Returns a version number of Hardware or Software. Depending of key keyword applied, more Parameters deviceldx and boardIdx will be needed. This Function returns a text.</p>

	<p>0 = ProgramVersion; no additional parameter 1 = ServerVersion; deviceIndex 2 = BoardHardwareVersion; deviceIndex and boardIndex 3 = BoardDriverVersion; deviceIndex and boardIndex 4 = BoardFirmwareVersion; deviceIndex and boardIndex 5 = InputHardwareVersion; deviceIndex and boardIndex\</p> <p>Example: <pre> ; create a dictionary verDict = Dictionary() ; read all parameters verDict("ProgramVersion") = GetVersion(ProgramVersion) verDict("ServerVersion") = GetVersion(ServerVersion, 0) verDict("BoardHardwareVersion") = GetVersion(BoardHardwareVersion ,0,0) verDict("BoardDriverVersion") = GetVersion(BoardDriverVersion,0,0) verDict("BoardFirmwareVersion") = GetVersion(BoardFirmwareVersion,0,0) verDict("InputHardwareVersion") = GetVersion(InputHardwareVersion,0,0) ; print e.g. TranAX version pause("TranAX version: "+verDict("ProgramVersion")) </pre></p>
<p>InputHardwareVersion</p>	<p>Keyword for the function GetVersion.</p> <p>Example: <pre> ; Returns the input hardware version of Board 0A inputHwVer = GetVersion(InputHardwareVersion, 0, 0) </pre></p>
<p>MergeTraces([insertMark,], trace*) MergeTraces([insertMark,], list) MergeTraces(filepath, name, [channelIdx, [blockNumber, triggerSample, triggerTime,]] trace*)</p>	<p>Merges multiple curves trace to one curve. When the optional parameter insertMark = True is set, special markers are created to separate the original curves. Please see also the function Merge.</p> <p>Merges multiple curves from list to one curve. When the optional parameter insertMark = True is set, special markers are created to separate the original curves. Please see also the function Merge.</p> <p>Merges multiple curves trace to one curve. filepath defines the path and filename in which the new trace will be saved. name defines the name of the trace within the file. The optional parameter channelIdx as an integer beginning with 0 defines the channel. The optional parameters blockNumber defines a specific block, beginning with 0, , triggerSample the sample for trigger location and triggerTime the time stamp of the trigger event. Please see also the function Merge.</p> <p>Example: <pre> ; merge multiple blocks into one trace </pre></p>

	<pre> s10 = c0A1.0 ; Block 0 (Multiblock Recording) s11 = c0A1.1 ; Block 1 s12 = c0A1.2 ; Block 2 s1All = MergeTraces(True, s10, s11, s12) ; create a new signal tr1 = Sinus(10,50,0,1E3,1E3) tr2 = Sinus(10,100,0,1E3,1E3) tr = MergeTraces(false, tr1, tr2, tr1) ; create a file with a Multi Block recording trace1 = Sinus(1, 1e3, 0, 1e6, 1e4) trace2 = Sinus(1, 1e3, 0, 1e6, 1e4) trace2 = Shift(trace2, 15e-3) trace2 = SetTrigger(trace2, 15e-3) filepath = "test.tpc5" MergeTraces(filepath, "trace", 0, 0, 0, TTrigger(trace1), trace1) MergeTraces(filepath, "trace", 0, 1, 0, TTrigger(trace2), trace2) CloseFile(filepath) </pre>
<p>NBlks(trace) NBlks(filepath, index)</p>	<p>Returns the number of blocks of the measurement trace.</p> <div data-bbox="596 891 1345 1028" style="border: 1px solid gray; padding: 5px; background-color: #f0f0f0;"> <p> trace is for logical reasons a measurement curve recorded with Multi Block or ECR Mode.</p> </div> <p>Returns the number of blocks of the measurement curve in filepath with the index index.</p> <p>Example:</p> <pre> ; get the maximum of all blocks fname\$ = "Test-1.tpc5" trace = file(fname\$,0) blk = NBlks(trace) Mx=-1000 ; initialize i=0 For i = 0 to blk-1 value = Max(file(fname\$,0).i) If value > Mx Then Mx = value Endif Next </pre>
<p>NormDist(a, mean, standardDev, cumulativeBool)</p>	<p>Returns the normal distribution for the specified mean and standard deviation. Values for variable are 0 ...1 (0 =0%, 1= 100%).</p> <p>a: Is the value for which the distribution is calculated.</p> <p>mean: the arithmetic mean of the distribution.</p> <p>standardDev: The standard deviation of the distribution.</p> <p>cumulativeBool: Which determines the form of the function. If TRUE, NormDist returns the cumulative distribution function. If it is FALSE, the probability mass function is returned.</p> <p>Example:</p> <pre> ; create some random values ; e.g. time duration for a light barrier in milliseconds items = 100 </pre>

	<pre> myArr = Array(0 to items-1) as double for a = 0 to items-1 step 1.0 myArr(a) = random(58, 74) ; random values form 58ms to 74ms next tmp = StdDev(myArr, items) ; StdDev returns an array standardDeviation = tmp(items-1) ; use last item meanValue = mean(myArr) ; calculate mean ; calculate the percentage of light barrier values below 68 ms pcVal = NormDist(68, meanValue, standardDeviation, true) ; calculate the minimal duration to be on the upper 75% msVal = NormInv(0.25, meanValue, standardDeviation) </pre>
<p>NormInv(probability, mean, standardDev)</p>	<p>Returns the inverse of the normal cumulative distribution for the specified mean and standard deviation.</p> <p>probability: A probability according to the normal distribution. mean: The arithmetic mean of the distribution. standardDev: The standard deviation of the distribution.</p> <p>Example:</p> <pre> ; create some random values ; e.g. time duration for a light barrier in milliseconds items = 100 myArr = Array(0 to items-1) as double for a = 0 to items-1 step 1.0 myArr(a) = random(58, 74) ; random values form 58ms to 74ms next tmp = StdDev(myArr, items) ; StdDev returns an array standardDeviation = tmp(items-1) ; use last item meanValue = mean(myArr) ; calculate mean ; calculate the percentage of light barrier values below 68 ms pcVal = NormDist(68, meanValue, standardDeviation, true) ; calculate the minimal duration to be on the upper 75% msVal = NormInv(0.25, meanValue, standardDeviation) </pre>
<p>NTracesInReference(a)</p>	<p>Returns the number of curves in a reference a, Return value variable is a number.</p> <p>If a reference is empty or does not exist, this function returns 0, otherwise the number of curves available.</p> <p>Please see also the function GetReference and GetReferenceNames.</p> <p>Example:</p> <pre> ; simple example ntr = NTracesInReference("ref12") ; more advanced example ListRef = GetReferenceNames() ; get a list with all references ppValList = List() ; prepare a list for peak-peak values </pre>

	<pre> for each item in ListRef ; go through each reference ntr = NTracesInReference(item) ; read the number of traces in the reference for n = 0 to ntr-1 step 1.0 ; to through each trace tr = GetReference(item, n) ppVallist() = PeakPeak(tr) ; Do something the the traces next next </pre>
Pi	<p>Keyword Pi = 3.14159265359.</p> <p>Example:</p> <pre> ; RC lowpass R = 1E3 C = 100E-9 fg = 1/(2*Pi*R*C) ; fg = approx. 1.6kHz ; area of a disk r = 0.5 A = r^2 * pi </pre>
PlaySound(a [, fs])	<p>Audio and tpc5 files can be played by the audio system in the computer. *.wav, *.mp3 and *.tpc5 files are supported. Also a resulting from an earlier calculation can be played (equivalent to a *.tpc5 File).</p> <p>The parameter fs must be set if a *.tpc5 files or a calculated trace is used. The max. possible Sample Rate (in Hz) samplerate and max. Amplitude depends on the installed audio system. This means: Playing of a trace or *.tpc5 file would only be warranted, if the sample rate of the trace is appropriate. It is advised to first resample with scaling of the trace to a suitable audio frequency and amplitude, see Example. For *.wav and *.mp3 files the parameter fs must not be set.</p> <p>Example:</p> <pre> AmpFact=0.5 ; Left channel sLeft=File("SoundData.tpc5",0) ; Right channel sRight=File("SoundData.tpc5",1) ; Adapt to audio frequency sLeft=AmpFact*Resampling(sLeft, 44100) sRight=AmpFact*Resampling(sRight, 44100) ; Left and right channel in one file Save ("SoundStereo.tpc5", sLeft, sRight) srRec=1/TSample(sLeft) ; Play it PlaySound("SoundStereo.tpc5", srRec) </pre>
ProgramVersion	<p>Keyword for the function GetVersion.</p> <p>Example:</p> <pre> ; Returns the version of the application progVer = GetVersion(ProgramVersion) </pre>
Random(min, max)	<p>Returns a random number between the defined limits min and max.</p> <p>Example:</p> <pre> ; one random number </pre>

	<pre>n = Random(0,100) ; create an array with random numbers myArr = Array(0 to 9) as double for a = 0 to Length(myArr)-1 step 1.0 myArr(a) = Random(-10, 10) next</pre>
ServerVersion	<p>Keyword for the function GetVersion.</p> <p>Example:</p> <pre>; Returns the version of the server of device 0 progVer = GetVersion(ServerVersion, 0)</pre>
SetFormulaError(text)	<p>This function may be used to set an error trap on a dedicated position in the formula program to prevent further erroneous results.</p> <p>Example:</p> <pre>Tb=TBegin(trc) ; initialize for a = 1 to 10 Tcr = TCross(trc,tb,TEnd(trc),0) tb = Tcr ; for next If Tcr = NotDefined Then text\$ = "Missing Zero Crossings" SetFormulaError(text\$) endif next</pre>
SetRecordingMode(deviceIdx, mode)	<p>Sets the recording mode of the DAQ-device deviceIdx. The parameter deviceIdx is an integer and in most cases 0, if just one device is used. In case of multiple DAQ-devices (clustering), the parameter deviceIdx is ascending according to the integration into TranAX, usually according to its IP address.</p> <p>The parameter mode as an integer sets the recording mode. The following values are valid: Scope = 0, MultiBlock = 1, Continuous = 3, EcrSingle = 4, EcrMul-ti = 5, EcrSingleDual = 6, EcrMultiDual = 7</p> <p>Example:</p> <pre>; continuous recording SetRecordingMode(0, 3)</pre>
SetRecordingPar(deviceIdx, boardIdx, inputIdx, parameter, value)	<p>Sets the settings of a hardware channel.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Note: The Parameters deviceIdx and boardIdx starts with 0, input starts with 1. </div> <p>Depending on the command, parameter is of type number or text. Depending on the command parameter, either one or the other type is used. A list of all available parameters can be found in the documentation of TpcAccess: Elsys-Wiki</p>

	<p>Also depending on the command parameter, function SetRecordingPar expects a number or a text.</p> <p>Example:</p> <pre> ; set range of A1 to 10V SetRecordingPar(0, 0, 1, 23, 10) ; set channel name A1 SetRecordingPar(0, 0, 1, "ChName", "Input Voltage") ; set offset 75% SetRecordingPar(0, 0, 1, 24, 75) ; ; set input mode "Single Ended" SetRecordingPar(0, 0, 1, 20, 1) ; 0:Off, 1:SE, 2:Diff endformula ; change settings for each channel nrDev = GetNrOfDevices() for dev = 0 to nrDev-1 nrBoard = GetNrOfBoards(dev) for brd = 0 to nrBoard-1 nrInp = GetNrOfInputs(dev, brd) for inp = 1 to nrInp ; do some code SetRecordingPar(dev,brd,inp,23, 10) ; range 10V next next next </pre>
<p>SetReference(name, fileList*) SetReference(name, assignHardware, trace*)</p>	<p>Assigns hardware channels or curves to a reference. References can be used to replace traces in a defined layout quickly and easily. See also the possibilities of the Signal Source Browser. Parameters are:</p> <p>name: Is the reference itself as a string ("ref1", "ref2", ... "refn")</p> <p>fileList: From type List for assignment of curves from files in *.tpc5 or *.bdf format</p> <p>Assigns hardware channels or curves to a reference. References can be used to replace traces in a defined layout quickly and easily. See also the possibilities of the Signal Source Browser. Parameters are:</p> <p>name: Is the reference itself as a string ("ref1", "ref2", ... "refn")</p> <p>assignHardware: Is used to assign a hardware channel, parameter is of type Boolean.</p> <p>If a hardware channel is assigned via a variable, the channel name is specified in the case of True, the variable name in the references in the case of False. For a direct assignment of a hardware channel (c0A1, ...) to the function parameters, True must be used.</p> <p>trace*: Are channels, variables or calculated curves from formulas</p> <p>Example:</p> <pre> ; Add hardware channels to a reference ; Hardware name (c0A1, ...) will be listed in the references SetReference("ref1", true, c0A1, c0A2) </pre>

	<pre> ; Add hardware channels as a variable to a reference ; Hardware names (c0A1, ...) will be listed in the references tr1 = c0A1 tr2 = c0A2 SetReference("ref2", true, tr1, tr2) ; Add hardware channels as a variable to a reference ; Variable names will be listed in the references tr1 = c0A1 tr2 = c0A2 SetReference("ref3", false, tr1, tr2) ; Mixed application, traces and hardware channels ; Variable names will be listed in the references measured = c0A1 calculated = Sinus(10,50,0, measured) delta = measured - calculated filtered = Smooth(delta,500) SetReference("ref4", false, measured, calculated, delta, filtered) ; Using files fname = "mytrace.tpc5" Save(fname, calculated, filtered) listFname = List(fname, 1) ; Either a trace number in the file (starts with 0) or -1 for all files SetReference("ref5", listFname) </pre>
StartWatch()	<p>Starts the internal counter, this can be used for debugging Formula Editor code to measure the duration of calculations. Please see also the functions ElapsedWatch and StopWatch.</p> <p>Example: StartWatch() Delay(1.1) t1 = ElapsedWatch() ; approx. 1'100ms Delay(0.5) t2 = StopWatch() ; approx. 1'600ms</p>
StopWatch()	<p>Stops the internal counter, started with the function StartWatch. Returned time will be in milliseconds. Please see also the functions StartWatch and ElapsedWatch.</p> <p>Example: StartWatch() Delay(1.1) t1 = ElapsedWatch() ; approx. 1'100ms Delay(0.5) t2 = StopWatch() ; approx. 1'600ms</p>
UseMemory(bool)	<p>Curve results are stored in the memory and not as a file in directory when bool = True.</p> <div data-bbox="710 1758 1428 1892" style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  This will shorten processing time considerably </div> <p>UseMemory(bool) corresponds to the function SetEnvironment(0, true).</p> <p>Example: UseMemory(True) ; Curve data is stored in RAM</p>

34.12 PLC Functions

<p>plcAnalogRead(connectionAddr, pin [,mode])</p>	<p>Defines a pin as an analogue input. Please note not every pin can be configured as an analogue input. Return value is the ADC value itself, for Elsys PLC devices a value between 0 and 1023 (10bit ADC).</p> <p>connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address pin: Arduino Pin according pin out list mode The optional parameter mode is used to read statistic data from the analogue input. Update interval is set to 100ms. Possible values for parameter mode are: 0 (or not defined): Actual ADC value 1: Minimum value 2: Maximum value 3: Mean value 4: RMS value 666: Reset statistic data for mode 1-4 above</p> <p>Example:</p> <pre> ; Input AI0 used as an analogue input WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; Clear Analogue Statistic plcAnalogRead(port, 54, 666) adcval = plcAnalogRead(port, 54) ; equal to plcAnalogRead(port, 54, 0) minval = plcAnalogRead(port, 54, 1) maxval = plcAnalogRead(port, 54, 2) meanval = plcAnalogRead(port, 54, 3) rmsval = plcAnalogRead(port, 54, 4) plcClose(port) </pre>
<p>plcAnalogWrite(connectionAddr, pin, value [, duration] [, mode])</p>	<p>Defines a pin as an analogue output. Please note not every pin can be configured as an analogue output, just pins which supports PWM.</p> <p>connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address pin: Arduino Pin according pin out list value: Defines the output signal in thousandth (0.1%) interval. 0 means 0V, 1000 means max output signal, 10V for Elsys PLC devices. duration: The optional parameter duration is used to generate a ramp as an output signal. It will reach the analogue output value after the time duration in seconds. mode: The optional parameter mode is used to generate an exponential output signal, which allows linear dimming connected LEDs. Not defined, False or 0 means Linear output, True or 1 means exponential output</p>

	<p>Example:</p> <pre> ; Example 1, Output A00 used as an analogue output. WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; Set output to 0V plcAnalogWrite(port, 12, 0) ; Wait 1 second delay(1) ; Set output to 10V, ramp for 2 seconds plcAnalogWrite(port, 12, 1000, 2) plcClose(port) ; Example 2, Output A00 used as an analogue output for LEDs. WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; Set output to 0V plcAnalogWrite(port, 12, 0) ; Wait 1 second delay(1) ; Set output to 10V, ramp for 2 seconds plcAnalogWrite(port, 12, 1000, 2, True) plcClose(port) </pre>
<p>plcClose(connectionAddr)</p>	<p>Closes a Serial Port or Ethernet connection to an Elsys PLC device.</p> <p>connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address</p> <p>Example:</p> <pre> ; Examples 1, default Ethernet connection: ; open a Ethernet Connection WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; your code here plcClose(port) ; Examples 2, default serial port connection: ; open a Serial Port Connection WriteForNext("PORT", "COM15") port = ReadOfPrevious("PORT") plcOpen(port) ; your code here plcClose(port) ; Examples 3, custom serial port connection: dictParams = Dictionary() dictParams("baudrate") = 115200 dictParams("readTimeout") = 2000 dictParams("writeTimeout") = 2000 dictParams("newline") = "\n" dictParams("dtrEnable") = False WriteForNext("PORT", "COM15") port = ReadOfPrevious("PORT") plcOpen(port, dictParams) ; your code here </pre>

<p>plcDigitalRead(connectionAddr, pin [,overwrite])</p>	<p>plcClose(port)</p> <p>Defines a pin as a digital input. Return value would be 0 if input signal is low, and 1 if input signal is high.</p> <p>connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address pin: Arduino Pin according pin out list overwrite: The optional parameter overwrite is used to redefine a pin function. If a pin is defined as an output, plcDigitalRead returns its value. With the optional parameter overwrite = true, the pin will be reconfigured as a digital input.</p> <p>Example:</p> <pre> ; Input AI0 is connection with a push button to ; Vin of the Elsys PLC. WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; Pushbutton pressed InputVal= 1, else = 0 InputVal = plcDigitalRead(port, 54) </pre> <p>plcClose(port)</p>
<p>plcDigitalWrite(connectionAddr, pin, value [, pulseDuration])</p>	<p>Defines a pin as a digital output, sets its status according to value.</p> <p>connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address pin: Arduino Pin according pin out list value: Valid parameter for are: plcLow, equal to integer value 0, output is OFF, plcHigh, equal to integer value 1, output is ON, plcToggle, equal to integer value 2, inverts the output (high switches to low and vice versa) All three parameters are predefined keywords. pulseDuration; The optional parameter pulseDuration is used to define a Monoflop function, the output will set to value for the defined duration in seconds (integer or float).</p> <p>Example:</p> <pre> ; Example 1: Two examples of a Monoflop Output. ; The second solution with additional parameter ; pulseDuration doesn't block the code ; execution WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; 1. Monoflop plcDigitalWrite(port, 22, plcHigh) Delay(1.5) plcDigitalWrite(port, 22, plcLow) </pre>

	<pre> ; 2. Monoflop plcDigitalWrite(port,22,plcHigh, 1.5) plcClose(port) ;Example 2: Toggle output from ON to OFF and vice versa. WriteForNext("PORT","192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) plcDigitalWrite(port,22,plcToggle) plcClose(port) </pre>
plcHigh	<p>Keyword value in function plcDigitalWrite</p> <p>plcLow, equal to integer value 0, output is OFF plcHigh, equal to integer value 1, output is ON plcToggle, equal to integer value 2, inverts the output (high switches to low and vice versa)</p> <p>Example:</p> <pre> ; Example 1: Two examples of a Monoflop Output. ; The second solution with additional parameter ; pulseDuduration doesn't block the code execution WriteForNext("PORT","192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; 1. Monoflop plcDigitalWrite(port,22,plcHigh) Delay(1.5) plcDigitalWrite(port,22,plcLow) ; 2. Monoflop plcDigitalWrite(port,22,plcHigh, 1.5) plcClose(port) ;Example 2: Toggle output from ON to OFF and vice versa. WriteForNext("PORT","192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) plcDigitalWrite(port,22,plcToggle) plcClose(port) </pre>
plcLow	<p>Keyword value in function plcDigitalWrite</p> <p>plcLow, equal to integer value 0, output is OFF plcHigh, equal to integer value 1, output is ON plcToggle, equal to integer value 2, inverts the output (high switches to low and vice versa)</p> <p>Example:</p> <pre> ; Example 1: Two examples of a Monoflop Output. ; The second solution with additional parameter ; pulseDuduration doesn't block the code execution WriteForNext("PORT","192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) </pre>

	<pre> ; 1. Monoflop plcDigitalWrite(port,22,plcHigh) Delay(1.5) plcDigitalWrite(port,22,plcLow) ; 2. Monoflop plcDigitalWrite(port,22,plcHigh, 1.5) plcClose(port) ;Example 2: Toggle output from ON to OFF and vice versa. WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) plcDigitalWrite(port,22,plcToggle) plcClose(port) </pre>
<p>plcOpen(connectionAddr [, parameters])</p>	<p>Opens a serial or ethernet (port 80) connection to an Elsys PLC device. It is possible to combine multiple devices. Therefore, each PLC function and command needs the parameter connectionAddr. It is recommended to save the connection Address as a variable. In case of changing the IP Address or the serial port number, changes has just to be done once in the code. In the examples, connctionAddr will be saved with the function WriteForNext and read with ReadOfPrevious. The benefit of these two functions is to use the connectionAddr in code segments of the Measurement Flow Control (MFC) and in functions inside the Formula Editor code.</p> <p>In case of a still opened connection at start of a Formula, plcOpen will automatically try to reuse the Serial Port or connection from the previous calculation. In case a serial device is already opened by another application, this application must be closed first.</p> <p>connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address parameters: Generally, there are no additional parameters necessary. In case of redefining serial port parameters, a dictionary with the following parameters are allowed: baudrate, dataBits, stopBits, readTimeout, writeTimeout, newline and dtrEnable</p> <p>Example:</p> <pre> ; Examples 1, default Ethernet connection: ; open a Ethernet Connection WriteForNext("PORT", "192.168.0.35") </pre>

	<pre> port = ReadOfPrevious("PORT") plcOpen(port) ; your code here plcClose(port) ; Examples 2, default serial port connection: ; open a Serial Port Connection WriteForNext("PORT","COM15") port = ReadOfPrevious("PORT") plcOpen(port) ; your code here plcClose(port) ; Examples 3, custom serial port connection: dictParams = Dictionary() dictParams("baudrate") = 115200 dictParams("readTimeout") = 2000 dictParams("writeTimeout") = 2000 dictParams("newLine") = "\n" dictParams("dtrEnable") = False WriteForNext("PORT","COM15") port = ReadOfPrevious("PORT") plcOpen(port, dictParams) ; your code here plcClose(port) </pre>
<p>plcRead (connectionAddr [, delayMs [, collectResults]])</p>	<p>Read a parameter from connectionAddr. Returns a string or a double value, or a list depending on the response. If a timeout occurs the return value will be NotDefined. connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address delayMs: Defines a delay in milliseconds after reading the data. This can be used if the next read or write command must wait a specific time. collectResults: If you want to do multiple reads set this to true. You will receive a list with all the return values.</p> <p>Example:</p> <pre> WriteForNext("PORT","COM12") port = ReadOfPrevious("PORT") plcOpen(port) ; read data from a 3rd party serial device val = plcRead(port, 200, true) plcClose(port) </pre>
<p>plcTemperatureRead(connectionAddr, pin [, sensor])</p>	<p>Defines a pin as a OneWire pin for DS1820 temperature sensors.</p> <p>connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address pin: Arduino Pin according pin out list sensor: The optional parameter sensor is used to define the number of a connected sensor. OneWire support up to 127 sensors. Not defined means the first sensor in</p>

	<p>chain, 0 ... 126 a specific sensor number, 666 get number of devices in OneWire chain.</p> <p>Return value variable is then either the temperature of the sensor or the number of available sensors.</p> <p>Example:</p> <pre> ; OneWire on Pin A1, 55, with two DS18B20 sensors, 4k7 Ohm Resistor Pullup WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; Get Number of sensors TempSensors = plcTemperatureRead(port, 55, 666) ; Read first sensor val1 = plcTemperatureRead(port, 55) ; Read second sensor val2 = plcTemperatureRead(port, 55, 1) plcClose(port) </pre>
plcToggle	<p>Keyword value in function plcDigitalWrite</p> <p>plcLow, equal to integer value 0, output is OFF plcHigh, equal to integer value 1, output is ON plcToggle, equal to integer value 2, inverts the output (high switches to low and vice versa)</p> <p>Example:</p> <pre> ; Example 1: Two examples of a Monoflop Output. ; The second solution with additional parameter ; pulseDuduration doesn't block the code execution WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; 1. Monoflop plcDigitalWrite(port, 22, plcHigh) Delay(1.5) plcDigitalWrite(port, 22, plcLow) ; 2. Monoflop plcDigitalWrite(port, 22, plcHigh, 1.5) plcClose(port) ; Example 2: Toggle output from ON to OFF and vice versa. WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) plcDigitalWrite(port, 22, plcToggle) plcClose(port) </pre>
plcWrite(connectionAddr, message [, delayMs [, collectResults]])	<p>Write a message, based on a string to connectionAddr. If a timeout occurs the return value will be NotDefined.</p>

	<p>connectionAddr: Serial port or IP-Address of the Elsys PLC as String. E.g. "COM12" for a serial port or "192.168.0.35" for an IP-Address message: Elsys PLC specific String, for other devices according its specification. delayMs: Defines a delay in milliseconds after writing the data. This can be used if the next read or write command must wait a specific time. collectResults Set collectResults to true if you want to automatically capture all the responses. You will receive a list with values.</p> <p>Example:</p> <pre> ; Example 1: Read an analogue input WriteForNext("PORT", "192.168.0.35") port = ReadOfPrevious("PORT") plcOpen(port) ; read analog input A0 RMS value val = plcWrite(port, "PN:54;TP:3;P0:4;") plcClose(port) ; Example 2: Control an Elsys AE-Amp device dictParams = Dictionary() dictParams("baudrate") = 19200 dictParams("readTimeout") = 2000 dictParams("writeTimeout") = 2000 dictParams("newLine") = "\n" dictParams("dtrEnable") = False WriteForNext("PORT", "COM3") port = ReadOfPrevious("PORT") plcClose(port) plcOpen(port, dictParams) plcWrite(port, "RESET;") Delay(0.2) plcWrite(port, "MUX:0;SETHVVAL:400;") Delay(0.2) pause("Check High Voltage Signal!"); plcWrite(port, "MUX:0;SETHVPULSE:60;") Delay(0.2) plcWrite(port, "MUX:0;SETHVINR:5;") Delay(0.2) plcWrite(port, "ADD:0;CHN:1;SETHV:1;") Delay(0.2) plcWrite(port, "MUX:0;PULSESTART:666;") Delay(0.1) plcClose(port) </pre>
--	--

34.13 Power Functions

<p>ApparentPower(a, b [, start, end [, level, hysteresis]])</p>	<p>Calculates the apparent power from a and b. One parameter should represent a voltage and the other a current. The return value is a scalar of type double. The optional parameters start and end are time marks. Between this two, the calculation will be done. Other optional parameters are level and hysteresis. If they are not defined, level is set at the</p>
--	---

	<p>average of the peak value, hysteresis is defined as 5% of the peak value.</p> <p>Example:</p> <pre> ; Generate two test traces, phase shift of current deltaPhase = 20 ; would be 20° which indicates an inductive load voltageTrace = Sinus(330, 50, 0, 1e6, 1e6) currentTrace = Sinus(10, 50, -1*deltaPhase, 1e6, 1e6) ; 1.) Formula Editor power functions cosphi_1 = CosPhi(voltageTrace, currentTrace) ApparentPower_1[VA] = ApparentPower(voltageTrace, currentTrace) RealPower_1[W] = RealPower(voltageTrace, currentTrace) ReactivePower_1[VAR] = ReactivePower(voltageTrace, currentTrace) crestfac_1 = CrestFactorPeriodic(voltageTrace) powerfac_1 = PowerFactor(voltageTrace, currentTrace) ; 2.) Manual calculation phaseTrace = Phase(voltageTrace, currentTrace) phaseShift_2[°] = phaseTrace(TEnd(phaseTrace)) rmsU[V] = RMS(voltageTrace) rmsI[A] = RMS(currentTrace) cosphi_2 = cos(phaseShift_2) ApparentPower_2[VA] = rmsU* rmsI RealPower_2[W] = rmsU* rmsI * cos(phaseShift_2) ReactivePower_2[VAR] = rmsU* rmsI * sin(phaseShift_2) crestfac_2 = max(voltageTrace) / rmsu powerfac_2 = cos(phaseShift_2) ; the same as cosPhi ; 3.) some more information ; cosPhi = RealPower / ApparentPower ; cosPhi = PowerFactor ; cosPhi = R / Z = resistance / impedance = Wirkwiderstand / Scheinwiderstand </pre>
<p>CosPhi(a, b [, start, end [, level, hysteresis]])</p>	<p>Calculates the cosine phi of a and b. The return value is a scalar of type double. The optional parameters start and end are time marks. Between this two, the calculation will be done. Other optional parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is defined as 5% of the peak value.</p> <div data-bbox="555 1576 1347 1715" style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  In order to achieve meaningful results, there should be available at least 3.5 periods. </div> <p>Example:</p> <pre> ; Generate two test traces, phase shift of current deltaPhase = 20 ; would be 20° which indicates an inductive load voltageTrace = Sinus(330, 50, 0, 1e6, 1e6) currentTrace = Sinus(10, 50, -1*deltaPhase, 1e6, 1e6) ; 1.) Formula Editor power functions cosphi_1 = CosPhi(voltageTrace, currentTrace) ApparentPower_1[VA] = ApparentPower(voltageTrace, currentTrace) RealPower_1[W] = RealPower(voltageTrace, currentTrace) ReactivePower_1[VAR] = ReactivePower(voltageTrace, currentTrace) </pre>

	<pre> crestfac_1 = CrestFactorPeriodic(voltageTrace) powerfac_1 = PowerFactor(voltageTrace, currentTrace) ; 2.) Manual calculation phaseTrace = Phase(voltageTrace, currentTrace) phaseShift_2[°] = phaseTrace(TEnd(phaseTrace)) rmsU[V] = RMSP(voltageTrace) rmsI[A] = RMSP(currentTrace) cosphi_2 = cos(phaseShift_2) ApparentPower_2[VA] = rmsU* rmsI RealPower_2[W] = rmsU* rmsI * cos(phaseShift_2) ReactivePower_2[VAR] = rmsU* rmsI * sin(phaseShift_2) crestfac_2 = max(voltageTrace) / rmsu powerfac_2 = cos(phaseShift_2) ; the same as cosPhi ; 3.) some more information ; cosPhi = RealPower / ApparentPower ; cosPhi = PowerFactor ; cosPhi = R / Z = resistance / impedance = Wirkwiderstand / Scheinwiderstand </pre>
<p>CrestFactor(a [, start, end])</p>	<p>Calculates the crest factor of a. This is the ratio of the absolute peak value to RMS of a. The return value is a scalar of type double. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <p>Example:</p> <pre> ; Generate two test traces, phase shift of current deltaPhase = 20 ; would be 20° which indicates an inductive load voltageTrace = Sinus(330, 50, 0, 1e6, 1e6) currentTrace = Sinus(10, 50, -1*deltaPhase, 1e6, 1e6) ; 1.) Formula Editor power functions cosphi_1 = CosPhi(voltageTrace, currentTrace) ApparentPower_1[VA] = ApparentPower(voltageTrace, currentTrace) RealPower_1[W] = RealPower(voltageTrace, currentTrace) ReactivePower_1[VAR] = ReactivePower(voltageTrace, currentTrace) crestfac_1 = CrestFactorPeriodic(voltageTrace) powerfac_1 = PowerFactor(voltageTrace, currentTrace) ; 2.) Manual calculation phaseTrace = Phase(voltageTrace, currentTrace) phaseShift_2[°] = phaseTrace(TEnd(phaseTrace)) rmsU[V] = RMSP(voltageTrace) rmsI[A] = RMSP(currentTrace) cosphi_2 = cos(phaseShift_2) ApparentPower_2[VA] = rmsU* rmsI RealPower_2[W] = rmsU* rmsI * cos(phaseShift_2) ReactivePower_2[VAR] = rmsU* rmsI * sin(phaseShift_2) crestfac_2 = max(voltageTrace) / rmsu powerfac_2 = cos(phaseShift_2) ; the same as cosPhi ; 3.) some more information ; cosPhi = RealPower / ApparentPower ; cosPhi = PowerFactor ; cosPhi = R / Z = resistance / impedance = Wirkwiderstand / Scheinwiderstand </pre>
<p>CrestFactorPeriodic(a [, start, end])</p>	<p>Calculates the crest factor of entire periods from a. This is the ratio of the absolute peak value to RMS of a. The return value</p>

	<p>is a scalar of type double. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <p>Example:</p> <pre> ; Generate two test traces, phase shift of current deltaPhase = 20 ; would be 20° which indicates an inductive load voltageTrace = Sinus(330, 50, 0, 1e6, 1e6) currentTrace = Sinus(10, 50, -1*deltaPhase, 1e6, 1e6) ; 1.) Formula Editor power functions cosphi_1 = CosPhi(voltageTrace, currentTrace) ApparentPower_1[VA] = ApparentPower(voltageTrace, currentTrace) RealPower_1[W] = RealPower(voltageTrace, currentTrace) ReactivePower_1[VAR] = ReactivePower(voltageTrace, currentTrace) crestfac_1 = CrestFactorPeriodic(voltageTrace) powerfac_1 = PowerFactor(voltageTrace, currentTrace) ; 2.) Manual calculation phaseTrace = Phase(voltageTrace, currentTrace) phaseShift_2[°] = phaseTrace(TEnd(phaseTrace)) rmsU[V] = RMS(voltageTrace) rmsI[A] = RMS(currentTrace) cosphi_2 = cos(phaseShift_2) ApparentPower_2[VA] = rmsU* rmsI RealPower_2[W] = rmsU* rmsI * cos(phaseShift_2) ReactivePower_2[VAR] = rmsU* rmsI * sin(phaseShift_2) crestfac_2 = max(voltageTrace) / rmsu powerfac_2 = cos(phaseShift_2) ; the same as cosPhi ; 3.) some more information ; cosPhi = RealPower / ApparentPower ; cosPhi = PowerFactor ; cosPhi = R / Z = resistance / impedance = Wirkwiderstand / Scheinwiderstand </pre>
<p>FundamentalPower(a, b [, start, end [, level, hysteresis]])</p>	<p>Calculates the effective power of the fundamental wave, excluding the contribution of harmonics of the two curves a and b. One should represent a voltage and the other a current. The return value is a scalar of type double. The optional parameters start and end are time marks. Between this two, the calculation will be done. Other optional parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is defined as 5% of the peak value. In order to achieve meaningful results, the signals should be periodically.</p> <p>Example:</p> <pre> tr1 = c0a1 tr2 = c0a2 fPower = FundamentalPower(tr1, tr2) </pre>
<p>PowerFactor(a, b [, start, end [, level, hysteresis]])</p>	<p>Calculates the ratio of real power to apparent power of the two curves a and b. One should represent a voltage and the other a current. The return value is a scalar of type double. The optional parameters start and end are time marks. Between this two, the calculation will be done. Other optional</p>

	<p>parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is defined as 5% of the peak value.</p> <p>Example:</p> <pre> ; Generate two test traces, phase shift of current deltaPhase = 20 ; would be 20° which indicates an inductive load voltageTrace = Sinus(330, 50, 0, 1e6, 1e6) currentTrace = Sinus(10, 50, -1*deltaPhase, 1e6, 1e6) ; 1.) Formula Editor power functions cosphi_1 = CosPhi(voltageTrace, currentTrace) ApparentPower_1[VA] = ApparentPower(voltageTrace, currentTrace) RealPower_1[W] = RealPower(voltageTrace, currentTrace) ReactivePower_1[VAR] = ReactivePower(voltageTrace, currentTrace) crestfac_1 = CrestFactorPeriodic(voltageTrace) powerfac_1 = PowerFactor(voltageTrace, currentTrace) ; 2.) Manual calculation phaseTrace = Phase(voltageTrace, currentTrace) phaseShift_2[°] = phaseTrace(TEnd(phaseTrace)) rmsU[V] = RMS(voltageTrace) rmsI[A] = RMS(currentTrace) cosphi_2 = cos(phaseShift_2) ApparentPower_2[VA] = rmsU* rmsI RealPower_2[W] = rmsU* rmsI * cos(phaseShift_2) ReactivePower_2[VAR] = rmsU* rmsI * sin(phaseShift_2) crestfac_2 = max(voltageTrace) / rmsU powerfac_2 = cos(phaseShift_2) ; the same as cosPhi ; 3.) some more information ; cosPhi = RealPower / ApparentPower ; cosPhi = PowerFactor ; cosPhi = R / Z = resistance / impedance = Wirkwiderstand / Scheinwiderstand </pre>
<p>ReactivePower(a, b [, start, end [, level, hysteresis]])</p>	<p>Calculates the reactive power from of real power and apparent power of the two curves a and b. One should represent a voltage and the other a current. The return value is a scalar of type double. The optional parameters start and end are time marks. Between this two, the calculation will be done. Other optional parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is defined as 5% of the peak value.</p> <p>Example:</p> <pre> ; Generate two test traces, phase shift of current deltaPhase = 20 ; would be 20° which indicates an inductive load voltageTrace = Sinus(330, 50, 0, 1e6, 1e6) currentTrace = Sinus(10, 50, -1*deltaPhase, 1e6, 1e6) ; 1.) Formula Editor power functions cosphi_1 = CosPhi(voltageTrace, currentTrace) ApparentPower_1[VA] = ApparentPower(voltageTrace, currentTrace) RealPower_1[W] = RealPower(voltageTrace, currentTrace) ReactivePower_1[VAR] = ReactivePower(voltageTrace, currentTrace) </pre>

	<pre> crestfac_1 = CrestFactorPeriodic(voltageTrace) powerfac_1 = PowerFactor(voltageTrace, currentTrace) ; 2.) Manual calculation phaseTrace = Phase(voltageTrace, currentTrace) phaseShift_2[°] = phaseTrace(TEnd(phaseTrace)) rmsU[V] = RMS(voltageTrace) rmsI[A] = RMS(currentTrace) cosphi_2 = cos(phaseShift_2) ApparentPower_2[VA] = rmsU* rmsI RealPower_2[W] = rmsU* rmsI * cos(phaseShift_2) ReactivePower_2[VAR] = rmsU* rmsI * sin(phaseShift_2) crestfac_2 = max(voltageTrace) / rmsu powerfac_2 = cos(phaseShift_2) ; the same as cosPhi ; 3.) some more information ; cosPhi = RealPower / ApparentPower ; cosPhi = PowerFactor ; cosPhi = R / Z = resistance / impedance = Wirkwiderstand / Scheinwiderstand </pre>
<p>RealPower(a, b [, start, end [, level, hysteresis]])</p>	<p>Calculates the real power from the instantaneous power of the two curves a and b. One should represent a voltage and the other a current. The return value is a scalar of type double. The optional parameters start and end are time marks. Between this two, the calculation will be done. Other optional parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is defined as 5% of the peak value. In order to achieve meaningful results, the signals should be periodically.</p> <p>Example:</p> <pre> ; Generate two test traces, phase shift of current deltaPhase = 20 ; would be 20° which indicates an inductive load voltageTrace = Sinus(330, 50, 0, 1e6, 1e6) currentTrace = Sinus(10, 50, -1*deltaPhase, 1e6, 1e6) ; 1.) Formula Editor power functions cosphi_1 = CosPhi(voltageTrace, currentTrace) ApparentPower_1[VA] = ApparentPower(voltageTrace, currentTrace) RealPower_1[W] = RealPower(voltageTrace, currentTrace) ReactivePower_1[VAR] = ReactivePower(voltageTrace, currentTrace) crestfac_1 = CrestFactorPeriodic(voltageTrace) powerfac_1 = PowerFactor(voltageTrace, currentTrace) ; 2.) Manual calculation phaseTrace = Phase(voltageTrace, currentTrace) phaseShift_2[°] = phaseTrace(TEnd(phaseTrace)) rmsU[V] = RMS(voltageTrace) rmsI[A] = RMS(currentTrace) cosphi_2 = cos(phaseShift_2) ApparentPower_2[VA] = rmsU* rmsI RealPower_2[W] = rmsU* rmsI * cos(phaseShift_2) ReactivePower_2[VAR] = rmsU* rmsI * sin(phaseShift_2) crestfac_2 = max(voltageTrace) / rmsu powerfac_2 = cos(phaseShift_2) ; the same as cosPhi ; 3.) some more information </pre>

	<pre> ; cosPhi = RealPower / ApparentPower ; cosPhi = PowerFactor ; cosPhi = R / Z = resistance / impedance = Wirkwiderstand / Scheinwiderstand </pre>
TotalHarmonicDistortion(a [, start, end [, level, hysteresis]])	<p>Analyses the curve a for the fundamental frequency and calculates the value of THD (Total Harmonic Distortion) in % of the periodic signal. Return value is a scalar of type double in percent. The optional parameters start and end are time marks. Between this two, the calculation will be done. Other optional parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is defined as 5% of the peak value.</p> <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin: 10px 0;">  In order to achieve meaningful results, there should be available at least two periods. </div> <p>Example: trace = c0A1 THD = TotalHarmonicDistortion(trace)</p>

34.14 Programming Functions

Calculate(filepath, waitForCalculation, showResults)	<p>Computes a formula *.for within a formula. The return value variable is of type Dictionary and contains the calculated variables of the called formula.</p> <p>filepath is the formula file to be calculated with the extension *.for. You can also run formulas from ExperimentSets *.zip. The parameter waitForCalculation defines if formula waits for completion. waitForCalculations set to False may be used if the calculated values are not used within this formula, e.g. Update in Waveform or other detached functions. showResults displays the calculated values in Traces and Results tabs.</p> <p>Example: path = GetPath("experimentset") path = PathCombine(path, "Formula 2.for")</p> <pre> ; contains dictionary of results from Formula 2.for retval = Calculate(path, true, true) globalVar = ReadOfPrevious("GLOBALVAR") pause(globalVar) EndFormula ; Create a second Formula with the name Formula 2.for ; Copy and paste the code below into this file ; Save file with Ctrl+s a = 5 tr = Sinus(10,50,0,10E3,1E3) mylist = List() mylist() = 12 mylist() = "Example" WriteForNext("GLOBALVAR","Formula 2 calculated!") </pre>
Delay(seconds)	<p>The process will be delayed by seconds.</p>

	<p> The delay will not be accurate. It depends on parallel running processes of the operating system (up to a few milliseconds).</p> <p>Example: <code>Delay(1) ; 1 second</code> <code>Delay(100E-3) ; 100 milliseconds</code></p>
Do-Until	<p>Creates a loop which runs block until the condition a=0 is True. The condition expression can use the following operators: "=", ">", "<", ">=", "<=", "<>". Multiple conditions can be concatenated with the words and, or and not.</p> <p> The operator <> corresponds to not equal (!=)</p> <p>Example: <code>i = 100</code> <code>do until i = 15</code> <code> i = i-1</code> <code>loop</code></p>
Do-While	<p>Creates a loop which runs block at least once. The condition a=0 is evaluated at the end of the loop and decides if the loop will be finished. The condition expression can use the following operators: "=", ">", "<", ">=", "<=", "<>". Multiple conditions can be concatenated with the words and, or and not.</p> <p> The operator <> corresponds to not equal (!=)</p> <p>Example: <code>i = 0</code> <code>do</code> <code> i = i + 1</code> <code>loop while i < 100</code></p>
EndFormula	<p>This operation indicates the last line of the formula code. If it is missing, all formula lines are processed. It is especially useful in the development of formula programs. It prevents the execution of code after Endformula, without deleting or commenting out.</p> <p>Example: <code>sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4)</code> <code>; We stop the calculation from this line</code> <code>EndFormula</code> <code>; Add an offset to the curve</code> <code>sineTrace = sineTrace + 10</code></p>
Exitloop	<p>With this operation, a loop will be exited.</p> <p>Example: <code>for i=1 to 4 step 1</code> <code> if i=3 then</code> <code> exitloop</code> <code> endif</code></p>

<p>False</p>	<p><code>next</code></p> <p>Keyword False, corresponds to the value 0. It indicates that a condition is not met.</p> <p>Example:</p> <pre>if FileExist("xy.tpc5") <> False then BLOCK endif</pre>
<p>For-Each</p>	<p>The loop for each allows easily going through countable elements. It will return every single item from the enumeration enumerable. The following data types can be used: List, Dictionary, Array, String, Trace.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p> When a Dictionary is used for enumeration item will contain the key from the value pair.</p> </div> <p>Example:</p> <pre>Pause("Start Multi Block recording with 5 blocks. Click continue when done.") ; 1. Example ; get maximum signal from each block maxVal = NotDefined for each item in EnBlocks(c0A1,false) Mx = Max(item) if (Mx > maxVal) or (maxVal = NotDefined) then maxVal = Mx endif endif next ; 2. Example ; - Analyse hardware channel ; - Data from reference ; - Data from saved file ; Save channel A1 to a file Save("myMultiBlockSignal.tpc5", "0A1") ; saves all recorded blocks ; Please note: This saves just the first block! Save("mytest.tpc5", c0A1) ; Set channel A1 to Reference 1 SetReference(ref1,true, c0A1) ; true means references to the hardware channel n1 = NBlks(c0A1) n2 = NBlks(ref1) n3 = NBlks(File("myMultiBlockSignal.tpc5",0)) cnt1 = 0 cnt2 = 0 cnt3 = 0 minVal1 = List() minVal2 = List() minVal3 = List() ; Hardware channel for each item in EnBlocks(c0A1,false) cnt1 += 1 tr1 = item minVal1() = Min(item) endif next</pre>

	<pre> ; Reference for each item in EnBlocks(ref1,false) cnt2 += 1 tr2 = item minVal2() = Min(item) next ; From File for each item in EnBlocks(File("myMultiBlockSignal.tpc5",0),false) cnt3 += 1 tr3 = item minVal3() = Min(item) next </pre>
<p>For-Loop</p>	<p>Creates a loop which runs block while the condition a <= limit or a >=limit is True. The variable a will be incremented at the end of the loop by the step value b.</p> <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p> If the variable a is changed during the loop you have to make sure that it doesn't result into an infinite loop.</p> </div> <p>Example:</p> <pre> Pause("Start Multi Block recording with 5 blocks. Click continue when done.") ; 1. Example ; get maximum signal from each block maxVal = NotDefined for i = 0 to NBlks(c0A1)-1 Mx = Max(c0A1.i if (Mx > maxVal) or (maxVal = NotDefined) then maxVal = Mx endif endif next ; 2. Example ; - Analyse hardware channel ; - Data from reference ; - Data from saved file ; Save channel A1 to a file Save("myMultiBlockSignal.tpc5", "0A1") ; saves all recorded blocks ; Please note: This saves just the first block! Save("mytest.tpc5", c0A1) ; Set channel A1 to Reference 1 SetReference(ref1,true, c0A1) ; true means references to the hardware channel n1 = NBlks(c0A1) n2 = NBlks(ref1) n3 = NBlks(File("myMultiBlockSignal.tpc5",0)) cnt1 = 0 cnt2 = 0 cnt3 = 0 minVal1 = List() minVal2 = List() minVal3 = List() ; Hardware channel for a = 0 to n1-1 step 1.0 </pre>

	<pre> tr1 = c0A1.a minVal1() = Min(tr1) next ; Reference for a = 0 to n1-1 step 1.0 tr2 = ref1.a minVal2() = Min(tr2) next ; From File for a = 0 to n1-1 step 1.0 tr3 = File("myMultiBlockSignal.tpc5",0).a minVal3() = Min(tr3) next </pre>
Function	<p>Function is a keyword that is used to define the beginning of a formula block. After the keyword Function a name for the function is expected. Between parenthesis all the parameters can be declared. The number of parameters is not limited. They can be of any type (number, trace, array etc.). EndFunction declares the end of the Function and its formula block.</p> <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin: 10px 0;">  Attention: All parameters are transferred by Reference. This means that they also could (and would) be modified in the main formula environment. </div> <p>The immediate return value must be assigned in the formula block to the name of the Function. It can be any type of variables (number, trace, array etc.). See also the functions Using and Endformula.</p> <p>Example:</p> <pre> ; Example: Function inside the same Formula x = 5 y = 7 z = Addition(x,y) Endformula Function Addition(value0, value1) Addition = value0 + value1 EndFunction </pre>
GetEnvironment(key)	<p>Reads with the parameter key environment variables from the current formula. The return value variable is a number. Parameter key has the following functions:</p> <ul style="list-style-type: none"> 0 = UseMemory (Please see also function UseMemory) 1 = AlwaysCloseFiles 5 = DontSaveHdf 6 = IgnoreDataAvailability 7 = UpdateGUI 8 = IsDebug 9 = IgnoreAbortDuringTryCatch <p>Parameter 1, 6 and 7 are active by default, all others inactive.</p> <p>Example:</p> <pre> ; disable writing files to expr folder SetEnvironment(0, True) ; switch ON </pre>

	<pre>tr1 = Sinus(10,50,0,1E3,1E3) ; Tr1 is saved in RAM SetEnvironment(0, False) ; switch OFF tr2 = tr1 ; Tr2 is saved as TPC5 file ; read changes of function UseMemory() val1 = GetEnvironment(0) UseMemory(True) val2 = GetEnvironment(0) UseMemory(False) val3 = GetEnvironment(0)</pre>
<p>GetVariableValue(name [, useGlobalScope])</p>	<p>Returns the value of the variable with the name name. In case the variable is not available, the returnvalue will be NotDefined.</p> <p>The optional parameter useGlobalScope can be set to access from within a function call the global scope.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> This function is mostly used in combination with the function SetVariableValue.</p> </div> <p>Example:</p> <pre>; create some variables for a = 0 to 10 step 1.0 str = "myvariable_"+a SetVariableValue(str, a^2) next ; read back the variables mylist = List() for a = 0 to 12 step 1.0 str = "myvariable_"+a val = GetVariableValue(str) if not (val = NotDefined) then ; check if valid or not mylist() = val endif next</pre>
<p>If-Else</p>	<p>Creates a decision expression which can be used to decide which case (block1, block2 or block3) should be used for a certain condition. If condition i = b is True the formula processes the if-statement otherwise the else-statement. The condition expression can use the following operators: "=", ">", "<", ">=", "<=", "<>"</p> <p>Multiple conditions can be concatenated with the words and, or and not.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> The operator <> corresponds to not equal (!=)</p> </div> <p>If there is a need for further conditional expressions you can use "elseif i < b then" after the first if-statement. There is no limit how many elseif statements you can write.</p> <p>Example:</p> <pre>if Mx > Biggest then Biggest=Mx else Biggest = 0 endif</pre>

<p>IsDebug()</p>	<p>Checks if Formula is running in Debug Mode. Return value variable is from type Boolean, debug active = True, normal mode = False.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  GetEnvironment(8) returns the same value. </div> <p>Example:</p> <pre>if IsDebug()=1 then ; do some debug stuff endif</pre>
<p>lock-endlock</p>	<p>Blocks other processes in parallel calculations. This feature is used in combination with parallel computational tasks such as the parfor statement. The code block between lock and endlock is then disabled by the other parallel processes so that these other processes have no effect on e.g. have an intermediate result. Please see also parfor and parfor-each.</p> <p>Example:</p> <pre>; Scope recording, maximum block length for this example UseMemory(True) tr = c0A1 maxValParFor = NotDefined tasks = 4 ; number of tasks lengthMeas = Length(tr) ; number of samples dt = TSample(tr) ; time between samples subCount = (lengthMeas / tasks)*dt ; duration of signal for one task t1 = TBegin(tr) ; time at beginning of trace StartWatch() parfor index = 0 to (tasks-1) tStart = t1 + (index * subCount) tStop = tStart + (subCount - dt) maxVal = Max(tr, tStart, tStop) ; In order to access the variable "maxValParFor" of several ; tasks, this must take place between a lock statement. ; Otherwise, it can happen that the tasks overwrite each ; other and thus the correct maximum value in the variable is not ; available. lock if maxValParFor = NotDefined then maxValParFor = maxVal elseif maxValParFor < maxVal then maxValParFor = maxVal endif endlock next tParFor[s] = Stopwatch()/1000 SetUnit(maxValParFor, GetUnit(tr)) ; Single task operation StartWatch() maxValStd = Max(tr) tStandard[s] = Stopwatch()/1000 SetUnit(maxValStd, GetUnit(tr))</pre>

	<code>pause("Single task vs. four tasks", lengthMeas, tStandard, tParFor, maxValStd, maxValParFor)</code>
NotDefined	<p>Keyword. Response on functions from which no valid value could be determined.</p> <p>Example: <code>tx=TCross(trc, -10,10,0.5)</code></p> <pre>If tx = NotDefined Then ; No crossing found Endif</pre>
parfor-each	<p>For more performance of a modern computer system, parfor each allows to split calculations on multiple CPU cores. Parallel analysis of several hardware channels will give a significant decrease of time.</p> <p>Lock and EndLock has to be used for synchronisation with common used resources, in the example below, a Dictionary. Please see also the functions parfor and lock-endlock</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Performance gain will only occur if you have many samples to process per Channel. Just a few samples doesn't give you any benefit. Parallel loops in general should only be used if the work load is big enough. Otherwise there will be a performance hit because changing between parallel tasks is expensive. </div> <p>Here are some general rules to follow for using a parallel loop:</p> <ul style="list-style-type: none"> • Heavy workload is required (many files, many samples, complex algorithms, etc.) • Fetching channel data directly from the local PC is much faster than via Network. • Save data to tpc5 files if possible and process does files to get maximum throughput. <p>Example: <code>UseMemory(True)</code> <code>values = Dictionary()</code></p> <pre>parfor each input in enChannels() desc = GetChannelDesc(input) key = "" + desc(0) + ("A" + desc(1)) + desc(2) tMax = TOfMax(input) tMin = TOfMin(input) traceSlice = Slice(input, tMax, tMin) lowPassTrace = LowPass(traceSlice, Bessel, 2, 1e3) meanVal = Mean(lowPassTrace) Lock values(key) = meanVal EndLock next</pre>
parfor	Similar to function for next . Should be used on multi core CPU system to increase calculation speed.

Parfor next can be used to analyse multiple file in parallel, readout values from its traces and save them to a dictionary.

Please see also the functions **parfor-each** and **lock-endlock**



Performance gain will only occur if you have many samples to process per Channel. Just a few samples doesn't give you any benefit. Parallel loops in general should only be used if the work load is big enough. Otherwise there will be a performance hit because changing between parallel tasks is expensive.

Here are some general rules to follow for using a parallel loop:

- Heavy workload is required (many files, many samples, complex algorithms, etc.)
- Fetching channel data directly from the local PC is much faster than via Network.
- Save data to tpc5 files if possible and process does files to get maximum throughput.

Example:

; Scope recording, maximum block length for this example

UseMemory(True)

tr = c0A1

maxValParFor = NotDefined

tasks = 4 ; number of tasks

lengthMeas = Length(tr) ; number of samples

dt = TSample(tr) ; time between samples

subCount = (lengthMeas / tasks)*dt ; duration of signal for one task

t1 = TBegin(tr) ; time at beginning of trace

StartWatch()

parfor index = 0 to (tasks-1)

 tStart = t1 + (index * subCount)

 tStop = tStart + (subCount -dt)

 maxVal = Max(tr, tStart, tStop)

; In order to access the variable "maxValParFor" of several tasks, this must take place between a lock statement.

; Otherwise, it can happen that the tasks overwrite each other and thus the correct maximum value in the variable is not available.

 lock

 if maxValParFor = NotDefined then

 maxValParFor = maxVal

 elseif maxValParFor < maxVal then

 maxValParFor = maxVal

 endif

 endlock

next

tParFor[s] = Stopwatch()/1000

SetUnit(maxValParFor, GetUnit(tr))

; Single task operation

StartWatch()

maxValStd = Max(tr)

tStandard[s] = Stopwatch()/1000

	<pre>SetUnit(maxValStd, GetUnit(tr)) pause("Single task vs. four tasks", lengthMeas, tStandard, tParFor, maxValStd, maxValParFor)</pre>
<p>ReadOfPrevious(key)</p>	<p>ReadOfPrevious can be used together with WriteForNext as some kind of global variable. key is a string and the name of the variable.</p> <p>The stored values of the variable can be read or written on every new calculation of the formula. Calculation can be done over several measurements.</p> <p>Example:</p> <pre>; Averaging ; get last trace mytrace = ReadValueOfPrevious("mytrace") ; measure and smooth new trace tr = c0A1; tr = Smooth(tr, 100) ; first run? if mytrace = NotDefined then ; yes, first run mytrace = tr else ; no, not the first run mytrace = (mytrace + tr) /2 endif ; store value for next calculation WriteValueForNext("mytrace",mytrace)</pre>
<p>SetEnvironment(key, value)</p>	<p>Sets special parameters for optimizing Formula Editor performance.</p> <p>Parameter key has the following functions:</p> <ul style="list-style-type: none"> • 0 = UseMemory (Please see also function UseMemory) - value => Bool • 1 = AlwaysCloseFiles - value => Bool • 5 = DontSaveHdf - value => Bool • 6 = IgnoreDataAvailability - value => Bool • 7 = UpdateGUI - value => Bool • 8 = IsDebug - value => Bool • 9 = IgnoreAbortDuringTryCatch - value => Bool <p>Parameters 1, 6 and 7 are active by default, all others inactive.</p> <p>Example:</p> <pre>; disable writing files to expr folder SetEnvironment(0, True) ; switch ON tr1 = Sinus(10,50,0,1E3,1E3) ; Tr1 is saved in RAM SetEnvironment(0, False) ; switch OFF tr2 = tr1 ; Tr2 is saved as TPC5 file ; read changes of function UseMemory() val1 = GetEnvironment(0) UseMemory(True) val2 = GetEnvironment(0) UseMemory(False) val3 = GetEnvironment(0)</pre>

SetMessage(text [, useOverlayWindow])	<p>Writes the text information text to the status display of the Formula Editor in the bottom right corner of Tranax. The optional parameter useOverlayWindow as a Boolean allows writing the text either in the status bar (false or not specified) or in the middle of TranAX window (true). Is useOverlayWindow defined as true, the message shows up and slowly fades out again. Is this parameter defined as false, the same as without this optional parameters, the message will be showed in the bottom right corner.</p> <p>Example:</p> <pre> ; message in status bar SetMessage("my message") ; message as window SetMessage("my message", true) </pre>
SetVariableValue(name, value [, useGlobalScope])	<p>Sets or creates a variable named name. If a variable with the same name already exists, it will be set to the new value. The parameter value will be assigned to the variable name. The optional parameter useGlobalScope defines if the value of the variable name should be set in the global scope of the formula.</p> <p>To get the value of a set or defined variable, the function GetVariableValue can be used.</p> <p>Example:</p> <pre> ; create some variables for a = 0 to 10 step 1.0 str = "myvariable_" + a SetVariableValue(str, a^2) next ; read back the variables mylist = List() for a = 0 to 12 step 1.0 str = "myvariable_" + a val = GetVariableValue(str) if not (val = NotDefined) then ; check if valid or not mylist() = val endif next </pre>
SplitString(text, symbol)	<p>Splits the String text with the indicator symbol and creates an array with respective partial strings (the indicator symbol will be removed). Symbol can also consist of multiple indicators. The original String text is then split at the corresponding positions. In case symbol occurs multiple times, it will be converted to multiple partial-Strings in text. The array then will be proportionally longer.</p> <p>Example:</p> <pre> SplitString(text, symbol) </pre>
StopCalculation(filepath)	<p>Stops the calculation of a formula, which was started with the function Calculate. The parameter filepath is the formula file to be calculated with the extension *.for. You can also run formulas from ExperimentSets *.zip. Return value is a number.</p>

	<p>If the formula was stopped, the return value = 1. If the formula was not in the calculation, the return value = 0.</p> <p>Example:</p> <pre>; contains dictionary of results from Formula 2.for myFormula = "Empty 123.zip\Formula 2.for" Calculate(myFormula, false, true) Delay(5) val = StopCalculation(myFormula) EndFormula ; Create a second Formula with the name Formula 2.for ; Copy and paste the code below into this file ; Save file with Ctrl+s cntr = 0 do while cntr < 1000 cntr += 1 Delay(0.1) loop</pre>
<p>StringFormat(a, format)</p>	<p>Converts a into a string. The formatting can be defined with the string format.</p> <div data-bbox="528 913 1345 1055" style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p> Check all the examples how to format date time, time span, numbers, etc.</p> </div> <p>Example:</p> <pre>; Date Time Formats ; ----- dateTimeNow = GetDateTime() dateTime_d = StringFormat(dateTimeNow, "d") ; d -> Short Date -> 10/04/2019 dateTime_DU =StringFormat(dateTimeNow, "D") ; D -> Long Date -> Friday, 04 October 2019 dateTime_t =StringFormat(dateTimeNow, "t") ; t -> Short Time -> 14:20 dateTime_TU =StringFormat(dateTimeNow, "T") ; T -> Long Time -> 14:20:44 dateTime_f =StringFormat(dateTimeNow, "f") ; f -> Long Date Time -> Friday, 04 October 2019 14:20 dateTime_FU =StringFormat(dateTimeNow, "F") ; F -> Long Date Time (seconds) -> Friday, 04 October 2019 14:20:44 dateTime_g =StringFormat(dateTimeNow, "g") ; g -> Short Date Time -> 10/04/2019 14:20 dateTime_GU =StringFormat(dateTimeNow, "G") ; G -> Short Date Time (seconds) 10/04/2019 14:20:44 dateTime_M =StringFormat(dateTimeNow, "M") ; M -> Short Date -> October 04 dateTime_r =StringFormat(dateTimeNow, "r") ; r -> RFC1123 Date Time String -> Fri, 04 Oct 2019 14:20:44 GMT dateTime_s =StringFormat(dateTimeNow, "s") ; s -> Sortable Date Time String -> 2019-10-04T14:20:44 dateTime_u =StringFormat(dateTimeNow, "u") ; u -> Universal Sortable Date -> 2019-10-04 14:20:44Z dateTime_UU =StringFormat(dateTimeNow, "U") ; U -> Universal full date -> Friday, 04 October 2019 04:20:44 dateTime_Y =StringFormat(dateTimeNow, "Y") ; Y -> Year month pattern -> 2019 October dateTime_custom = StringFormat(dateTimeNow, "dd.MM.yyyy HH:mm:ss") ; dd: Days, MM: Month, yyyy: Year, HH: Hours, mm: Minutes, ss: Seconds</pre>

	<pre> ; Number Formats ; ----- number = 7656.456 number_c = StringFormat(number, "c") ; c -> currency -> \$7,656.46 number_e = StringFormat(number, "e3") ; e -> Scientific (number = digits after decimal point) -> 7.656e+003 number_f = StringFormat(number, "f2") ; f -> Fixed Point (number = digits after decimal point) -> 7656.46 number_g = StringFormat(number, "g5") ; g -> General (number = total amount of digits) -> 7656.5 number_n = StringFormat(number, "n4") ; n -> Thousand Separator -> 7,656.4560 ; Custom Formats ; ----- number = 81456.3467 number_0 = StringFormat(number, "00.00") ; 0 -> Zero Placeholder -> 81456.35 number_hash = StringFormat(number, "#.#####") ; 0 -> Digit Placeholder -> 81456.3467 number_point = StringFormat(number, "0.0") ; 0 -> Decimal Point -> 81456.3 number_sep = StringFormat(number, "0,0") ; 0 -> Thousand Separator -> 81,456 number_per = StringFormat(0.758, "0%") ; 0 -> Percent -> 76% </pre>
StringToNumber(text)	<p>Changes the String text in a number. Presumed is, of course, that it concerns a number in a String. Otherwise the response will be NotDefined.</p> <p>Example: StringToNumber(text)</p>
True	<p>Keyword True, corresponds to the value 1. It indicates that a condition is met.</p> <p>Example: if FileExist("xy.tpc5") = True then BLOCK endif</p>
try-catch	<p>Used to catch critical errors.</p> <p>The code part in block is executed and monitored. If an error occurs which would stop the formula with an error message, the code will be executed in errorBlock. Detailed error message can be stored or a counter with the number of further retries are decremented.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Please note: The written code in the formula editor should generally be able to run without try-catch statements! </div> <p>Clean code should be checked for borderline cases, exceptions and, if possible, intercepted. Limits of Arrays, Lists and counters should be able to run without errors even without these try-catch functions. There are exceptions on which one has no influence as a programmer. For example, reading files from a network drive can have many external impacts: Network</p>

interruption, delays in the network, file being blocked by another application, etc. In such cases, the try-catch statement can provide valuable help.

Example:

```
; 1. Useful but more sophisticated example
; copy the latest BDF file from a TraNET FE device
; you may have to connect first with the file Explorer to its
IP address
```

```
noOfRetries=5
t_retry[s]=0.1
fNameDest = "myFile#.bdf"

for nbOfCopyTry=1 to noOfRetries
  fileCopySuccessful=True
  try
    ;Path to the DAQ data directory
    tmp=GetInfo("url",0)
    tmp = SplitString(tmp, ":")
    urlDAQ = tmp(0)
    pathDAQ = PathCombine("\\", urlDAQ, "Measurement Data")
    fNameSource=GetNewestFile(pathDAQ, "*.bdf")
    fNameDestNew = FileNameInc(fNameDest)
    CopyFile(fNameSource, fNameDestNew, false)

  catch errorDict
    ; an error has occurred
    fileCopySuccessful=False
    msg = errorDict("Message")
    srcPath = errorDict("SourcePath")
    dateTime = errorDict("DateTime")
    lineNr = errorDict ("LineNr")
    errorPos = errorDict ("Pos")
    errorLen = errorDict("Length")
    cancelled = errorDict("Cancelled")
    dateTimeString=StringFormat(dateTime,"")
    WriteLine("myLog.txt", Tabulator, msg, srcPath,
dateTimeString, lineNr, errorPos, errorLen, cancelled,
nbOfCopyTry)
  endtry

  if fileCopySuccessful=True then
    exitloop
  endif
  delay(t_retry)
next
```

```
; 2.) Easy to understand, but not so good example
; create an array with 10 entries
myArr = Array(0 to 9) as double
```

```
try
  val1 = myArr(4) ; no Error
  val2 = myArr(12) ; will raise an Error
catch errorDict
  ; define a value for val2
  val2 = NotDefined
  msg = errorDict("Message")
  srcPath = errorDict("SourcePath")
  dateTime = errorDict("DateTime")
  lineNr = errorDict ("LineNr")
  errorPos = errorDict ("Pos")
  errorLen = errorDict("Length")
  cancelled = errorDict("Cancelled")
```

	<pre>dateTimeString=StringFormat(dateTime,"") ; write a line into a Logfile WriteLine("myLog.txt", Tabulator, msg, srcPath, dateTimeString, lineNr, errorPos, errorLen, cancelled) endtry</pre>
UnescapeString(text)	<p>The function UnescapeString simplifies the formatting of texts. The parameter text is a string. Special characters such as tabs or line breaks can be easily inserted into a text. Commonly used escape character are line break \r, \n or tabulator \ t</p> <p>Example:</p> <pre>; \r\n creates a new line str = UnescapeString("1st Row with text\r\n2nd Row with more text") Pause(str) ; Same example without using function "UnescapeString" str = "1st Row with text" + "\r"C + "\n"C + "2nd Row with more text" Pause(str) ; Write a line to text file with tabulator separated str = UnescapeString("1st item\t2nd item\t3rd item") WriteLine("myText.txt", "", str) ; This can also be written like this WriteLine("myText.txt", Tabulator, "1st item", "2nd item", "3rd item")</pre>
Using(filepath)	<p>With the keyword Using a function file (*.fnc) is included into the formula code (See example).</p> <p>Example:</p> <pre>; Create a file "MathOperators.fnc" with the following content: Function Addition(value0, value1) Addition = value0 + value1 EndFunction ; Include the above file "MathOperators.fnc" into a new formula with the following statement: using "MathOperators.fnc" ; The function Addition from "MathOperators.fnc" will be included because ; of the usage of using. result = Addition(2, 3)</pre>
VariableToString(a)	<p>Reads the name of a variable.</p> <p>The parameter a is a defined variable in the Formula Editor. Return value value is of the type String shows the name of this variable.</p> <p>Please see also the functions GetVariableValue and SetVariableValue.</p> <p>Example:</p> <pre>myVariable = 20 name = VariableToString(myVariable) Pause(name) ;myVariable</pre>
While	<p>Creates a loop which runs block as long as the condition i=b is True. The condition expression can use the following operators:</p>

	<p>"=", ">", "<", ">=", "<=", "<>". Multiple conditions can be concatenated with the words and, or and not.</p> <div data-bbox="528 248 1251 389" style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  The operator <> corresponds to not equal (!=) </div> <p>Example:</p> <pre>i = 100 do while i > 15 i = i - 1 loop</pre>
<p>WriteForNext(key, value)</p>	<p>WriteForNext can be used together with ReadOfPrevious as some kind of global storage. The variables will be available for the next calculation. The parameter value will be stored under the name of key.</p> <p>For further information and example, please see the function ReadOfPrevious.</p> <p>Example:</p> <pre>; Averaging ; get last trace mytrace = ReadOfPrevious("mytrace") ; measure and smooth new trace tr = c0A1; tr = Smooth(tr, 100) ; first run? if mytrace = NotDefined then ; yes, first run mytrace = tr else ; no, not the first run mytrace = (mytrace + tr) /2 endif ; store value for next calculation WriteForNext("mytrace", mytrace)</pre>

34.15 Report Generator

XICenter	<p>Keywords alignment in function XISetHeader and XISetFooter</p> <p>Example: <code>XISetHeader("report.xlsx", XICenter, "Report")</code></p>
XIChangeColumn(cell, number)	<p>Adds a given quantity number to the defined cell cell. This influences the column of cell. cell must be string (e.g. "B5").</p> <p>Example: <code>cell = "B2"</code> <code>newCell = XIChangeColumn(cell, 2) ; "D2"</code></p>
XIChangeRow(cell, number)	<p>Adds a given quantity number to the defined cell cell. This influences the row of cell. cell must be string (e.g. "B5").</p> <p>Example: <code>cell = "B2"</code> <code>newCell = XIChangeRow(cell, 2) ; "B4"</code></p>
XICreateEmptyFile()	<p>Creates an empty Excel file and provides a random name with file extension .xlsx. This name must be applied in successive formulas in order to be able to work with the worksheet. By default, this file is saved in the folder "data" of the actual experiment.</p> <p>This function is recommended when fast storage of computed values into an Excel worksheet is required. It is not depending on prior created templates, but can afterwards be further processed with Excel. When this function is called, XIOpenFile can be omitted.</p> <p>Example: <code>fnetwork = XICreateEmptyFile() ; creates a random pattern file name</code> <code>XISetCellValue(fnetwork, "A10", 7.123) ; write some data</code></p> <p><code>fnameFinal = "MyReport_#.xlsx" ; define a file name, # will be replace by an increasing number</code> <code>fnameFinal = FileNameInc(fnameFinal) ; increase counter and replace #</code> <code>XISave(fnetwork, fnameFinal) ; save the created ExcelSheet into the final File.</code></p>
XIGetCellReferences(filepath, searchKey [, all])	<p>This function searches in the entire Excel-file filepath for the phrase searchKey and returns the cell designation****(e.g. B5). These parameters must be string types. Keywords (searchKey) must be logged on beforehand in the Excel file (template). In case searchKey isnt found the prompt n.def is returned. It is recommended to choose keywords that otherwise are not used often (e.g. %V1). No distinction is being made between upper- and lower-case characters. In case the optional parameter indicates all = True an array respectively a list is returned.</p>

	<p>In those all cell notations are stored where the phrase searchKey has been found.</p> <p>Example: <code>fn\$ = ".\Excel_Templates\Template_1.xlsx"</code> <code>fnam\$ = XlOpenFile(fn\$)</code> <code>cellRef = XlGetCellReferences(fnam\$, "%x")</code></p>
<p>XlGetCellValue(filepath, cell)</p>	<p>Returns the value in a cell. Usually cell means a single cell (B3). cell must be of String type.</p> <p>Depending on what is present in the cell it returns a string or a number. However, a cell range also can be defined (e.g. A1:A4 or "A1:D1"). The cell range only be defined as in the same column or line (e.g. not "A1:C3"). Also, on beforehand a corresponding array or list must be made available. That array or list then will be filled with the string or number values of the cells.</p> <p>Arrays must be set up as String-Array or Double-Array, depending whether strings or numbers are expected in the cells. In case a cell is empty a n.def or an empty string ("") is returned. When a list is set up, its content eventually shall be strings or numbers.</p> <p>Example: <code>fn\$ = ".\Excel_Templates\Template_1.xlsx"</code> <code>fnam\$ = XlOpenFile(fn\$)</code> <code>v1 = XlGetCellValue(fnam\$, "A1") ; Einzelwert</code> <code>valArr\$ = Array(0 to 3) as String</code> <code>valArr\$ = XlGetCellValue(fnam\$, "A2:A5")</code> <code>valArrD = Array(0 to 3) as Double</code> <code>valArrD = XlGetCellValue(fnam\$, "B2:E2")</code> <code>valArrL = List()</code> <code>valArrL = XlGetCellValue(fnam\$, "A2:E2")</code></p>
<p>XlInitializeCOM()</p>	<p>XlInitializeCOM must have a valid version of Excel as of 2007. A report file is opened which can also be edited directly during or after the measurement with Excel. XlInitializeXML does not require the installation of Excel. The generated report is formed by an XML file (*.xlsx file). This can then be reused and edited with Excel later.</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;">  <p>If none of the above function (XlInitializeCOM/XlInitializeXML) is called the default will be XML Excel.</p> </div>
<p>XlInitializeXML()</p>	<p>XlInitializeCOM must have a valid version of EXCEL as of 2007. A report file is opened which can also be edited directly during or after the measurement with EXCEL. XlInitializeXML does not require the installation of EXCEL. The generated report is formed by an XML file (*.xlsx file). This can then be reused and edited with EXCEL later.</p>

	 If none of the above function (XlInitializeCOM/XlInitializeXML) is called the default will be XML Excel.
XlInsertImage(filepath, name, cell [, width [, height]])	<p>Inserts the screen contents in the waveform name at the specified position cell in the EXCEL file filepath. This can be the name of an individual waveform display, scalar table, whole page or an image file (including path information). cel of type string is typically a single cell (e.g., "B2"). It determines the upper left corner of the picture to be inserted. For cell, a range can be specified (for example, "A4: F30"). The cell area determines the location and the size of the inserted image.</p> <p>Optionally, the parameters width and height of the inserted image can be defined. If only the width is specified, then the height is proportionally determined (same aspect ratio as in the original image). width and height must be specified in number of pixels and of the type integer. If a cell range is specified (not just a single cell), width and height are ignored.</p> <p>The settings under "Tools / Settings / User Interface / Snapshot / White Background" have an influence on the inserted graphic.</p> <p>Example: fn\$ = ".\Excel_Templates\Template.xlsx" fnam\$ = XlOpenFile(fn\$) XlInsertImage(fnam\$, "Waveform 1", "A4:E20") XlInsertImage(fnam\$, "Skalar_A 1", "A21", 100, 300)</p>
XlLeft	<p>Keywords alignment in function XlSetHeader and XlSetFooter</p> <p>Example: XlSetHeader("report.xlsx", XlLeft, "Report")</p>
XlOpenFile(filepath)	<p>Opens an existing Excel file with the name filepath (normally a template). It returns the file name completed with the absolute file path. These has to be used for additional formula-based Excel report generator functions.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Note: If Excel is initialized with XlInitializeXML, only *.xlsx files can be processed. With XlInitializeCOM also older Excel file formats can be processed. </div> <p>Example: XlInitializeXML() fn\$ = ".\Excel_Templates\Template_1.xlsx" filename\$ = XlOpenFile(fn\$) XlSetCellValue(filename\$, "A10", 7)</p>
XlRight	<p>Keywords alignment in function XlSetHeader and XlSetFooter</p> <p>Example: XlSetHeader("report.xlsx", XlRight, "Report")</p>

XISave(filepath [, newFilepath])	<p>Saves the open Excel file filepath including all changes. For the second parameter newFilepath another name (incl. path) must be chosen to prevent overwriting the template. The Excel file stays open as long as the CloseFile command is not executed.</p> <p>Example: <code>XIInitializeXML ()</code> <code>fn\$ = ".\Excel_Templates\Template_1.xlsx"</code> <code>fnam\$ = XIOpenFile(fn\$)</code> <code>fnRep\$ = ".\Excel_Reports\Report_#.xlsx"</code> <code>; Block with value insertions in EXECL Sheet.</code> <code>XISave (fn\$, fnRep\$) ; Saves the EXCEL file (incl. all insertions) in the Report File.</code></p>
XISelectSheet(filepath, sheetname)	<p>Selects in the Excel-file filepath the worksheet with the name sheetname. If this command fails, by default the first sheet will be selected.</p> <p>Example: <code>fn\$ = ".\Excel_Templates\Template_1.xlsx"</code> <code>filename\$ = XIOpenFile(fn\$)</code> <code>XISelectSheet(filename\$, "Tabelle2")</code></p>
XISetCellValue(filepath, cellOrKey, value)	<p>Set the value value in cell cellOrKey in the Excel file filepath. Instead of a cell (for example "B2"), a key cellOrKey can also be specified (this can be omitted XIGetCellReferences).</p> <p>If value is an array, list, or curve, the values are inserted by default in the column of the specified cell cellOrKey. If you want to insert the values into a line, the cell must be defined as follows: "A7: row". Curves (as well as arrays or lists) must not be too long for transmission in Excel. The maximum length (number of samples, 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 line: 16'385 - Column of the first value (e.g., <= 116'382 if cell = "Cx: row"). Too long curves (large block lengths) can be shortened beforehand with the functions Slice, Resampling or Skip.</p> <p>Example: <code>filename = "excel\Template.xlsx"</code> <code>filename = XIOpenFile(filename)</code> <code>colArr = Array(1,2,3,4)</code> <code>XISetCellValue(filename, "A1", colArr)</code> <code>XISetCellValue(filename, "A1:row", colArr)</code></p>
XISetFooter(filepath, alignmentOrKey, value)	<p>Sets the value value in the footer line of the Excel file filepath. The parameter alignmentOrKey determines the position at which value should be inserted. This can be a keyword XILeft, XICenter, XIRight (for Left, Center, Right) or a search term. A search term must first have been inserted in the header or footer line of the Excel template.</p> <p>Example: <code>Fn\$ = "excel\Template.xlsx"</code></p>

	<pre> Fn\$ = X1OpenFile(Fn\$) Skey\$="%Hd-1" ; Should be set in Template X1SetHeader(Fn\$, X1Left, "Report") X1SetHeader(Fn\$, Skey\$, "Unit Test 1") X1SetFooter(Fn\$, X1Center, "Result Table") </pre>
<p>X1SetHeader(filepath, alignmentOrKey, value)</p>	<p>Sets the value value in the header line of the Excel file filepath . The parameter alignmentOrKey determines the position at which value should be inserted. This can be a keyword X1Left, X1Center, X1Right (for Left, Center, Right) or a search term. A search term must first have been inserted in the header or footer line of the Excel template.</p> <p>Example:</p> <pre> Fn\$ = "excel\Template.xlsx" Fn\$ = X1OpenFile(Fn\$) Skey\$="%Hd-1" ; Should be set in Template X1SetHeader(Fn\$, X1Left, "Report") X1SetHeader(Fn\$, Skey\$, "Unit Test 1") X1SetFooter(Fn\$, X1Center, "Result Table") </pre>

34.16 Signal Analysis

<p>a(index*)</p>	<p>Gets the value at position index of a. Multiple indices are used for multidimensional arrays.</p> <div data-bbox="528 304 1236 443" style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  If a is a curve then index is a time in seconds. </div> <p>Example: <code>tr1[V] = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude (20Vpp), 50Hz</code> <code>t1 = 0.022</code> <code>val1[V] = tr1(t1) ; val1=5.8779</code></p> <p><code>SetCrs("Waveform 1", "A", t1); set cursor A to position of t1</code></p>
<p>Aic(a [, start, end])</p>	<p>Used for Acoustic Emission (AE) application. Algorithm for automatic detection of AE first motion based on AIC picker.</p> <p>AIC definition according Wikipedia EN: The Akaike information criterion (AIC) is an estimator of the relative quality of statistical models for a given set of data a. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Thus, AIC provides a means for model selection. AIC is founded on information theory: it offers an estimate of the relative information lost when a given model is used to represent the process that generated the data. (In doing so, it deals with the trade-off between the goodness of fit of the model and the simplicity of the model.)</p> <p>The parameters start and end can be used to limit the calculation to a certain length.</p> <p>Example:</p> <pre> ; AIC Example tr = GenerateSignal() SamplingRate = 1/TSample(tr) triggerTime = TTrigger(tr) - TBegin(tr) ; generate an AIC trace ; the AIC array has to be converted back to a trace trAic = ConvToTrace(Aic(ConvToArray(tr)), SamplingRate) ; set trigger time according source trace trAic = SetTrigger(trAic, triggerTime) ; search minimum of AIC Trace and set cursor to its position tmin = TOfMin(trAic) SetCrs("Waveform 2", "A", tmin) ; create a test signal Function GenerateSignal() tr1 = Noise(2, 1E6, 1E3) tr2 = tr1 * 2 trNew = Merge(tr1, tr2) t1 = TBegin(trNew) t2 = TEnd(trNew) ; place trigger to the last quarter of the signal trNew = SetTrigger(trNew, (t2-t1)*0.75) </pre>

	<p>GenerateSignal = trNew EndFunction</p>
Area(a [, start, end])	<p>Calculates the area of a relative to zero level. Values above zero contribute positively to the area, values below zero negatively. Optional parameters are start and end Similar to previous slicing of a trace, defines start and end time of the calculation.</p> <p>Please see also the function Int for results as a trace.</p> <p>Example:</p> <pre>; create a signal v[m/s] with 10[m/s^2] acceleration v[m/s] = Ramp(10,1E3,10E3) a[m/s^2] = diff(v) s_trace = Int(v) ; creates a trace s_num = Area(v) ; numeric result</pre>
DutyCycle(a [, start, end [, level, hysteresis]])	<p>Calculates the ratio of positive pulse width to the period duration of a. The return value is a number. The optional parameters start and end are time marks, between this two, the duty cycle will be calculated. Additional optional parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is 5% of the peak value.</p> <p>Example:</p> <pre>tr = sigRectangle() ; create a rectangle signal dutC = DutyCycle(tr) ; dutC = 25.275 pwNeg = PulseWidthNeg(tr) ; pwNeg = 136us pwPos = PulseWidthPos(tr) ; pwPos = 46us</pre> <p>EndFormula</p> <p>Function sigRectangle()</p> <pre>; Create a rectangle signal with a duty cycle of 25% periodCount = 10 ; 3 periods fs = 1e6 ; 1 MHz sampling rate freqRect[Hz] = 5.5e3 ; 5.5 kHz signal frequency dt = 1/freqRect peak[V] = 3.3 ; 3.3V peak value dutyCyclePer[%] = 25 ; 25% duty cycle listPair = List() listPair() = 0 listPair() = peak listPair() = dt * dutyCyclePer / 100 listPair() = peak listPair() = listPair(2) + 1/fs listPair() = 0 listPair() = listPair(4) + dt * (100 - dutyCyclePer) / 100 - 1/fs listPair() = 0 sigRectangle = CreateSignal(periodCount, fs, listPair)</pre> <p>EndFunction</p>
Energy(a [, start, end])	<p>Calculates the integral of the squared samples of a. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <p>Example:</p> <pre>trace = c0A1 E = Energy(trace)</pre>

<p>FindEvent(a, type, start, end, level, hysteresis)</p>	<p>The function can be used to detect slopes in a. The parameter type is a number (0 ... 2) and defines the searching for rising or falling slopes:</p> <ul style="list-style-type: none"> • Positive = 0 • Negative = 1 • both = 2 <p>All other values result in an error in the formula.</p> <p>The parameters start and end are time marks. Between this two, the slopes (events) will be detected. The Parameter level defines the search value; hysteresis can be used for to better detection in noisy signals. Generally, hysteresis should not be set to zero.</p> <div data-bbox="528 745 1236 887" style="background-color: #f0f0f0; padding: 5px;">  Findevent is an enhanced function of TCross. </div> <p>Example:</p> <pre>trace = Sinus(10, 50E-3, 0, 1E3, 30E3) t1 = TBegin(trace) t2 = tEnd(Trace) ; Slopes: 0=positive, 1=negative, 2=both type = 0 tEvent1 = FindEvent(trace, type, t1, t2, 2, 0.1) ; tEvent1 = 0.64094 SetCrs("Waveform 1", "A", tEvent1) ; Slopes: 0=positive, 1=negative, 2=both type = 1 tEvent2 = FindEvent(trace, type, t1, t2, 2, 0.1) ; tEvent2 = 9.3591 SetCrs("Waveform 1", "B", tEvent2)</pre>
<p>Freq(a [, level, hysteresis])</p>	<p>The parameter a must be a periodic curve. The function Freq calculates the mean frequency from the zero crossings of the curve.</p> <p>The result variable is a number in Hz. With the optional parameters level and hysteresis, the zero line and the hysteresis can be adjusted.</p> <div data-bbox="528 1653 1345 1794" style="background-color: #f0f0f0; padding: 5px;">  This function is still available because of the backward compatibility. Please use function frequency() instead. </div> <p>Example:</p> <pre>; create a signal tr = Sinus(10,50,0,1E3,1E3) tr = tr + Noise(2,tr) ; calculate frequency f = Freq(tr) ; f = 50Hz</pre>
<p>Frequency(a [, start, end [, level, hysteresis]])</p>	<p>Calculates the frequency of a. The return value is a number. To obtain useful results, the signal a should be periodic.</p>

	<p>The parameters start and end are time marks. Between this two, the frequency will be detected. Other optional parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is 5% of the peak value.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  This function is similar to the function Freq. The function Freq is still available because of the backward compatibility. </div> <p>Example:</p> <pre> ; create a signal tr = Sinus(10,50,0,1E3,1E3) tr = tr + Noise(2,tr) ; define start and end t1 = GetCrs("Waveform 1","A") t2 = GetCrs("Waveform 1","B") ; calculate frequency f = Frequency(tr,t1, t2) ; f = approx. 50Hz </pre>
<p>GetBlockIndexes(a, axisMode, start, end)</p>	<p>The function GetBlockIndexes returns all available block numbers between start and end. Return value variable is an array which contains the start and end block. The parameter axisMode defines how the values of start and end are to be interpreted. Possible values are:</p> <ul style="list-style-type: none"> • 0 = Relative Time since Start • 1 = absolute time • 2 = relative time (zero at trigger) • 3 = samples <p>The two parameters start and end are either the time stamps in seconds, an absolute time of the type DateTime, or the position in samples.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  This function makes most sense if you analyze multiple blocks in a. </div> <p>Example:</p> <pre> ; multiblock recording, approx. 20 blocks for this example ; place cursors somewhere within the first and the last block UseMemory(True) tr = c0A1 ; read position of cursors t1 = GetCrs("Waveform 1","A") t2 = GetCrs("Waveform 1","B") ; get first and last selected block crsBlocks = GetBlockIndexes(tr,0,t1,t2) ; make sure position 0 is the smaller block number if crsBlocks(0) > crsBlocks(1) then tmp = crsBlocks(0) crsBlocks(0) = crsBlocks(1) crsBlocks(1) = tmp endif </pre>

	<pre> averagedFFTCuve = NotDefined SetReference("ref1",true,tr) ; workaround to read one block for blkIndex = crsBlocks(0) to crsBlocks(1) blk = ref1.blkIndex ;read one block ; calculate averaged FFT of selected blocks traceFFT = FFT(blk) if averagedFFTCuve = NotDefined then averagedFFTCuve = ConvToSpectrumTrace(traceFFT) else averagedFFTCuve = (ConvToSpectrumTrace(traceFFT) + averagedFFTCuve * blkIndex) / (blkIndex + 1) endif next </pre>
<p>GetDate([a])</p>	<p>The function GetDate returns the date and time from the computer. If the optional parameter a is specified, the date and time of a will be returned. The return value variable is a number of type double that contains date and time information.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> See example how to extract the date and time from the returned number.</p> </div> <p>Example:</p> <pre> date = GetDate(c0A1) date = date / 10E3 year = Integer(date) date = date - year date = date * 100 month = Integer(date) date = date - month date = date * 100 day = Integer(date) date = date - day date = date * 100 hour = Integer(date) date = date - hour date = date * 100 minute = Integer(date) date = date - minute date = date * 100 sec = Integer(date) </pre>
<p>GetDateTime([a]) GetDateTime(text, format)</p>	<p>Returns the current date and time. The optional parameter a will return the trigger time of the curve.</p> <p>Converts a date string text with date and time information into a DateTime variable. The parameter format defines how to interpret the date string text.</p> <p>Example:</p> <pre> computerDateTime = GetDateTime() traceDateTime = GetDateTime(c0A1) dateTime3 = GetDateTime("12.01.2017 13:15:00", "MM.dd.yyyy HH:mm:ss") </pre>

<p>GetNrOfSamples(a, start, end)</p>	<p>The function returns number of samples of a between the time marks start and end which are in seconds.</p> <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin: 10px 0;">  If t1 < t0, a negative value is returned. </div> <p>Example:</p> <pre> ; Read the number of samples from a trace tr = c0A1 t0 = TBegin(tr) t1 = TEnd(tr) n1 = GetNrOfSamples(tr, t0, t1) ; use the position of Cursors t0 = GetCrs("Waveform 1", "A") t1 = GetCrs("Waveform 1", "B") n2 = Abs(GetNrOfSamples(tr, t0, t1)) </pre>
<p>GetTimeSpan(seconds [, minutes [, hours [, days]])</p>	<p>Converts the parameters seconds, minutes, hours days into a TimeSpan. A TimeSpan variable is just an amount of time. The parameter seconds is mandatory, the default parameters are optional.</p> <p>Example:</p> <pre> tsp1 = GetTimeSpan(100) ; 00:01:40 -> 1min, 40sec tsp2 = GetTimeSpan(100, 4) ; 00:05:40 -> 5min, 40sec tsp3 = GetTimeSpan(86400) ; 1.00:00:00 -> 1day </pre>
<p>GetTimeValues(a)</p>	<p>Converts the parameter a into a dictionary which gives the possibility to easily access the date or time information.</p> <p>String keys of the Dictionary for DateTime: "day", "month", "year", "second", "minute", "hour"</p> <p>String keys of the Dictionary für TimeSpan: "seconds", "minutes", "hours", "days", "totalMilliseconds", "totalSeconds", "totalMinutes", "totalHours", "totalDays"</p> <p>Example:</p> <pre> ; DateTime varibale t1 = GetDateTime() dictDateTime = GetTimeValues(t1) year = dictDateTime("year") Delay(1) ; TimeSpan variable tDelta = GetDateTime()-t1 dictTimeSpan = GetTimeValues(tDelta) milis = dictTimeSpan("totalMilliseconds") </pre>
<p>GetUnit(a)</p>	<p>Returns the name of the physical unit of a. Return value variable is of type string.</p> <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin: 10px 0;">  Please see also the function SetUnit. </div> <p>Example:</p>

	<pre> ; create a signal tr = Sinus(10,50,0,10E3,1E3) tr[Bar] = tr + Noise(2,tr) ; filtering keeps the unit tr2 = LowPass(tr,Bessel,4,100) ; calculation removes the unit tr3 = tr2 *1 ; restore origin unit Unit = GetUnit(tr) SetUnit(tr3, Unit) </pre>
Max(a [, start, end])	<p>The function searches the largest value of a. The optional parameter start and end can limit the search range.</p> <p>The result variable is a number.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  With the use of start and end, data can be reduced and the calculation can be accelerated. </div> <p>Example:</p> <pre> ; example with trace tr = Sinus(10,50,0,1E3,1E3) tr = tr + Noise(2,tr) trMax = Max(tr) trMin = Min(tr) ; example with list myList = List() myList() = 3 myList() = -2 myList() = 8 myList() = -10 listMax = Max(myList) listMin = Min(myList) </pre>
Mean(a)	<p>The function forms the arithmetic mean of a. The result variable is a number.</p> <p>Example:</p> <pre> ; create a sinewave signal tr = Sinus(10,50,0,1E3,1E3) tr += 5 trmean = Mean(tr) ; trmean = 5 </pre>
MeanP(a)	<p>The function forms the arithmetic mean of a. Curve a should be periodical. The result variable is a number.</p> <p>Example:</p> <pre> fs = 1e6 offset = 1.55 noiseTrace = Noise(0.01, fs, 1e4) sineTrace = Sinus(1, 1e3, 0, fs, 1e4) + noiseTrace + offset ; Evaluate the given offset evaluatedOffset = MeanP(sineTrace) </pre>

Min(a [, start, end])	<p>The function searches the smallest value of a. The result variable is a number. The optional parameter start and end can limit the search range.</p> <p>Example:</p> <pre> ; example with trace tr = Sinus(10,50,0,1E3,1E3) tr = tr + Noise(2,tr) trMax = Max(tr) trMin = Min(tr) ; example with list myList = List() myList() = 3 myList() = -2 myList() = 8 myList() = -10 listMax = Max(myList) listMin = Min(myList) </pre>
Overflow(a [, start, end])	<p>Indicates an overflow of the ADC in the signal a.</p> <p>The optional parameters start and end are time marks in seconds. Between this two, the overflow will be detected.</p> <p>The return value is undefined (NaN) if no overflow has been found. Otherwise it will return the time were the overflow begins.</p> <p>Example:</p> <pre> tr = c0A1 of = OverFlow(tr) uf = UnderFlow(tr) if (of = NotDefined)and(uf = NotDefined) then SetMessage("Signals within Range", true) else Pause("Check Range settings!") endif </pre>
OverShotNeg(a [, start, end])	<p>Returns the value in % of an undershot (negative overshoot) of a.</p> <p>The optional parameters start and end are time marks in seconds. Between this two, the undershot will be detected.</p> <p>Return value is the undershot in percentage. Analysis of periodic signals may lead to unusable or undefined results (NaN).</p> <p>Example:</p> <pre> trace = c0A1 t1 = GetCrs("Waveform 1","A") t2 = GetCrs("Waveform 1","B") oShot = OvershotNeg(trace,t2, t1) if oShot <> NotDefined then ; your code here endif </pre>
OvershotPos(a [, start, end])	<p>Returns the value in % of an overshoot (positive overshoot) of a.</p>

	<p>The optional parameters start and end are time marks in seconds. Between this two, the overshoot will be detected.</p> <p>Return value is the undershoot in percentage. Analysis of periodic signals may lead to unusable or undefined results (NaN).</p> <p>Example:</p> <pre>trace = c0A1 t1 = GetCrs("Waveform 1", "A") t2 = GetCrs("Waveform 1", "B") oShot = OvershotPos(trace, t2, t1) if oShot <> NotDefined then ; your code here endif</pre>
<p>PeakPeak(a [, start, end])</p>	<p>Calculates the difference between maximum and minimum of a.</p> <p>The optional parameters start and end are time marks in seconds. Between this two the maximum and minimum will be detected.</p> <p>Example:</p> <pre>tr = Sinus(10, 50, 0, 10E3, 1E3) pp1 = PeakPeak(tr) ; pp1 = 20 ; PeakPeak() = Max() - Min() pp2 = Max(tr) - Min(tr) ; pp2 = 20</pre>
<p>Period(a [, start, end [, level, hysteresis]])</p>	<p>Searches on the level of the curve a. The passages and calculates the average period. The return value is the period in seconds.</p> <p>The optional parameters start and end are time marks in seconds. Between this two, the frequency will be detected. Other optional parameters are level and hysteresis. If they are not defined, level is set at the average of the peak value. Hysteresis is 5% of the peak value.</p> <p>Example:</p> <pre>tr = Sinus(10, 50, 0, 10E3, 1E3) tper = Period(tr) ; tper = 20ms</pre>
<p>PulseWidthNeg(a [, start, end])</p>	<p>Calculates the negative pulse width at the level of the baseline (mean of peak to peak; maximum - minimum divided by 2) of a. If multiple periods are detected, the average pulse width will be calculated. Return value is a number.</p> <p>The optional parameters start and end are time marks. Between this two, the pulse width will be detected.</p> <p>Example:</p> <pre>tr = sigRectangle() ; create a rectangle signal dutC = DutyCycle(tr) ; dutC = 25.275 pwNeg = PulseWidthNeg(tr) ; pwNeg = 136us</pre>

	<pre> pwPos = PulseWidthPos(tr) ; pwPos = 46us EndFormula Function sigRectangle() ; Create a rectangle signal with a duty cycle of 25% periodCount = 10 ; 3 periods fs = 1e6 ; 1 MHz sampling rate freqRect[Hz] = 5.5e3 ; 5.5 kHz signal frequency dt = 1/freqRect peak[V] = 3.3 ; 3.3V peak value dutyCyclePer[%] = 25 ; 25% duty cycle listPair = List() listPair() = 0 listPair() = peak listPair() = dt * dutyCyclePer / 100 listPair() = peak listPair() = listPair(2) + 1/fs listPair() = 0 listPair() = listPair(4) + dt * (100 - dutyCyclePer) / 100 - 1/fs listPair() = 0 sigRectangle = CreateSignal(periodCount, fs, listPair) EndFunction </pre>
<p>PulseWidthPos(a [, start, end])</p>	<p>Calculates the positive pulse width at the level of the baseline (mean of peak to peak; maximum - minimum divided by 2) of a. If multiple periods are detected, the average pulse width will be calculated. Return value is a number.</p> <p>The optional parameters start and end are time marks. Between this two, the pulse width will be detected.</p> <p>Example:</p> <pre> tr = sigRectangle() ; create a rectangle signal dutC = DutyCycle(tr) ; dutC = 25.275 pwNeg = PulseWidthNeg(tr) ; pwNeg = 136us pwPos = PulseWidthPos(tr) ; pwPos = 46us EndFormula Function sigRectangle() ; Create a rectangle signal with a duty cycle of 25% periodCount = 10 ; 3 periods fs = 1e6 ; 1 MHz sampling rate freqRect[Hz] = 5.5e3 ; 5.5 kHz signal frequency dt = 1/freqRect peak[V] = 3.3 ; 3.3V peak value dutyCyclePer[%] = 25 ; 25% duty cycle listPair = List() listPair() = 0 listPair() = peak listPair() = dt * dutyCyclePer / 100 listPair() = peak listPair() = listPair(2) + 1/fs listPair() = 0 listPair() = listPair(4) + dt * (100 - dutyCyclePer) / 100 - 1/fs listPair() = 0 sigRectangle = CreateSignal(periodCount, fs, listPair) EndFunction </pre>

RectifiedMean(a [, start, end])	<p>Calculates the rectified average value of the signal a. Return value is a number. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <p>Example: <code>tr1 = Sinus(100, 50, 0, 1E3, 1E3) ; 10V amplitude -> 20Vpp</code> <code>vrms = RMS(tr1) ; 70.711V</code> <code>vrec = RectifiedMean(tr1) ; 63.138V</code> <code>; Ration between RMS and Rectified Mean is defined as approx. 1.1</code> <code>ratio = vrms/vrec ; 1.1199</code></p>
RectifiedMeanPeriodic(a [, start, end])	<p>Calculates the rectified average value of the signal a. Return value is a number. This function searches for the zero crossings of the baseline and calculates the rectified mean value of whole periods. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <p>Example: <code>tr1 = Sinus(100, 50, 0, 1E3, 1E3) ; 10V amplitude -> 20Vpp</code> <code>vrms = RMSP(tr1) ; 70.675</code> <code>vrec = RectifiedMeanPeriodic(tr1) ; 63.073</code> <code>; Ration between RMS and Rectified Mean is defined as approx. 1.1</code> <code>ratio = vrms/vrec ; 1.1205</code></p>
RiseFallTime(a, startLevel, errorLevel [, start, end])	<p>Determines the rise or fall time of a at the start level startLevel and between the error level errorLevel. The parameters startLevel and errorLevel are both in percentage. The return value is a number that represents the rise time in seconds. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <p>Example: <code>tr = Ramp(0,10,1E3,1E3)</code> <code>t1 = RiseFallTime(tr,10,90) ; t1 = 0.8s</code> <code>s1 = Slope(tr) ; s1 = 10.01</code></p>
RMS(a [, start, end])	<p>Calculates the RMS value of a. This function does not pay any attention to whole periods. The result variable is a number. The two optional parameters start and end are time values, within which RMS is calculated.</p> <p>Example: <code>tr = Sinus(10,50,0,1E3,1E3)</code> <code>t1 = GetCrS("Waveform 1","A")</code> <code>t2 = GetCrS("Waveform 1","B")</code> <code>valRMS = RMS(tr, t1, t2) ; exact between t1 and t2</code> <code>valRMSP = RMSP(tr, t1, t2) ; check for whole periods between t1 and t2</code></p>
RMSP(a [, start, end])	<p>Calculates the RMS value over one or more whole periods of a. The result variable is a number. The two optional parameters start and end are time values, within which RMSP is calculated. Baseline we automatically detected.</p>

	<p>Example:</p> <pre>tr = Sinus(10,50,0,1E3,1E3) t1 = GetCrs("Waveform 1","A") t2 = GetCrs("Waveform 1","B") valRMS = RMS(tr, t1, t2) ; exact between t1 and t2 valRMSP = RMSP(tr, t1, t2) ; check for whole periods between t1 and t2</pre>
<p>SetDate(a, b)</p>	<p>The function sets the current computer date and time of a. The optional parameter b can be used to set date and time from a curve or number which represents the date and time. The result variable is a curve again.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  See GetDate example to find out how a number represents the date and time. </div> <p>Example:</p> <pre>; crate a trace, wait some seconds and create a second trace tr1 = Sinus(10,50,0,1E3,1E3) Delay(2) tr2 = Sinus(12,50,0,1E3,1E3) ; calculate time difference deltatime1 = GetDate(tr2) - GetDate(tr1) ; adjust date of tr2 to the one of tr1 tr2 = SetDate(tr2, tr1) ; difference will be zero now deltatime2 = GetDate(tr2) - GetDate(tr1)</pre>
<p>SetDateTime(a, date)</p>	<p>Sets a new date and time to a.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Use function GetDateTime to acquire a date and time from the computer or a curve. </div> <p>Example:</p> <pre>dateTimeNow = GetDateTime() signal = c0A1 signal = SetDateTime(signal, dateTimeNow)</pre>
<p>SettlingTime(a, startLevel, endLevel [, start, end])</p>	<p>Specifies the time until the curve trace remains within an error band.</p> <p>Begin value (0%) is the beginning of the curve, of the final value (100%), the end of the curve.</p> <p>The error band refers to the final value. startlevel and endlevel are values in percent. The error band is defined as $\pm(\text{final value} - \text{endlevel})$. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <p>Example:</p> <pre>trace = c0A1 tSettling = SettlingTime(trace, 10, 90)</pre>
<p>SetUnit(a, unit)</p>	<p>Sets the physical unit of a to the name of the parameter unit.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Please see also the function GetUnit. </div>

	<p>Example:</p> <pre> ; create a signal tr = Sinus(10,50,0,10E3,1E3) tr[Bar] = tr + Noise(2,tr) ; filtering keeps the unit tr2 = LowPass(tr,Bessel,4,100) ; calculation removes the unit tr3 = tr2 *1 ; restore origin unit Unit = GetUnit(tr) SetUnit(tr3, Unit) </pre>
<p>Slope(a [, start, end])</p>	<p>Calculates the slope of a. Slope is defined as dy/dx. The optional parameters start and end are time marks. Between this two, the calculation will be done. Return value is number in [unit/second].</p> <p>Example:</p> <pre> tr = Ramp(0,10,1E3,1E3) t1 = RiseFallTime(tr,10,90) ; t1 = 0.8s s1 = Slope(tr) ; s1 = 10.01 </pre>
<p>SlopeLinearRegression(a [, start, end])</p>	<p>Calculates a linear regression of a, which will be used afterwards to calculate the slope. The optional parameters start and end are time marks. Between this two, the calculation will be done. Return value is a number in [unit/second].</p> <p>Example:</p> <pre> trace = c0A1 s1 = SlopeLinearRegression(trace) </pre>
<p>StdDevPeriodic(a [, start, end])</p>	<p>Detects level crossings of periods on the base line of a. Calculates the standard deviation of entire periods. The optional parameters start and end are time marks. Between this two, the calculation will be done. Return value variable is a number.</p> <p>Example:</p> <pre> trace = c0A1 dev = StdDevPeriodic(trace) </pre>
<p>TBegin(a)</p>	<p>The function returns the time (X-axis) of the first sample of a in seconds. The return value variable is a number.</p> <div data-bbox="528 1637 1347 1798" style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  In Scope mode, the time is relative to the trigger time; in the case of recording in Multiblock, Continuous or ECR mode, the time is relative to the time of the recording start. </div> <p>Example:</p> <pre> ; generate a signal tr = Noise(10,1E6,1E6) ; get TBegin and TEnd t1 = TBegin(tr) t2 = TEnd(tr) ; set cursors to its position SetCrs("Waveform 1", "A", t1) SetCrs("Waveform 1", "B", t2) </pre>

<p>TCross(a, start, end, level)</p>	<p>The function searches from start in the direction end (moves forwards and backwards) to the next position where a exceeds or falls below level. The result variable is a number in seconds. The function will return NotDefined if there is no crossing found.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Please see also the function EnEvents. </div> <p>Example:</p> <pre> ; generate a signal tr = Sinus(10,50,0,1E3,1E3) tr = tr + Noise(2,tr) ; crossing level level = 0.5 ; create a list with all crossings levelList = List() t1 = TBegin(tr) t2 = TEnd(tr) ts = TSample(tr) tpos = t1 ; start from beginning to the end of the trace do while tpos < t2 tpos = TCross(tr, tpos, t2, level) levelList() = tpos ; add to list tpos = tpos+ts ; move one sample forward loop </pre>
<p>TEnd(a)</p>	<p>The function returns the time (X-axis) of the last sample of a in seconds. The return value variable is a number.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  In Scope mode, the time is relative to the trigger time; in the case of recording in Multiblock, Continuous or ECR mode, the time is relative to the time of the recording start. </div> <p>Example:</p> <pre> ; generate a signal tr = Noise(10,1E6,1E6) ; get TBegin and TEnd t1 = TBegin(tr) t2 = TEnd(tr) ; set cursors to its position SetCrs("Waveform 1", "A", t1) SetCrs("Waveform 1", "B", t2) </pre>
<p>TOfMax(a [, start, end])</p>	<p>The function returns the time (seconds) of the maximum found in a. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  When recording in Multiblock, Continuous or ECR mode, the time is relative to the start recording. </div> <p>Example:</p> <pre> ; generate a signal </pre>

	<pre>tr = Noise(10,1E6,1E6) ; read TOFMax and TOFMin tmax = TOFMax(tr) tmin = TOFMin(tr) t1 ; set cursors to its position SetCrs("Waveform 1", "A", tmax) SetCrs("Waveform 1", "B", tmin)</pre>
TOFMin(a [, start, end])	<p>The function returns the time (seconds) of the minimum found in a. The optional parameters start and end are time marks. Between this two, the calculation will be done.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  When recording in Multiblock, Continuous or ECR mode, the time is relative to the start recording. </div> <p>Example:</p> <pre>; generate a signal tr = Noise(10,1E6,1E6) ; read TOFMax and TOFMin tmax = TOFMax(tr) tmin = TOFMin(tr) ; set cursors to its position SetCrs("Waveform 1", "A", tmax) SetCrs("Waveform 1", "B", tmin)</pre>
TSample(a)	<p>The function returns the sample rate (Timebase rate in seconds) of a. The result is a number.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  The sampling rate is the inverse value of the sampling frequency ($T = 1/f$ or $f=1/T$). </div> <p>Example:</p> <pre>tr = Sinus(10,50,0,1E3,1E3) t1 = TSample(tr) ; t = 0.001s = 1ms, interval between each sample f1 = 1/t1 ; f = 1'000Hz = 1kHz, sample rate frequency</pre>
TTrigger(a)	<p>The function returns the trigger time (X-axis) of the curve trace in seconds.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">  When recording in Scope or Continuous Mode, the result is 0. When recording in Multi Block or ECR mode, the time is relative to the time the start recording. </div> <p>Example:</p> <pre>; Multi Block recording tr = c0A1.1 ; second block t1 = TTrigger(tr) tr = c0A1.4 ; fifth block t2 = TTrigger(tr) ; create a list with all trigger time of all blocks mylist = List() for each item in EnBlocks(c0A1, false) mylist() = TTrigger(item)</pre>

<p>UnderFlow(a [, start, end])</p>	<p>next</p> <p>Indicates an underflow of the ADC in the signal a.</p> <p>The optional parameters start and end are time marks in seconds. Between this two, the underflow will be detected.</p> <p>The return value is undefined (NaN) if no underflow has been found. Otherwise it will return the time were the underflow begins.</p> <p>Example:</p> <pre>tr = c0A1 of = OverFlow(tr) uf = UnderFlow(tr) if (of = NotDefined)and(uf = NotDefined) then SetMessage("Signals within Range", true) else Pause("Check Range settings!") endif</pre>
---	---

34.17 Signal Generations

<p>CreateSignal(periodCount, fs, (t1, y1)+) CreateSignal(periodCount, fs, a) CreateSignal(periodCount, trace, (t1, y1)+) CreateSignal(periodCount, trace, a)</p>	<p>With CreateSignal, any signal curves can be generated. The parameter periodCount is the number of periods (repetitions). This value will be rounded to an integer. It should be ≥ 1. fs defines the sampling rate (in Hz) of the generated curve.</p> <p>(t1, y1)+ always consists of two numbers: The first for the time on the x-axis, the second for the amplitude value. The number of x-y-Value pairs is not limited.</p> <p>With CreateSignal, any signal curves can be generated. The parameter periodCount is the number of periods (repetitions). This value will be rounded to an integer. It should be ≥ 1. fs defines the sampling rate (in Hz) of the generated curve.</p> <p>a represents a collection (array, list) which starts with the x-value (time) followed by the y-value.</p> <p>With CreateSignal, any signal curves can be generated. The parameter periodCount is the number of periods (repetitions). This value will be rounded to an integer. It should be ≥ 1. Parameter trace sets all the settings for the x-axis (sample rate, etc.).</p> <p>(t1, y1)+ always consists of two numbers: The first for the time on the x-axis, the second for the amplitude value. The number of x-y-Value pairs is not limited.</p> <p>With CreateSignal, any signal curves can be generated. The parameter periodCount is the number of periods (repetitions). This value will be rounded to an integer. It</p>
---	---

	<p>should be ≥ 1. Parameter trace sets all the settings for the x-axis (sample rate, etc.).</p> <p>a represents a collection (array, list) which starts with the x-value (time) followed by the y-value.</p> <p>Example:</p> <pre> ; Create a triangle signal periodCount = 5 ; 5 periods fs = 1e6 ; 1 MHz sampling rate dt = 1/fs freqTriangle[Hz] = 1e3 ; 1 kHz signal frequency peak[V] = 1 ; 1V peak value sigTriangle = CreateSignal(periodCount, fs, 0, -peak, 1/freqTriangle, peak) ; Create a rectangle signal with a duty cycle of 25% periodCount = 3 ; 3 periods fs = 1e6 ; 1 MHz sampling rate freqRect[Hz] = 5.5e3 ; 5.5 kHz signal frequency dt = 1/freqRect peak[V] = 3.3 ; 3.3V peak value dutyCyclePer[%] = 25 ; 25% duty cylce listPair = List() listPair() = 0 listPair() = peak listPair() = dt * dutyCyclePer / 100 listPair() = peak listPair() = listPair(2) + 1/fs listPair() = 0 listPair() = listPair(4) + dt * (100 - dutyCyclePer) / 100 - 1/fs listPair() = 0 sigRectangle = CreateSignal(periodCount, fs, listPair) </pre>
<p>Noise(ampl, fs, length) Noise(ampl, trace)</p>	<p>This function returns a random noise signal (Gaussian F-distribution, with standard deviation ampl). ampl (in V or Unit), fs (in Hz) and length (number of samples) are all numbers.</p> <p>This function returns a random noise signal (Gaussian F-distribution, with standard deviation ampl). The paramter ampl represents the amplitude for the noise. trace is used for the sampling rate and length of the noise curve.</p> <p>Example:</p> <pre>tr = Noise(5, 1E3, 100E3)</pre>
<p>Ramp(slope, fs, length) Ramp(slope, trace) Ramp(y0, y1, trace) Ramp(y0, y1, fs, length)</p>	<p>This function returns a ramp with the gradient set to slope (Unit per second). fs (in Hz) and length (number of samples) are all numbers.</p> <p>Creates a ramp curve with the gradient set to slope (Unit per second). The parameter trace defines the Sampling rate and length of the ramp.</p> <p>Creates a ramp with the start point y0 and end point y1. The time between y0 and y1 will be defined by sampling rate (fs) and length of trace (time=length/fs).</p>

	<p>Creates a ramp with the start point y0 and end point y1. The time between y0 and y1 will be defined by fs and length (time=length/fs).</p> <p>Example:</p> <pre>; creat a sawtooth signal tr = Ramp(0,1,1E3,1E3) trst = Merge(tr,tr,tr,tr) ; create a signal v[m/s] with 10[m/s^2] acceleration v[m/s] = Ramp(10,1E3,10E3) a[m/s^2] = diff(v)</pre>
<p>Sinus(ampl, freq, phase, fs, length) Sinus(ampl, freq, phase, trace)</p>	<p>This function returns a sine wave trace. The parameters for this function are ampl (in V or Unit), freq (in Hz), phase (in °), fs (in Hz) and length (number of samples).</p> <p>All parameters are numbers. Peak-peak value is twice the Parameter ampl value.</p> <p>This function returns a sine wave trace. The parameters for this function are ampl (in V or Unit), freq (in Hz), phase (in °), sample rate (in Hz) and length (number of samples) from trace.</p> <p>All parameters are numbers. Peak-peak value is twice the Parameter ampl value.</p> <p>Example:</p> <pre>tr1 = Sinus(10, 50, 0, 1E3, 1E3) ; 10V amplitude -> 20Vpp tr2 = Sinus(8, 50, 180, tr1) ; 180° phase shift to tr1</pre>

34.18 Signal Processing

<p>Averaging</p>	<p>Keyword to type in function DataReduction.</p>
<p>Convolution(a, b)</p>	<p>Convolves two arrays a and b. The result will be an array as the weighted average of array a.</p> <p>Length of the convolution array will be a + b - 1.</p> <p>Example:</p> <pre>a = Array(1:3) ; a= 1, 2, 3 b = Array(2:4) ; b= 2, 3, 4 c = Convolution(a,b) ; c= 2, 7, 16, 17, 12 len = Length(c); len= 5</pre>
<p>Correlation(a, b)</p>	<p>Calculates the correlation of a and b. If a and b are the same, the result corresponds to an autocorrelation function.</p> <p>Example:</p> <pre>; Find a spike pattern in a sinus signal with the help of cross correlation. ; The spike pattern can be of any complexity. For this example, it is only ; a simple spike. ; ----- ----- fs = 1e6 dt = 1/fs traceLength = 1e4</pre>

	<pre> peak = 2.5 peakLength = dt * 5 tSpikeStart = Random(0, dt * traceLength - peakLength) ; Randomly generate a time where the spike starts sineTrace = Sinus(1, 1e3, 0, fs, traceLength) ; Generate a sinus signal spikeTrace = CreateSignal(1, fs, 0, 0, tSpikeStart, 0, tSpikeStart + peakLength/2, peak, tSpikeStart + peakLength, 0, dt * traceLength, 0) ; Create a spike signal sineWithSpike = sineTrace + spikeTrace + Noise(0.1, fs, traceLength) ; Add the spike and noise to the sinus signal spikeOnly = Slice(sineWithSpike, tSpikeStart - peakLength, tSpikeStart + 2*peakLength) ; Get only the spike with the help of the slice function xyCorr = Correlation(sineWithSpike, spikeOnly) ; Find the displacement of the spike within the sinus signal tSpike = TofMax(xyCorr) - traceLength * dt ; Calculate the position of the spike begin ; Find the phase shift of two signals with cross correlation ; ----- ----- fs = 1e6 freqSin = 1e3 dt = 1/fs traceLength = 1e4 sineTrace1 = Sinus(1, freqSin, 0, fs, traceLength) ; Generate sine trace 1 sineTrace2 = Sinus(1.45, freqSin, 30, fs, traceLength) ; Generate sine trace 1 with 30° phase shift xyCorrPhase = Correlation(sineTrace1, sineTrace2) ; Correlate both signals to find the time of maximum match tPhaseShift[s] = TofMax(xyCorrPhase) - traceLength * dt ; Calculate the position of the spike begin tPhaseInDegree[°] = 360 / (1/freqSin) * (tPhaseShift) ; Calculate the phase shift in degrees </pre>
<p>DataReduction(a, type, factor)</p>	<p>Reduce data of Trace a by factor. a is a measurement curve, factor is a number ≥ 1. The sampling time will be adapted.</p> <p>For type the following methods are valid:</p> <ul style="list-style-type: none"> • Skipping: Only every nth sample will be returned in the resulting trace. • Averaging: The moving average of n samples will be calculated and then as one sample returned in the resulting trace. • MinMaxEnvelope: For every nth sample the smallest and largest value will be returned. With this method, two values for each resulting sample will be returned, one containing the lower and one the upper envelope value. <p>Example: tr1 = c0A1 ; get a trace TrRed=DataReduction(tr1, Averaging,10)</p>
<p>ExponentialFit(a [, returnCoeffs [, coeff_a, coeff_b]])</p>	<p>The function ExponentialFit generates an Exponential curve which optimally fits the curve a. For a falling curve in the positive range or a rising curve in the negative range, the function assumes that a $\rightarrow 0$ at x $\rightarrow +$. If the ratios are reversed, it is assumed that a $\rightarrow 0$ bei x $\rightarrow -$.</p> <p>The parameter returnCoeffs is optional. If this is set to True, instead of a curve, an array with the coefficients y0 and x0 is</p>

	<p>returned. The coefficients are according to the following formula: $f(x) = y_0 * e^{-x/x_0}$ CoefArr(0)=y0 (amplitude value at x=0) CoefArr(1)=x0</p> <p>The time constant can be determined from x0 (T= 1/x0).</p> <p>The two optional parameters coeff_a and coeff_b can be used as start parameters to optimize the fitting.</p> <p>Example:</p> <pre> ; create a test signal tr = Ramp(0,10,1E3,1E3) tr = tr + noise(5,tr) ; trace trExp = ExponentialFit(tr, false) ; array trArr = ExponentialFit(tr, true) T = 1/trArr(1) </pre>
<p>FrqDemod(a [, level [, hysteresis]])</p>	<p>The function searches for zero crossings of a and calculates the frequency at the individual points. With the optional parameters level and hysteresis the level and the hysteresis can be adjusted. These values then determine the search algorithm for the zero crossings. The function returns a trace as a result.</p> <p>Example:</p> <pre> ; create a signal tr1 = Sinus(10,50,0,1E3,1E3) tr2 = Sinus(10,100,0,1E3,1E3) tr = MergeTraces(tr1,tr2,tr1) trFreq= FrqDemod(tr) </pre>
<p>GetXAxisMode(name)</p>	<p>Returns the current X-axis setting of Waveform name.</p> <p>Return values for variable are:</p> <ul style="list-style-type: none"> 0 = Relative time since Start 1= Absolute Time 2= Relative Time (Zero at Trigger) 3= Samples\ <p>Example:</p> <pre> ; 0 = Relative time since Start ; 1 = Absolute Time ; 2 = Relative Time (Zero at Trigger) ; 3 = Samples mode = GetXAxisMode("Waveform 1") </pre>
<p>MinMaxEnvelope</p>	<p>Keyword to type in function DataReduction.</p>
<p>Phase(a, b [, level_a, hysteresis_a, level_b, hysteresis_b])</p>	<p>The function searches for zero crossings of a and b, then calculates the phase shift from both and returns a curve with values in degrees.</p>

	<div data-bbox="550 165 632 248" style="float: left; margin-right: 10px;">  </div> <p data-bbox="643 188 1305 224">The phase shift of a is compared to b as reference.</p> <p data-bbox="507 315 1315 427">Optionally, levels (level_a, level_b) and hysteresis (hysteresis_a, hysteresis_b) of the two curves can be defined to affect the zero-crossing algorithm.</p> <p data-bbox="507 472 635 504">Example:</p> <pre data-bbox="507 508 1281 647">tr1 = Sinus(10,50, 0,1E3,1E3) tr2 = Sinus(10,50,20,1E3,1E3) ; tr2 starts with 20° offset ; compare tr2 to tr1 ph = Phase(tr2, tr1) ; curve rises up to 20°</pre>
<p data-bbox="132 656 453 960">RegressionPoly(a, order [, returnCoeffs [, tExtrapolAhead [, tExtrapolBehind]]]) RegressionPoly(a, fs, order, returnCoeffs [, tExtrapolAhead [, tExtrapolBehind]])</p>	<p data-bbox="507 656 1347 882">The function calculates a fitted curve of a that corresponds to the polynomial function of the n-th order (e.g. polynomials with $n = 2 \implies (k_1 * a^2 + k_2 * a + k_3)$. k_1, k_2, and k_3 are determined by appropriate regression). The result curve variable does not show any big jumps and discontinuities anymore.</p> <p data-bbox="507 927 1339 1039">If the optional third parameter returnCoeffs is set to True (= 1), the function returns an array with the polynomial coefficients k_0, k_1, k_2, ... instead of a curve.</p> <p data-bbox="507 1084 1321 1196">The other two optional parameters tExtrapolAhead and tExtrapolBehind indicate how many seconds should be added before and after the calculated regression.</p> <p data-bbox="507 1240 1347 1509">The function calculates a fitted curve of a that corresponds to the polynomial function of the n-th order (e.g. polynomials with $n = 2 \implies (k_1 * a^2 + k_2 * a + k_3)$. k_1, k_2, and k_3 are determined by appropriate regression). a has to be an array or list. fs is used as the sampling rate for the result curve. The result curve variable does not show any big jumps and discontinuities anymore. variable.</p> <p data-bbox="507 1554 1339 1666">If the optional third parameter returnCoeffs is set to True (= 1), the function returns an array with the polynomial coefficients k_0, k_1, k_2, ... instead of a curve.</p> <p data-bbox="507 1711 1321 1823">The other two optional parameters tExtrapolAhead and tExtrapolBehind indicate how many seconds should be added before and after the calculated regression.</p> <p data-bbox="507 1868 635 1899">Example:</p> <pre data-bbox="507 1904 1267 2051">; create a test signal tr = Ramp(0,10,1E3,1E3) tr = tr + noise(5,tr) trReg = RegressionPoly(tr,3,false, 0.1, 0.05) ; add 100ms before und 50ms after the calculated trace</pre>

Resampling(a, fs)	<p>Changes the sampling frequency fs (in Hz) of curve a. Is the fs faster as the original sampling frequency of a, then the missing samples are linearly interpolated. If it is slower, samples will be removed.</p> <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin: 10px 0;">  Likely the trigger point can shift a fraction, because it will be tied to the next sample point after the operation. </div> <p>Example: <code>tr_orig = c0A1</code> <code>OrigSR = 1/TSample(tr_orig)</code> <code>NewSR= 3.1234 * OrigSR</code> <code>tr_new = Resampling(tr_orig, NewSR)</code></p>
SetTrigger(a, t)	<p>The function sets the trigger time t (seconds) of curve a. The behavior is comparable to a manipulated trigger delay. The values for TBegin and TEnd remain at their original values.</p> <p>The return value variable is a curve.</p> <p>Example: <code>tr = Sinus(10,50,0,1E3,1E3)</code> <code>trNew = SetTrigger(tr, 0.03)</code></p>
SetTSample(a, t)	<p>The function returns the trace a with the new sampling period t (seconds). The time axis of the signal curve is changed. The return value variable is a curve. This function changes the sampling rate. The curve modified in this way has the same number of samples, but the time domain is stretched or compressed.</p> <p>Example: <code>trOrig = Sinus(10,50,0,1E3,1E3)</code> <code>tsNew = TSample(trOrig)/2</code> <code>trNew = SetTSample(trOrig, tsNew)</code></p>
SetXAxis(a, refTrace) SetXAxis(a, t0 [, dt [, triggerSample [, unit]]])	<p>Creates a curve with the amplitude values of a and the time axis values of curve refTrace (sampling rate, trigger delay, begin, end, etc.). Values at the beginning and end of the returned curve will be trimmed or filled with zero to fit into the parameters of refTrace.</p> <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin: 10px 0;">  If the sampling rate of refTrace is different, the returned curve will be resampled. </div> <p>Creates a curve with the amplitude values of a and the following time axis parameters:</p> <ul style="list-style-type: none"> • t0: Begin of result curve • dt: Sampling rate of result curve • triggerSample: The effective trigger sample of result curve

	<ul style="list-style-type: none"> • unit: Unit of result curve. <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;">  If the sampling rate dt is different of a, the returned curve will be resampled. </div> <p>Example: <code>y = File("measure.tpc5", 0)</code> <code>x = c0A3.0</code> <code>z = SetXAxis(y, x)</code></p>
SetXAxisMode(name, axisMode)	<p>Sets the Axis Mode of an existing waveform display name.</p> <p>The second parameter axisMode can have the following values:</p> <ul style="list-style-type: none"> • 0 = Relative Time since Start • 1 = Absolute Time • 2 = Relative Time (Zero at Trigger) • 3 = Samples <p>Example: <code>; set axis mode to Relative Time since Start</code> <code>SetXAxisMode("Waveform 1", 0)</code></p>
Shift(a, b)	<p>Shifts a curve a by the value b which can be another curve, time (seconds) or a name of a waveform display. The start time (TBegin) is reset. The shift is not rounded to whole samples. From the beginning, the support values of the new and old curves are no longer superimposed. This is to be considered, if during further evaluation operations the beginning and not the trigger time is taken as a reference. If instead of time a second curve b is specified, the result curve moves into the time range of b. The triggers of the two curves are then superimposed.</p> <p>Example: <code>tr = Sinus(10,50,0,1E3,1E3)</code> <code>trShift = Shift(tr, 0.5) ; 0.5 seconds to the right</code></p>
Skip(a, number)	<p>This function creates a resampling of a so that the result curve has correspondingly fewer or more samples (Downsampling or Upsampling). If number is positive a result curve with number times less samples results. With number as a negative number Abs(number) times more Samplings are generated.</p> <p>The parameter number can also be a fractional number. Intermediate values are determined by linear interpolation.</p> <p>This function, possibly combined with filters, allows for data reduction which is useful when oversampling signals have been recorded and a computation-intensive re-evaluation (for example, the function Correlation or FFT) is to be performed.</p> <p>The return value variable is a curve.</p> <p>Example: <code>tr = Sinus(10,50,0,1E3,1E3)</code></p>

	<pre>trLess = Skip(tr, 2) ; half of the samples trMore = Skip(tr, -5) ; 5 time more samples</pre>
Skipping	Keyword to type in function DataReduction .
Slice(a, start, end [,y0 [,y1]])	<p>The function cuts a slice between start and end out of a. If a is a curve then start and end is a time which will be in seconds. Otherwise start and end will act as an index.</p> <p>If start < TBegin(a) reps. end > TEnd(a), the curve will be extended, if at least y0 is given. The curve a will be extended with the constant amplitude value y0. if additionally, y1 is specified, curve a will be extended at its end with y1 (if end > TEnd(a)) and y0 at its start (if start < TBegin(a)). The result is a shortened or lengthened curve.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  The Slice function can also be used to cut a slice from an array, list or text. </div> <p>Example: <pre>variable = Slice(a, t0, t1) variable = Slice(a, t0, t1 [,y0 [,y1]]) variable = Slice(Array or List, startIndex, endIndex)</pre></p>
Spline(a, order, fs [, tExtrapolAhead[, tExtrapolBehind]]) Spline(a, order, refTrace [, tExtrapolAhead[, tExtrapolBehind]])	<p>Calculates a polynomial fit from a. If a is an array or a list, they have to contain time-value pairs (Time in seconds). The parameter order is currently set to 3 (cubic spline), other values are ignored.</p> <p>The parameter fs is the sampling rate (frequency) with which the resulting curve has been digitized.</p> <p>tExtrapolAhead and tExtrapolBehind are two optional time parameters (positive values in seconds) which can extend the result curve at start or end.</p> <p>Calculates a polynomial fit from a. If a is an array or a list, they have to contain time-value pairs (Time in seconds). The parameter order is currently set to 3 (cubic spline), other values are ignored.</p> <p>The parameter refTrace is another curve which sets all the x-axis relevant information (sample rate, trigger sample, tbegin, absolute start time, unit).</p> <p>tExtrapolAhead and tExtrapolBehind are two optional time parameters (positive values in seconds) which can extend the result curve at start or end.</p> <p>Example: <pre>; create a triangle by x,y value pairs arr = Array(0,0, 0.05,0, 0.1,1, 0.2,0, 0.25,0) tr = CreateSignal(1, 1e3, arr) tr_sp = Spline(arr, 3, 1e3, 0.02, 0.015)</pre></p>
StdDev(a, width)	The function calculates the standard deviation from a . The parameter width defines how many values should be included in the calculation.

	<p>The result variable is again a curve, an array or a list.</p> <p>Example:</p> <pre> ; create some random values ; e.g. time duration for a light barrier in milliseconds items = 100 myArr = Array(0 to items-1) as double for a = 0 to items-1 step 1.0 myArr(a) = random(58, 74) ; random values form 58ms to 74ms next tmp = StdDev(myArr, items) ; StdDev returns an array standardDeviation = tmp(items-1) ; use last item meanValue = mean(myArr) ; calculate mean ; calculate the percentage of light barrier values below 68 ms pcVal = NormDist(68, meanValue, standardDeviation, true) ; calculate the minimal duration to be on the upper 75% msVal = NormInv(0.25, meanValue, standardDeviation) </pre>
<p>Variance(a, width)</p>	<p>The function calculates the variance of a. The parameter width defines how many values should be included in the calculation.</p> <p>The result variable is again a curve, an array or a list.</p> <p>Example:</p> <pre> tr = Sinus(10, 50, 0, 1E3, 1E3) arr = Array(0:99) varTr = mean(Variance(tr, 20)) ; a number instead of a trace varArr = (Variance(arr, 10))(0) ; the first element of the array </pre>

34.19 Spectrum (FFT)

<p>AcousticAnalysis(a, doTerzAnalysis [, normedFrequencies])</p>	<p>Acoustic analysis of the a. The return value is an array of type Number. In an octave analysis there are 10, a 1/3 octave analysis 30 values in the array variable. These correspond to the representation of the FFT display.</p> <p>The 31 frequency bands (n+1) for a 1/3 octave analysis in Hz: 22.5, 28.2, 35.5, 45, 56, 71, 90, 112, 140, 180, 224, 280, 355, 450, 560, 710, 890, 1120, 1400, 1800, 2240, 2800, 3550, 4500, 5600, 7100, 9000, 11200, 14130, 17780, 22400</p> <p>The 11 frequency bands (n+1) for an octave analysis in Hz: 22.5, 45, 90, 180, 355, 710, 1400, 2800, 5600, 11200, 22400</p> <p>The spectrum curve a may needs to be calculated first with the function FFT of an existing measurement. The doTerzAnalysis parameter is of type boolean. Is this set true is a 1/3 octave will be calculated, false means a calculation of an octave.</p> <p>The parameter normedFrequencies is used to define custom frequency bands for analysis.</p> <p>Example:</p> <pre> ; create a signal tr1 = Sinus(10,1E3,0,1E6,1E3) tr2 = Noise(10,tr1) tr = MergeTraces(tr1, tr2, tr1) ; create a FFT array arrFFT = fft(tr, HammingWin) ; convert FFT array to spectrum trace trSpec = ConvToSpectrumTrace(arrFFT) ; Terz Analysis, array with 30 items (1/3 octave) trAc = AcousticAnalysis(trSpec, true) </pre>
<p>Angle(a)</p>	<p>Calculates the angle (or phase) of complex numbers. The result is a number or a double array. Please see also the functions Abs, Conjugate, Real, Imag.</p> <p>Example:</p> <pre> x = -3+5j ; complex number re = Real(x) ; re = -3 im = Imag(x) ; im = 5 ar = angle(x) ; ar = 120.96° co = Conjugate(x) ; co = -3-5j ; alternatively to function angle() deg = atan(imag(x),real(x)) ; deg = 120.96° </pre>
<p>BlackmanHarris3Win</p>	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example:</p> <pre> sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4) fftVal = FFT(sineTrace, BlackmanHarris3Win) </pre>
<p>BlackmanHarris4Win</p>	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p>

	<p>Example: sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4) fftVal = FFT(sineTrace, BlackmanHarris4Win)</p>
BlackmanWin	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4) fftVal = FFT(sineTrace, BlackmanWin)</p>
Conjugate(a)	<p>Mathematic function for complex numbers. Inverts the imaginary part of the elements in a. The result is a single complex number or an array of complex numbers. See also: Abs, Angle, Real, Imag</p> <p>Example: x = -3+5j ; complex number</p> <p>re = Real(x) ; re = -3 im = Imag(x) ; im = 5 ar = angle(x) ; ar = 120.96° co = Conjugate(x) ; co = -3-5j</p> <p>; alternatively to function angle() deg = atan(imag(x),real(x)) ; deg = 120.96°</p>
ConvToSpectrumTrace(a)	<p>With this function, an array with complex values a is transformed in a spectrum curve. Successively it can be shown then in an FFT window. The resulting spectrum curve is scaled as peak-value.</p> <p>Example: tr = c0A1 arrfft = FFT(tr, HannWin) trfft = ConvToSpectrumTrace(arrfft)</p>
FFT(a [, window])	<p>Fast Fourier Transformation is used to transform time domain curve a into the frequency domain Array of Complex (real and imaginary) values. All samples of a from beginning to end are transformed. However, the curve can be made shorter with the function Slice.</p> <p>The parameter window is the weighing functions for the signal in the time domain. If missing, automatically RectangleWin will be used (e.g. all values in time domain are equal in weight). With the function ConvToSpectrumTrace a spectrum curve can be formed from the Array of Complex. Then this curve can be shown in a FFT Spectrum Window.</p> <p>Example: tr = c0A1; arr1 = FFT(tr) arr2 = FFT(tr, HammingWin)</p> <p>trSpec1 = ConvToSpectrumTrace(arr1) trSpec2 = ConvToSpectrumTrace(arr2)</p>

FlatTopWin	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4) fftVal = FFT(sineTrace, FlatTopWin)</p>
FreqAtMax(spectrum)	<p>Determines the frequency with the highest amplitude. The curve spectrum is an amplitude spectrum and may need to be calculated from a recorded signal. Return value is a scalar of the type double in [Hz].</p> <p>Example: tr = c0A1 arrFFT = fft(tr, HammingWin) trSpec = ConvToSpectrumTrace(arrFFT) fAtMax = FreqAtMax(trSpec)</p>
HammingWin	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4) fftVal = FFT(sineTrace, HammingWin)</p>
HannWin	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4) fftVal = FFT(sineTrace, HannWin)</p>
IFFT(a)	<p>Inverse Fourier Transformation. This transforms an array with complex frequency domain values a in a time domain array. A rectangle window (RectangleWin) must have been used in the original FFT calculation in order to obtain a similar to original time domain curve.</p> <p>Example: tr_orig = c0A1</p> <pre> ; parameters for later calculations sr = 1/TSample(tr_orig) Tb =TBegin(tr_orig) ; calculate FFT array arr_fft = FFT(tr_orig, RectangleWin) ; convert back to time array and a trace arr_time = ifft (arr_fft) tr_new = ConvToTrace(arr_time, sr) ; Adjust x axis tr_new = Shift(tr_new, Tb) </pre>
Imag(a)	<p>Represents the imaginary part of complex numbers. a can be a single complex number or an array of complex numbers (e.g. an</p>

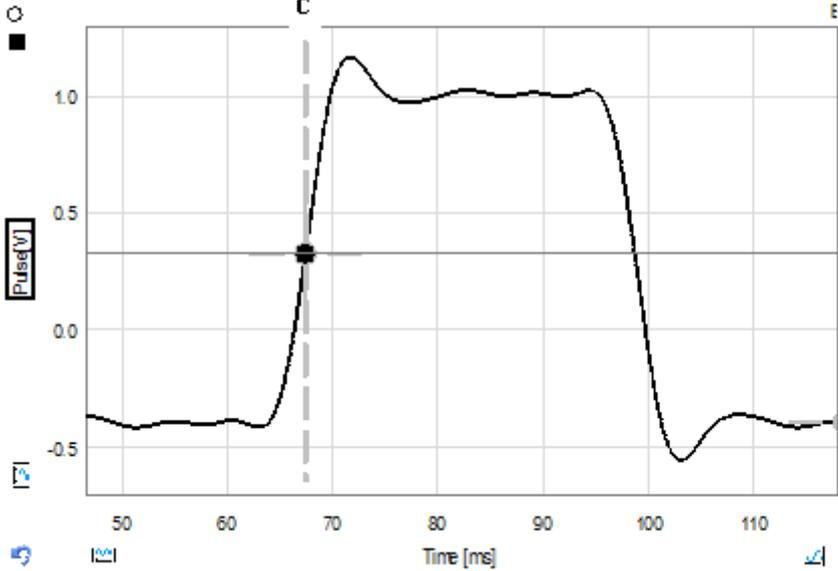
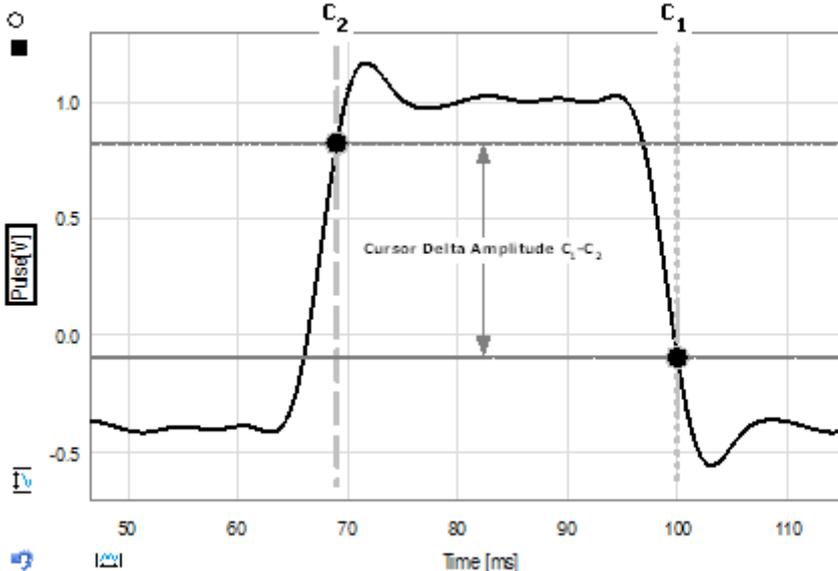
	<p>FFT result). The result again shall be a single number or an Array of Double. See also: Abs, Angle, Conjugate, Real</p> <p>Example: <code>x = -3+5j ; complex number</code></p> <code>re = Real(x) ; re = -3</code> <code>im = Imag(x) ; im = 5</code> <code>ar = angle(x) ; ar = 120.96°</code> <code>co = Conjugate(x) ; co = -3-5j</code> <p><code>; alternatively to function angle()</code> <code>deg = atan(imag(x),real(x)) ; deg = 120.96°</code></p>
Real(a)	<p>Represents the real part of complex numbers. a can be a single complex number or an array of complex numbers (e.g. an FFT result). The result again shall be a single number or an array of double numbers. See also: Abs, Angle, Conjugate, Imag</p> <p>Example: <code>x = -3+5j ; complex number</code></p> <code>re = Real(x) ; re = -3</code> <code>im = Imag(x) ; im = 5</code> <code>ar = angle(x) ; ar = 120.96°</code> <code>co = Conjugate(x) ; co = -3-5j</code> <p><code>; alternatively to function angle()</code> <code>deg = atan(imag(x),real(x)) ; deg = 120.96°</code></p>
RectangleWin	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: <code>sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4)</code> <code>fftVal = FFT(sineTrace, RectangleWin)</code></p>
Taylor60Win	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: <code>sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4)</code> <code>fftVal = FFT(sineTrace, Taylor60Win)</code></p>
Taylor80Win	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: <code>sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4)</code> <code>fftVal = FFT(sineTrace, Taylor80Win)</code></p>
TriangleWin	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: <code>sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4)</code> <code>fftVal = FFT(sineTrace, TriangleWin)</code></p>

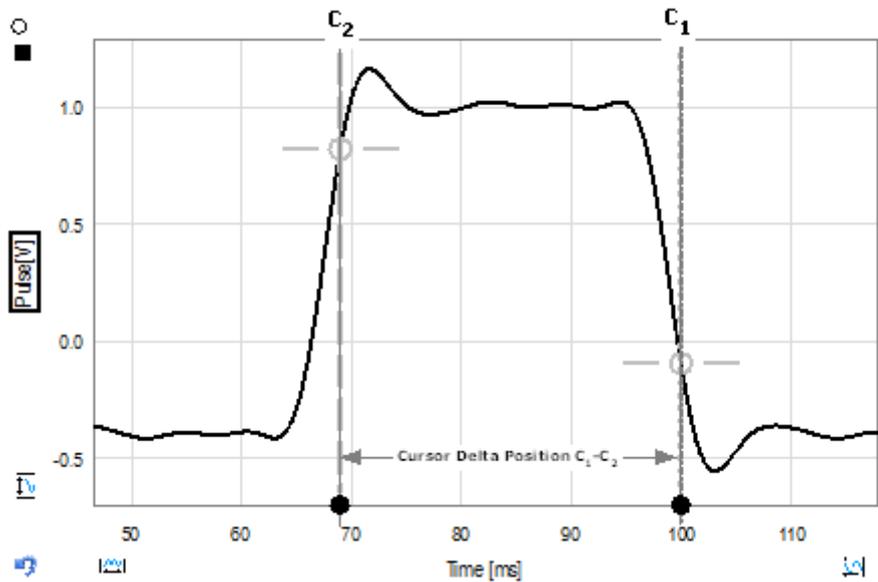
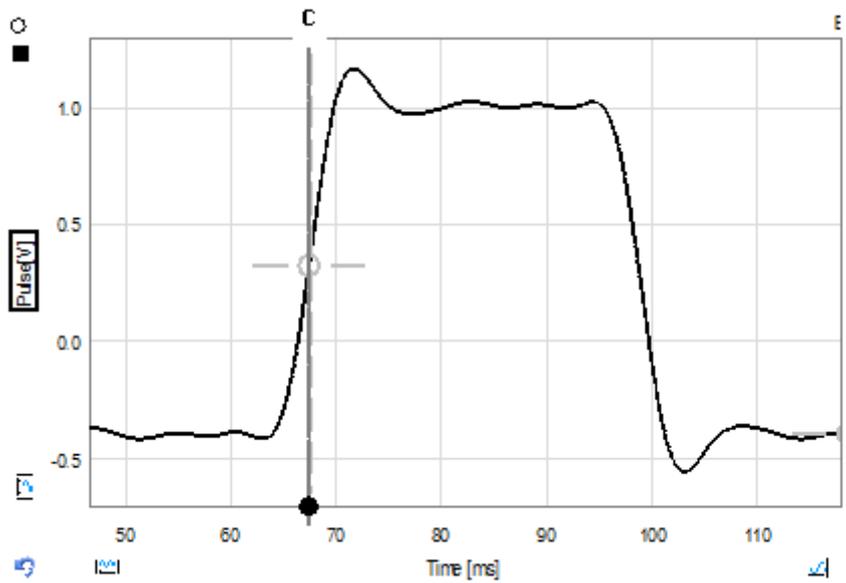
WelchWin	<p>Keyword for the parameter window of the function FFT, defines the evaluation window when executing the Fast Fourier Transformation.</p> <p>Example: sineTrace = Sinus(1, 1e3, 0, 1e6, 1e4) fftVal = FFT(sineTrace, WelchWin)</p>
-----------------	---

35 Scalar Functions Description Table

All Scalar Functions are available in the Formula Editor.

35.1 Group Cursor

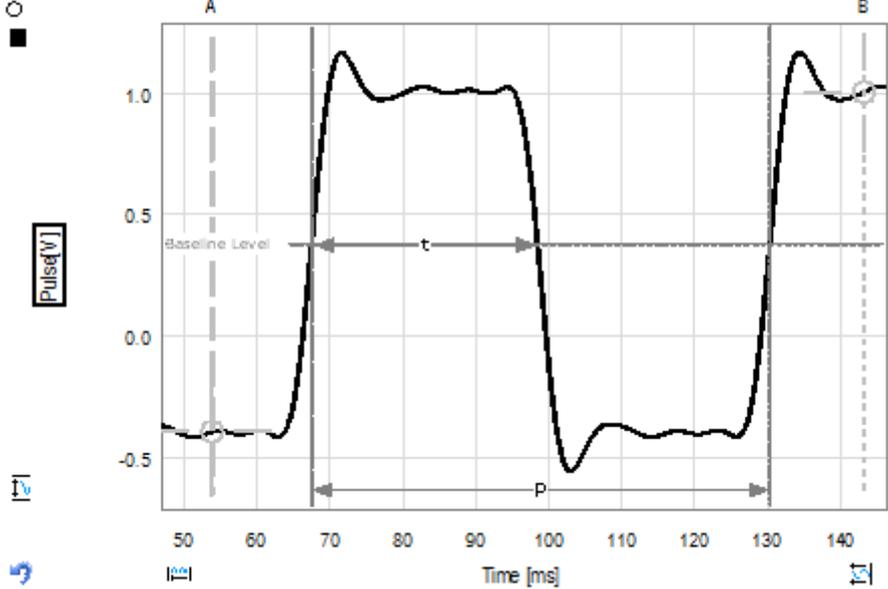
<p>Cursor Amplitude</p>	<p>Returns the y-value of the curve at the x-position of the selected cursor.</p>  <p>Cursor Amplitude = y_C</p>
<p>Cursor Delta Amplitude</p>	<p>Returns the difference of the y-values of the selected cursor pair.</p>  <p>Cursor Delta Amplitude = $y_{C1} - y_{C2}$</p>
<p>Cursor Delta Position</p>	<p>Returns the difference of the x-axis positions of the selected cursor pair.</p>

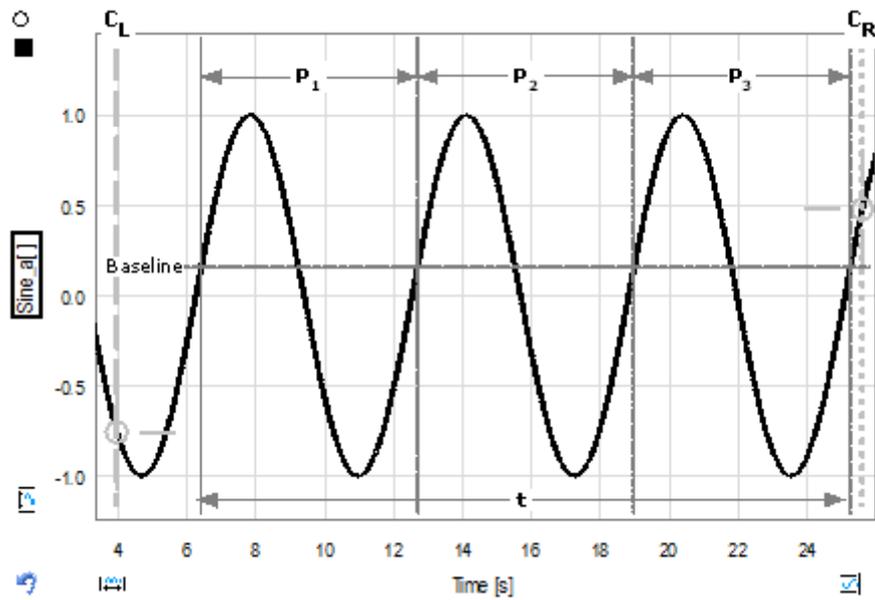
	 <p>Cursor Delta Position = $x_{C1} - x_{C2}$</p>
<p>Cursor Position</p>	<p>Returns the x-axis position of the selected cursor.</p>  <p>Cursor Position = x_C</p>
<p>Cursor Ratio Amplitude (dB)</p>	<p>Calculates the Amplitude Ratio between cursor 1 and cursor 2, scaled in dB.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  This is only available for scalar spectra (FFT) </div> <p>For Peak, RMS and phase (setting the Y-scale) is:</p> $CursorRation[dB] = 20 * \log_{10}\left(\frac{Cursor_1}{Cursor_2}\right)$

For **Power and RMS^2** (setting the Y-Scale) is:

$$CursorRatio[dB] = 10 * \log_{10}\left(\frac{Cursor_1}{Cursor_2}\right)$$

35.2 Group Horizontal

<p>Duty Cycle</p>	<p>Duty Cycle describes the ratio between the positive pulse width and the period time. This scalar searches for the crossing points at the baseline level to determine the positive pulse width and the period.</p>  <p>Duty Cycle = $\frac{t}{P}$</p> <p><i>t</i>: Pulse Width at baseline level <i>P</i>: Period at baseline level</p>
<p>Frequency</p>	<p>Searches for the baseline level crossing points to determine the average frequency. A hysteresis value is given to prevent noise causing erroneous crossover measurements.</p>



$$Frequency = \frac{N}{t}$$

C_L : Cursor left

C_R : Cursor right

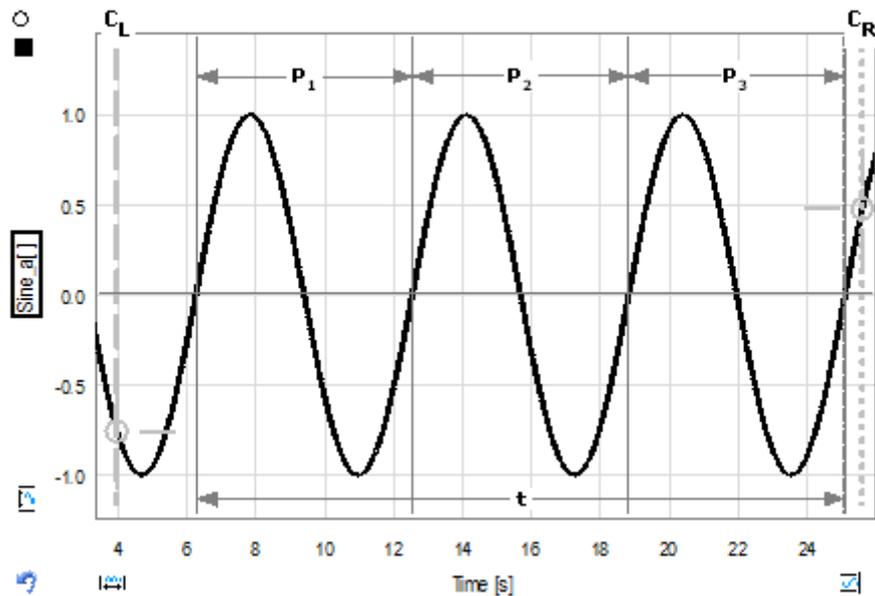
P_x : Period

N : Number of periods

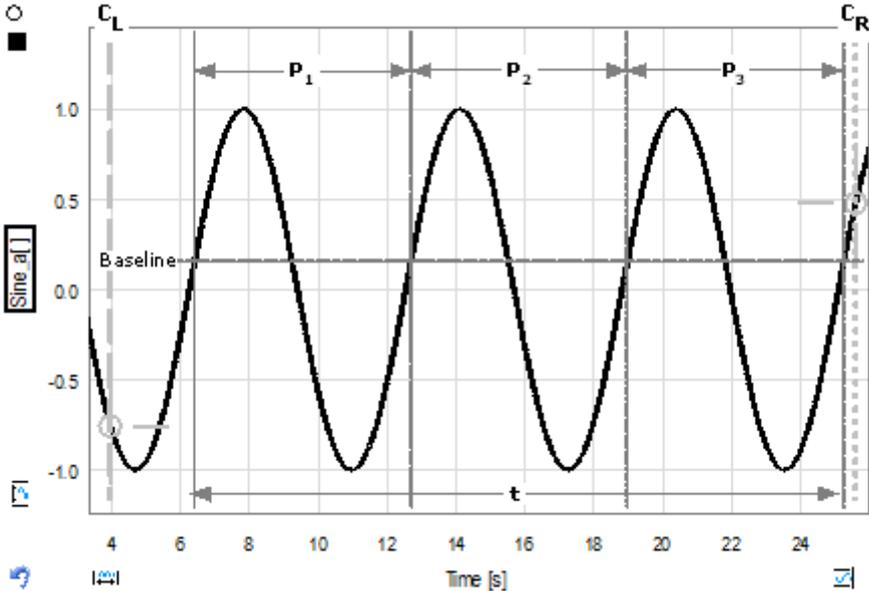
t : Time between first and last baseline crossing

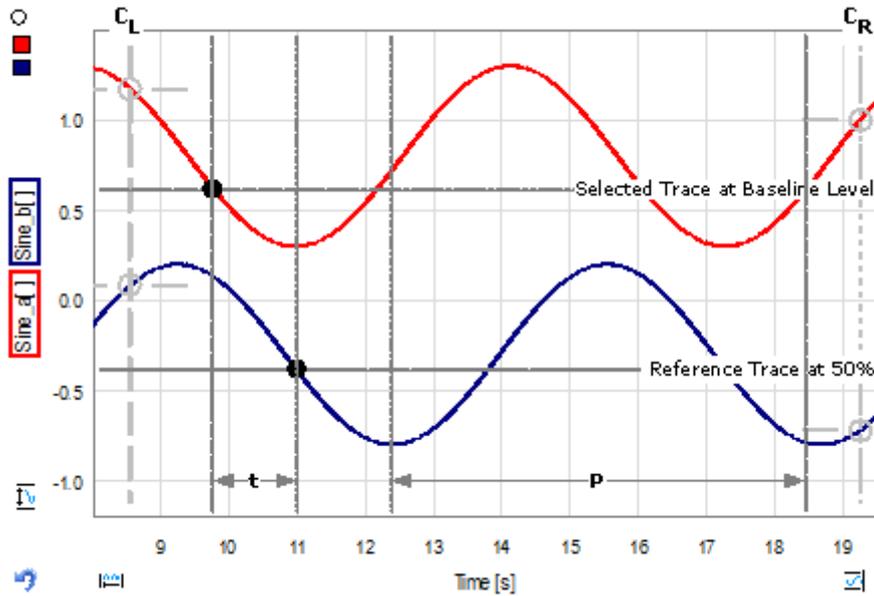
Number of Periods

Searches the crossing points at baseline level to determine the number of periods. A hysteresis can be applied to prevent noise causing erroneous measurements.



$$Number\ of\ Periods = \frac{t}{P}$$

	<p>C_L: Cursor left C_R: Cursor right P_x: Period t: Time between first and last level crossing</p>
<p>Period</p>	<p>Searches the crossing points at baseline level to determine the mean period length over the complete number of periods between the cursors. A hysteresis can be applied to prevent noise causing erroneous measurements.</p>  <p>$Period = \frac{t}{N}$</p> <p>C_L: Cursor left C_R: Cursor right N: Number of Periods t: Time between first and last level crossing</p>
<p>Phase</p>	<p>Returns the phase difference in degrees between the signal analyzed and the signal used as a reference of two periodic signals with identical frequency (usually voltage and current). The reference signal that can be selected from the corresponding drop down list is analyzed at 50% of the amplitude.</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p> A min. of 3.5 signal periods are required between the cursors to calculate the Phase. A negative result will be returned in the range of $\Phi \geq \pm 180^\circ$, if the reference signal lags the other one.</p> </div>



$$Phase = 360 \cdot \frac{t}{P}$$

C_L : Cursor left

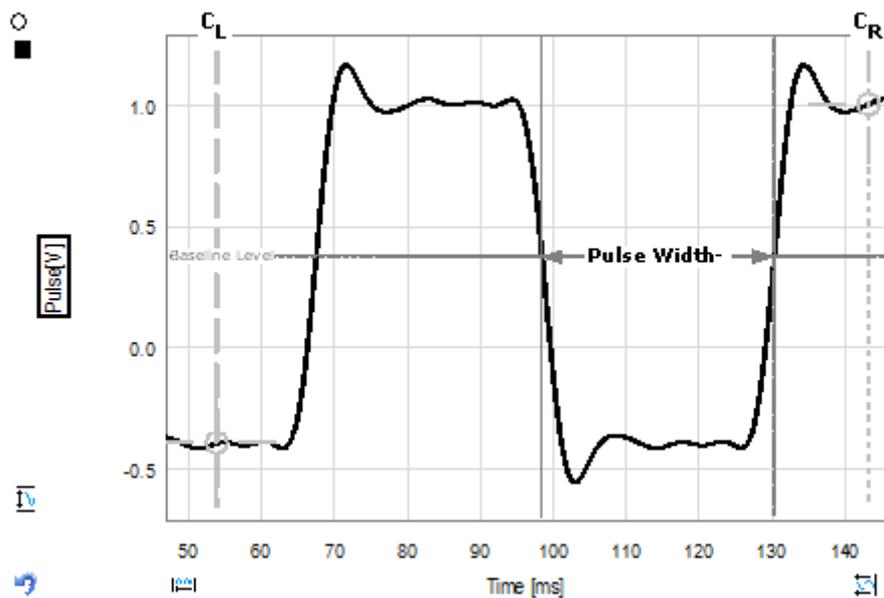
C_R : Cursor right

P : Period

t : Time difference at baseline level to 50%

Pulse Width (neg)

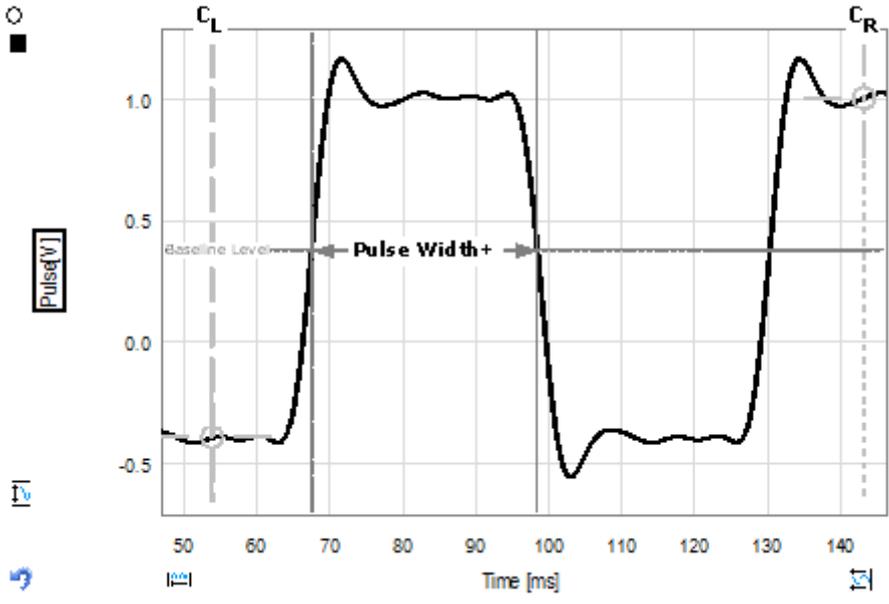
Returns the negative pulse width at baseline level. In case multiple periods are between the cursors, the average negative pulse width will be returned.

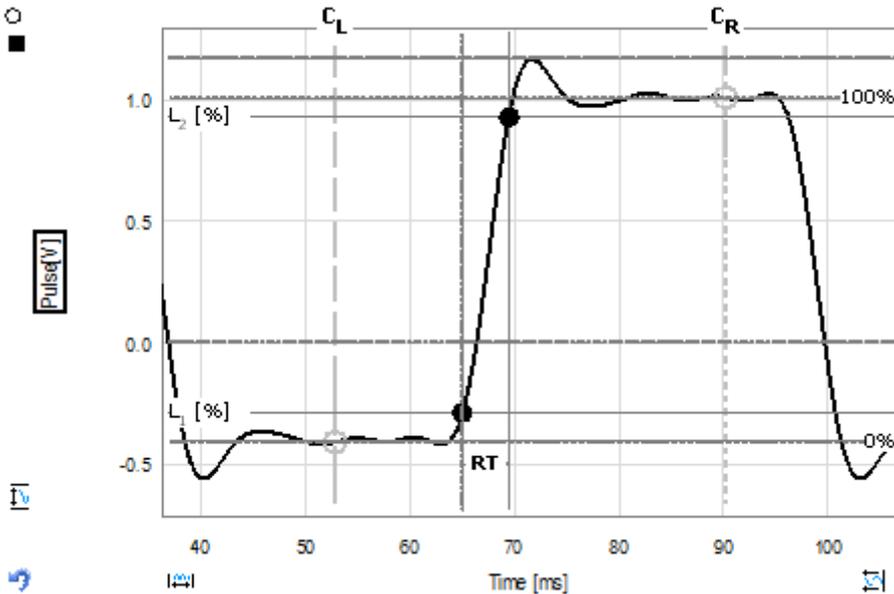
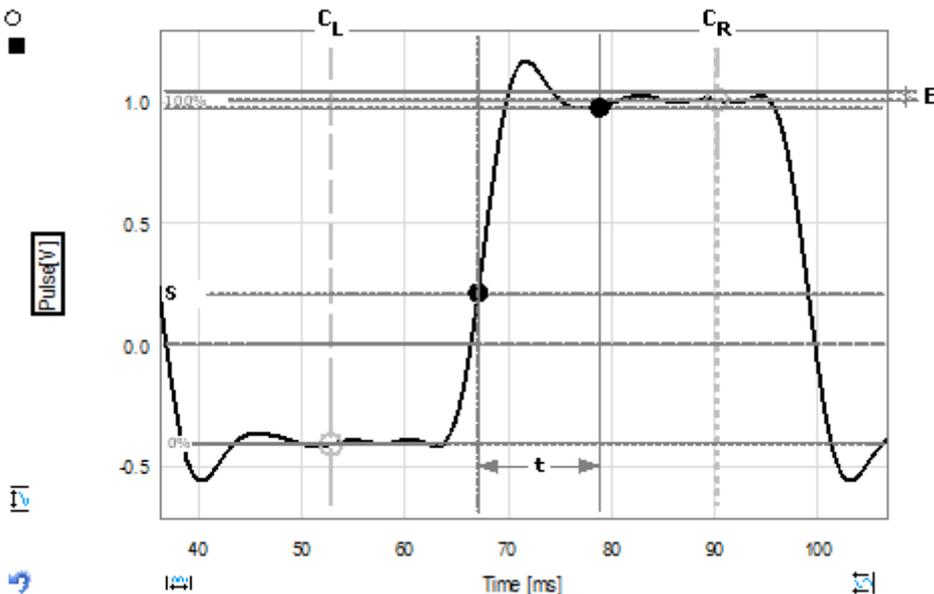


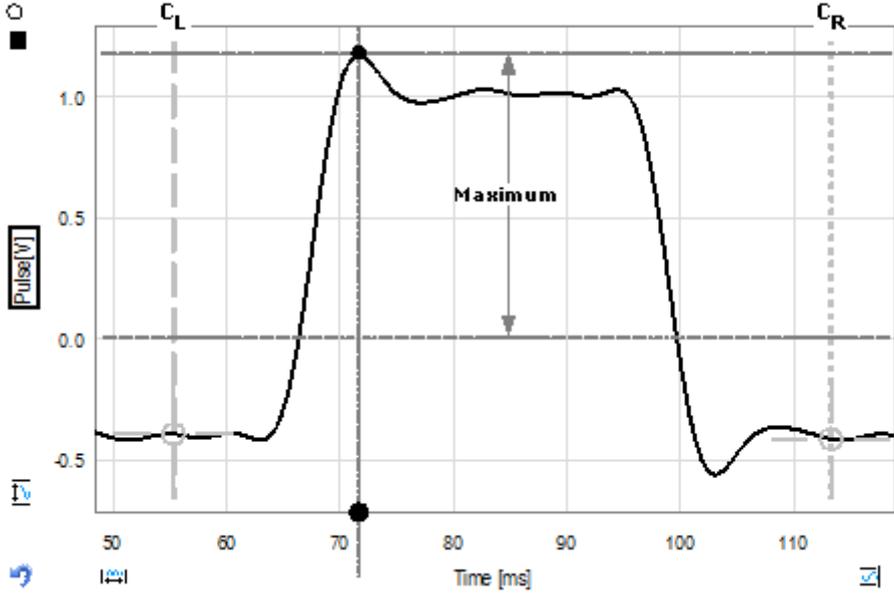
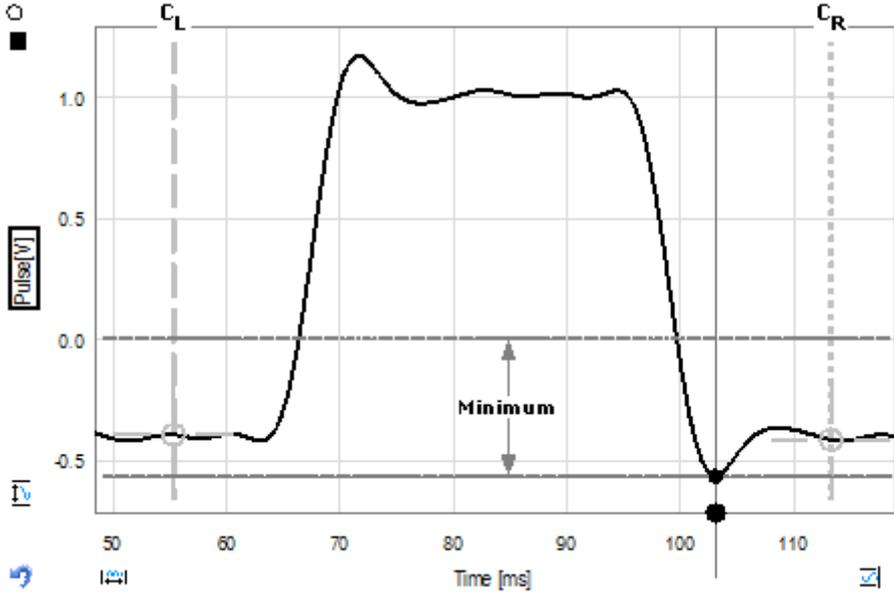
C_L : Cursor left

C_R : Cursor right

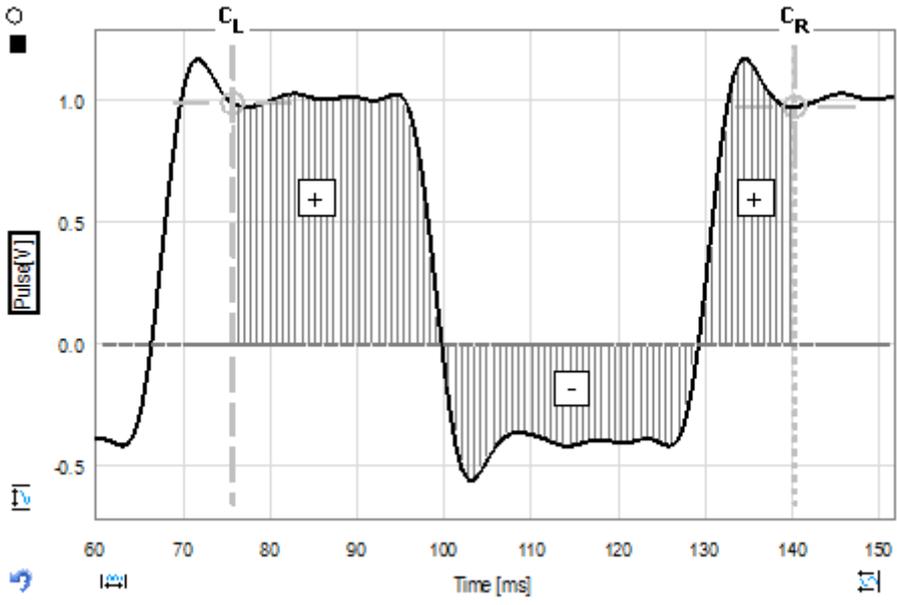
Pulse Width-: Negative Pulse Width at baseline level

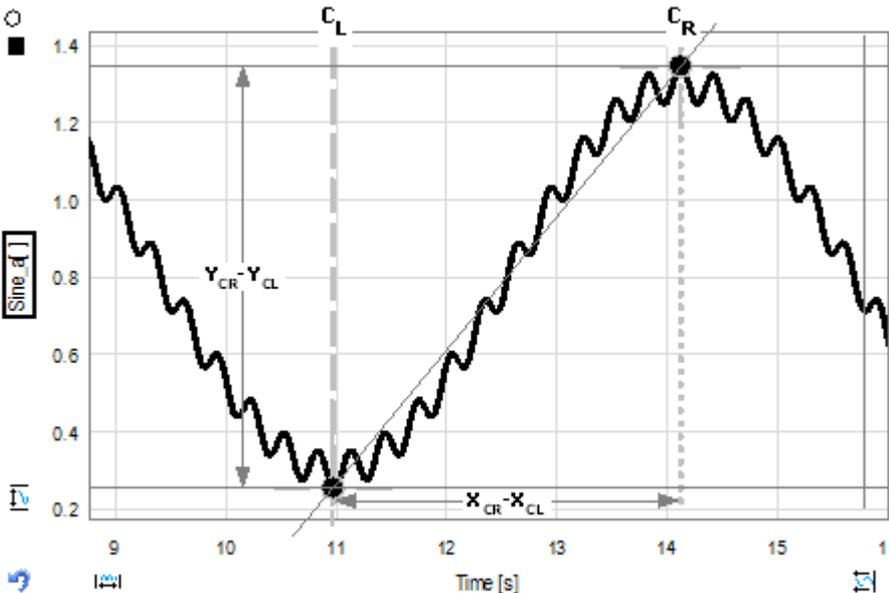
<p>Pulse Width (pos)</p>	<p>Returns the positive pulse width at baseline level. In case multiple periods are between the cursors, the average positive pulse width will be returned.</p>  <p>C_L: Cursor left C_R: Cursor right Pulse Width+: Positive Pulse Width at baseline level</p>
<p>Rise/Fall time</p>	<p>Calculates the rise or fall time of a trace at levels [%].</p> <ul style="list-style-type: none"> • To calculate the rise time, position the left cursor on the base level (0%) and the right cursor on the top level (100%). • To calculate the fall time, position the left cursor on the top level (100%) and the right cursor on the base level (0%). <p> For this scalar function two levels need to be set (e.g. 10% and 90%). They determine the crossing levels on the slope for the calculation of the rise time and fall time. The function takes the amplitude values at the positions of the cursors as 0% (Base) and 100% (Top).</p>

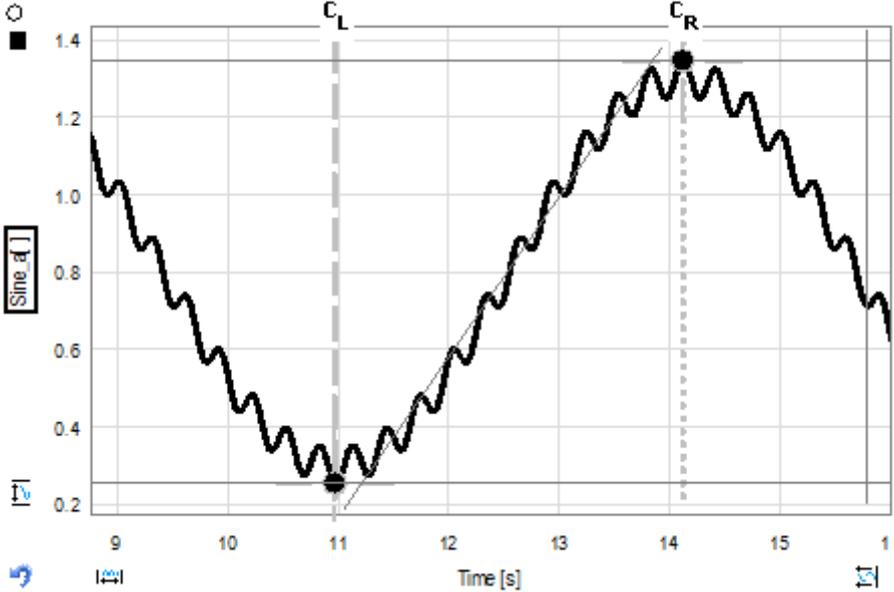
	 <p> C_L: Cursor left (0% amplitude) C_R: Cursor right (100% amplitude) L_1[%]: Level 1 (Lower Level [%] for Rise Time) L_2[%]: Level 2 (Upper Level [%] for Rise Time) RT: Rise Time </p>
<p>Setting time</p>	<p>Returns the time required for the signal to remain bounded between an error band whereas the error band is set around the end-level (100%).</p>  <p> C_L: Cursor left (0%) C_R: Cursor right (100%) S: Starting point level [%] E: Ending point level - Half error band [%] </p>

	<p>t: Settling time</p> <p>Time at Maximum Searches for the maximum in waveform and returns the time at maximum.</p>  <p>C_L: Cursor left C_R: Cursor right</p>
	<p>Time at Minimum Searches for the minimum in waveform and returns the time at minimum.</p>  <p>C_L: Cursor left C_R: Cursor right</p>

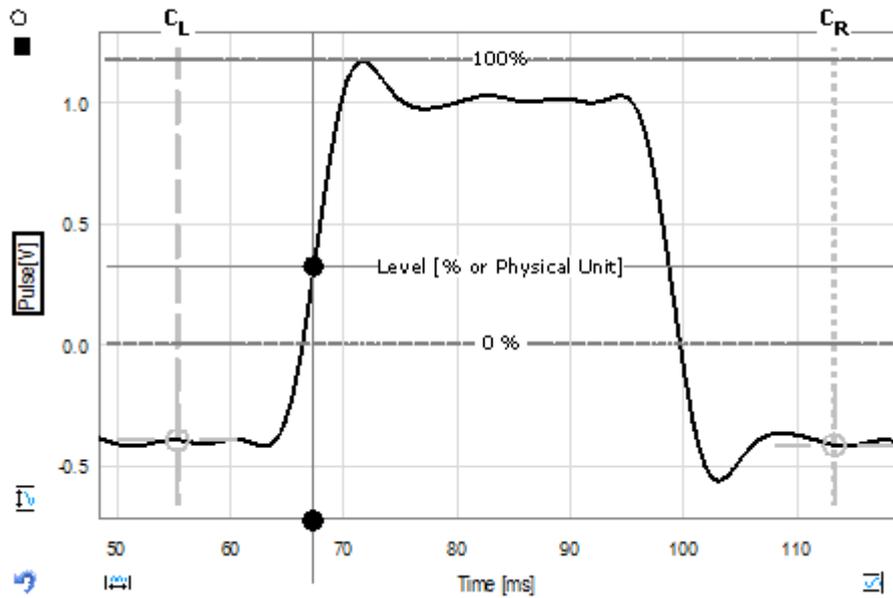
35.3 Group Misc

<p>Area</p>	<p>Calculates the area under the curve between cursors relative to zero level. Values above zero contribute positively to the area, values below zero negatively.</p>  $Area = \sum_{i=C_L}^{C_R} y_i \cdot \Delta t$ <p> C_L: Cursor left C_R: Cursor right y_i: y-value at position i Δt: Sampling Interval </p>
<p>Cadence</p>	<p>Calculates the frequency normalized to one minute.</p> $Cadence = Frequency \cdot 60$ <p><i>Frequency</i>: Calculated Scalar</p>
<p>Energy</p>	<p>Calculates the integral of the squared samples.</p> $Energy = \sum_{i=C_L}^{C_R} y_i^2 \cdot \Delta t$ <p> C_L: Cursor left C_R: Cursor right y_i: y-value at position i Δt: Sampling Interval </p>
<p>FormulaVariable</p>	<p>Inserts a calculated scalar value from a TranAX formula into the Scalar Functions Table.</p>

	<p> Only usefull for Scalar Functions Table B</p>
<p>Number of Blocks</p>	<p>Returns the total number of acquired blocks while the recording is active.</p> <p> This Scalar Function is independent of the cursor position.</p> <p> Number of Blocks is used in the Block- and ECR data acquisition modes.</p>
<p>Number of Triggers</p>	<p>Returns the total number of trigger events.</p> <p> This scalar function is independent of the cursor positions</p> <p> Number of Triggers is used in the Block- and ECR data acquisition modes.</p>
<p>Overflow</p>	<p>Overflow is watching out for positive overload of the ADC of the front-end.</p> <ul style="list-style-type: none"> • Returns 'No' and the background color green if there is no overflow • Returns 'Yes' and the background color red along with the x-axis position when the overflow occurred first
<p>Slope</p>	<p>Calculates the slope of the intersection of the curve bracketed by the cursors.</p> 

	$Slope = \frac{y_{CR} - y_{CL}}{x_{CR} - x_{CL}}$ <p> <i>y_{CL}</i>: y-value at position cursor left <i>y_{CR}</i>: y-value at position cursor right <i>x_{CL}</i>: x-value at position cursor left <i>x_{CR}</i>: x-value at position cursor right </p>
<p>Slope (Lin. Repr.)</p>	<p>Calculates with linear regression the slope of the curve between the cursors.</p>  <p>Slope by Linear Regression</p> <p> <i>C_L</i>: Cursor left <i>C_R</i>: Cursor right </p>
<p>Text</p>	<p>This is a simple placeholder that allows placing custom text and values for reporting reasons and organizing the scalar table.</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;">  Only available for Scalar Functions Table B </div>
<p>Trigger Real Time</p>	<p>Returns the time of the trigger event in either absolute or relative time. In Operation Modes Block- and ECR mode the returned result is from the nearest data block between the selected cursor. In Scope- and Continuous mode only the result in absolute time makes sense.</p>
<p>TxLeft</p>	<p>Returns the time in positive signal pulse where the rising edge crosses a given level starting from the left cursor. The level can be set in percent or as a positive absolute value (physical unit). There result is in relation to the zero on the time axis (depending on the setting "zero at trigger" or "zero at signal start").</p> <p>For the percentage calculation the amplitude value zero of the signal is taken as 0% and the maximum value between the cursors as 100%.</p>

If Abs. Max is selected, the absolute value of the trace will be calculated before the level is applied.



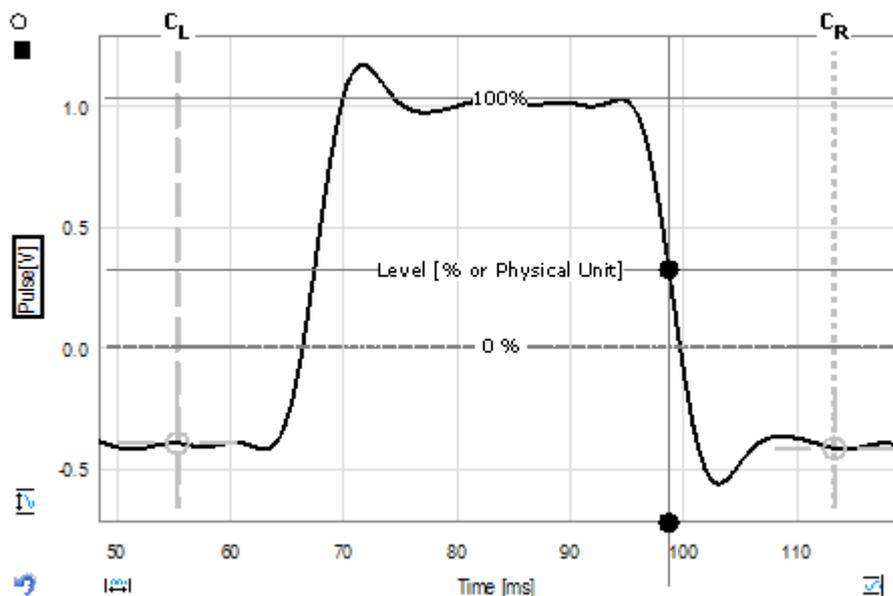
C_L: Cursor left
C_R: Cursor right

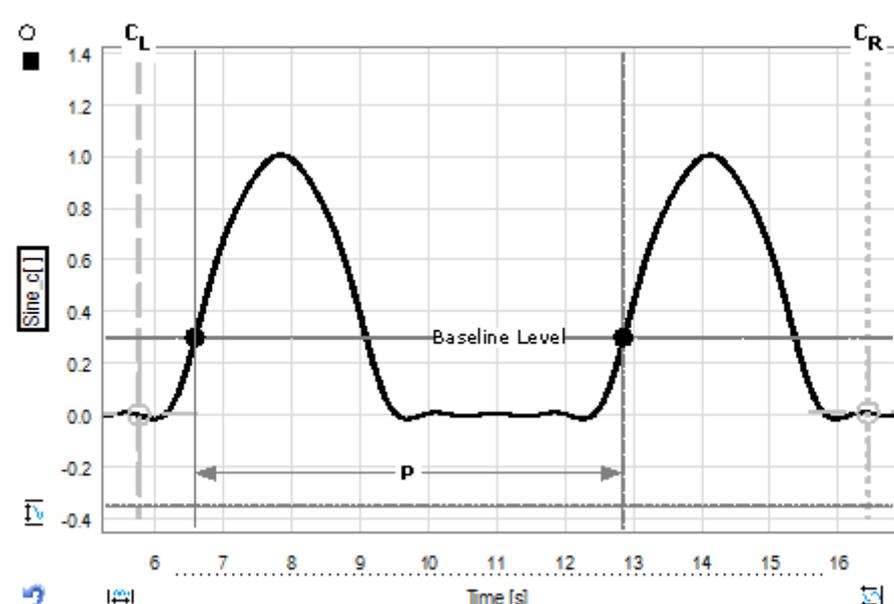
TxRight

Returns the time in positive signal pulse where the rising edge crosses a given level starting from the right cursor. The level can be set in percent or as a positive absolute value (physical unit). There result is in relation to the zero on the time axis (depending on the setting "zero at trigger" or "zero at signal start").

For the percentage calculation the amplitude value zero of the signal is taken as 0% and the maximum value between the cursors as 100%.

If Abs. Max is selected, the absolute value of the trace will be calculated before the level is applied.



	<p>C_L: Cursor left C_R: Cursor right</p>
<p>UnderFlow</p>	<p>Underflow is watching out for negative overload of the ADC of the front-end.</p> <ul style="list-style-type: none"> • Returns 'No' and the background color green if there is no underflow • Returns 'Yes' and the background color red along with the x-axis position when the underflow occurred first
<p>Velocity</p>	<p>Measures the period and calculates the Velocity with the provided Sensor Distance.</p>  <p>$Velocity = \frac{\text{Sensor Distance}}{P}$</p> <p>Sensor Distance: Sensor movement [m] P: Period</p>

35.4 Group Periodic

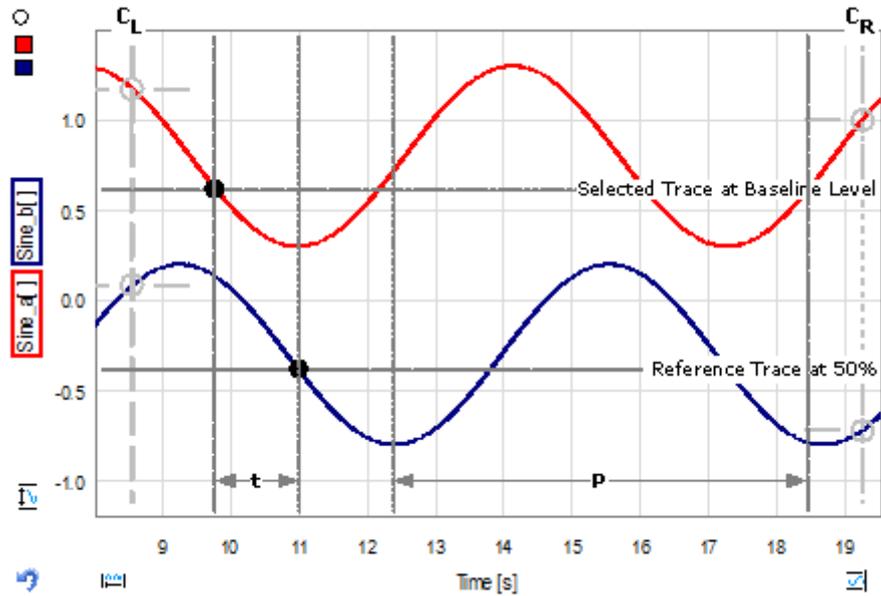
Crest Factor Periodic	<p>Calculates the absolute peak value divided by the periodic RMS. This scalar function is also known as the periodic peak-to-average ratio.</p> $CrestFactor_{Periodic} = \frac{ Peak }{RMS_{Periodic}}$ <p><i>Peak</i>: Absolute Peak <i>RMS_{Periodic}</i>: Periodic Root Mean Square</p>
Mean Periodic	<p>Searches for the baseline level crossing points of the signal and calculates the mean value of completed periods.</p> $Mean_{Periodic} = \frac{1}{n \cdot \Delta t} \cdot \sum_{i=LC_L}^{LC_R} y_i \cdot \Delta t$ <p><i>LC_L</i>: Level crossing left <i>LC_R</i>: Level crossing right <i>n</i>: # of Samples <i>y_i</i>: y-value at position i <i>Δt</i>: Sampling Interval</p>
Rectified Mean Periodic	<p>Searches for the baseline crossing points of the signal and calculates the rectified average value of completed cycles.</p> $Rect_{Periodic} = \frac{1}{n \cdot \Delta t} \cdot \sum_{i=LC_L}^{LC_R} y_i \cdot \Delta t$ <p><i>LC_L</i>: Level Crossing left <i>LC_R</i>: Level Crossing right <i>n</i>: # of Samples <i>y_i</i>: y-value at position i <i>Δt</i>: Sampling Interval</p>
RMS Periodic	<p>Searches for the baseline level crossing points of the signal and calculates the RMS value of completed cycles.</p> $RMS_{Periodic} = \sqrt{\frac{1}{n \cdot \Delta t} \cdot \sum_{i=LC_L}^{LC_R} y_i^2 \cdot \Delta t}$ <p><i>LC_L</i>: Level crossing left <i>LC_R</i>: Level crossing right <i>n</i>: # of Samples <i>y_i</i>: y-value at position i <i>Δt</i>: Sampling Interval</p>

<p>StdDev Periodic</p>	<p>Searches for the baseline level crossing points of the signal and calculates the standard deviation of completed cycles.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  Standard Deviation Periodic is similar to the calculation of the RMS Periodic value with the offset removed. </div> $SDev_{Periodic} = \sqrt{\frac{1}{n \cdot \Delta t} \cdot \sum_{i=LC_L}^{LC_R} (y_i - mean)^2 \cdot \Delta t}$ <p> <i>LC_L</i>: Level Crossing left <i>LC_R</i>: Level Crossing right <i>n</i>: # of Samples <i>y_i</i>: y-value at position i <i>Δt</i>: Sampling Interval </p>
-------------------------------	--

35.5 Group Power

<p>Apparent Power</p>	<p>Calculates the Apparent Power from two periodic signals voltage and current between the selected pair of cursors.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  It is recommended to use Scalar Table B for this function </div> $ApparentPower = RMS_{Periodic}(u) \cdot RMS_{Periodic}(i)$ <p> <i>RMS_{Periodic}</i>: See Scalar RMS Periodic <i>u</i>: Voltage waveform <i>i</i>: Current waveform </p>
<p>Cos(phi)</p>	<p>Calculates the $\cos(\phi)$ at baseline level of the fundamental wave of two different periodic signals with identical frequencies (normally, voltage and current). The reference signal can be selected from the corresponding drop down list and is analyzed at 50% of the amplitude.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  A minimum of 3.5 signal periods between the cursor are required for the calculation. </div> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  There may be a time skew between the voltage waveform and the current waveform caused by the different bandwidths and cable lengths of the probes. Such a skew would result in a phase shift and would need to be corrected before this measurement is performed. The function shift from the Formula Editor may be used to shift horizontally one trace against the other. </div> $\cos(\phi) = \cos\left(360 \cdot \frac{T}{P}\right)$

	<p><i>P</i>: Period <i>T</i>: Time difference of the rising edges</p>
Crest Factor	<p>Calculates the absolute peak value divided by the RMS value. This scalar function is also known as the peak-to-average ratio.</p> $CrestFactor = \frac{ Peak }{RMS}$ <p><i>Peak</i>: Absolute Peak <i>RMS</i>: Root Mean Square</p>
Crest Factor Periodic	<p>Calculates the absolute peak value divided by the periodic RMS. This scalar function is also known as the periodic peak-to-average ratio.</p> $CrestFactor_{Periodic} = \frac{ Peak }{RMS_{Periodic}}$ <p><i>Peak</i>: Absolute Peak <i>RMS_{Periodic}</i>: Periodic Root Mean Square</p>
Fundamental Power	<p>Searches for the baseline level crossing points to determine the fundamental frequency. A hysteresis value is given to prevent noise causing erroneous crossovers measurements. Then it calculates the fundamental frequency power (real power magnitude) of the two traces. It is assumed that one trace is line voltage and the second trace is line current.</p>
Phase	<p>Returns the phase difference in degrees between the signal analyzed and the signal used as a reference of two periodic signals with identical frequency (usually voltage and current). The reference signal that can be selected from the corresponding drop down list is analyzed at 50% of the amplitude.</p> <div style="background-color: #f0f0f0; padding: 5px; border: 1px solid #ccc;">  <p>A min. of 3.5 signal periods are required between the cursors to calculate the Phase. A negative result will be returned in the range of $\Phi \geq \pm 180^\circ$, if the reference signal lags the other one.</p> </div>



$$Phase = 360 \cdot \frac{t}{P}$$

C_L : Cursor left

C_R : Cursor right

P : Period

t : Time difference at baseline level to 50%

Power Factor

Returns the power factor of two periodic signals with identical frequency (usually voltage and current). The reference signal can be selected from the corresponding drop down list.



It is recommended to use Scalar Table B for this function



This calculation needs min. 1.5 signal periods.



There may be a time skew between the voltage waveform and the current waveform caused by the different bandwidths and cable lengths of the probes. Such a skew would result in a phase shift and would need to be corrected before this measurement is performed. The function shift from the Formula Editor may be used to shift horizontally one trace against the other.

$$Power\ Factor = \frac{P}{S}$$

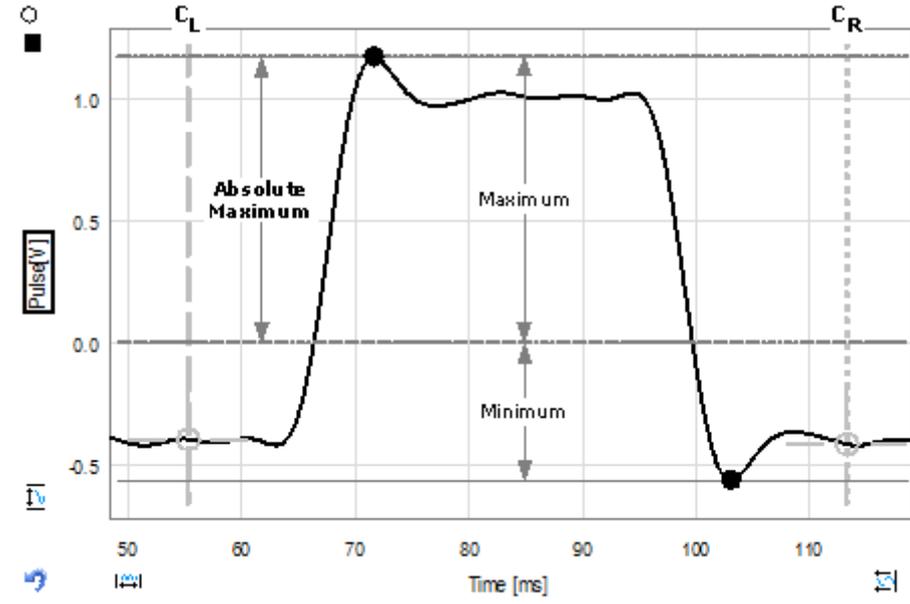
P : Real Power

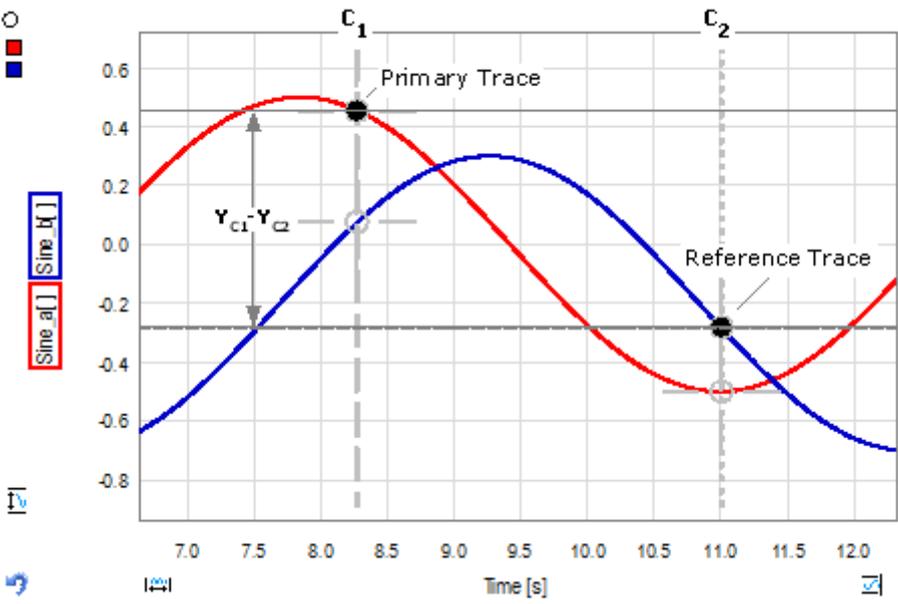
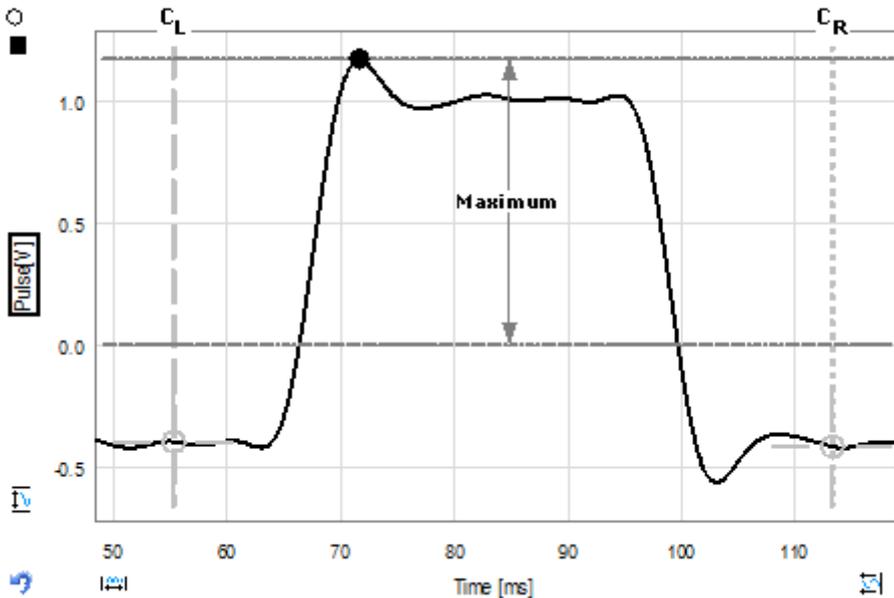
S : Apparent Power

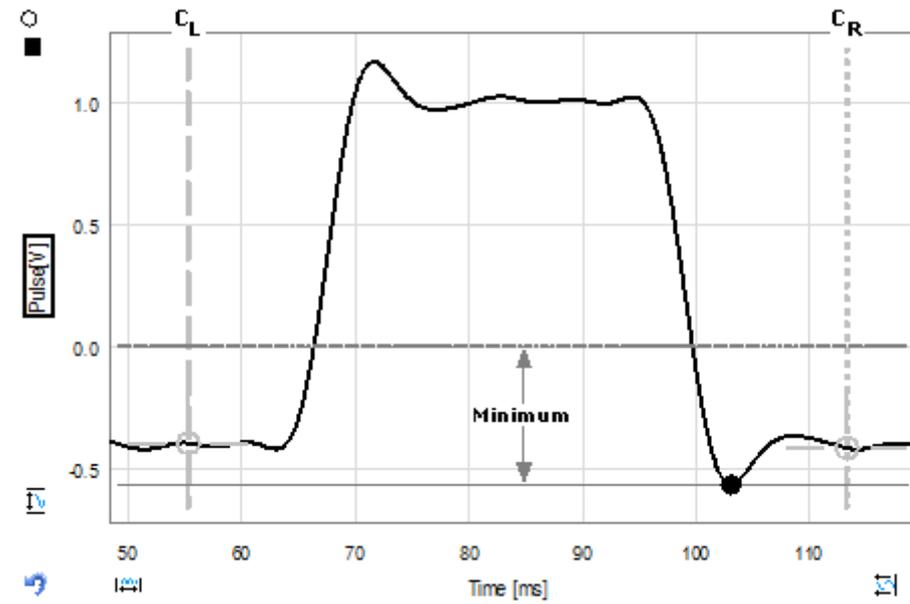
Reactive Power	<p>Calculates the Reactive Power from Apparent Power and Real Power.</p> <div data-bbox="368 212 1233 353" style="background-color: #f0f0f0; padding: 5px;">  It is recommended to use Scalar Table B for this function. </div> $P_{react} = RMS_{Periodic}(u) \cdot RMS_{Periodic}(i) \cdot \sin(\Delta\varphi)$ <p><i>RMS_{Periodic}</i>: See Scalar RMS Periodic</p> <p><i>u</i>: Voltage <i>i</i>: Current $\Delta\varphi$: Phase shift between voltage and current</p>
Real Power	<p>Calculates the Real Power from the instantaneous power.</p> <div data-bbox="368 734 1233 875" style="background-color: #f0f0f0; padding: 5px;">  It is recommended to use Scalar Table B for this function </div> $P_{real} = RMS_{Periodic}(u) \cdot RMS_{Periodic}(i) \cdot \cos(\Delta\varphi)$ <p><i>RMS_{Periodic}</i>: See Scalar RMS Periodic</p> <p><i>u</i>: Voltage <i>i</i>: Current $\Delta\varphi$: Phase shift between voltage and current</p>
Total Harmonic Distortion	<p>Searches for the fundamental frequency and calculates the Total Harmonic Distortion (THD) in % of a periodic signal.</p> <div data-bbox="368 1301 1345 1581" style="background-color: #f0f0f0; padding: 5px;">  A minimum of 2 signal period between the cursors is required for the calculation.  Noisy signals may need to be low pass filtered before applying this scalar function and the baseline level and hysteresis need to be set carefully. </div> $THD_U = \frac{\sqrt{U_2^2 + U_3^2 + U_4^2 + \dots + U_n^2}}{U_1} [\%]$ $THD_I = \frac{\sqrt{I_2^2 + I_3^2 + I_4^2 + \dots + I_n^2}}{I_1} [\%]$ $n = \min\left(50, \frac{\text{samplingFrequency}}{2 \cdot \text{baseFrequency}} - 1\right)$ <p><i>U</i>₁: Voltage at fundamental frequency <i>I</i>₁: Current at fundamental frequency</p>

	U_x : x^{th} harmonic of the voltage signal I_x : x^{th} harmonic of the current signal
--	--

35.6 Group Vertical

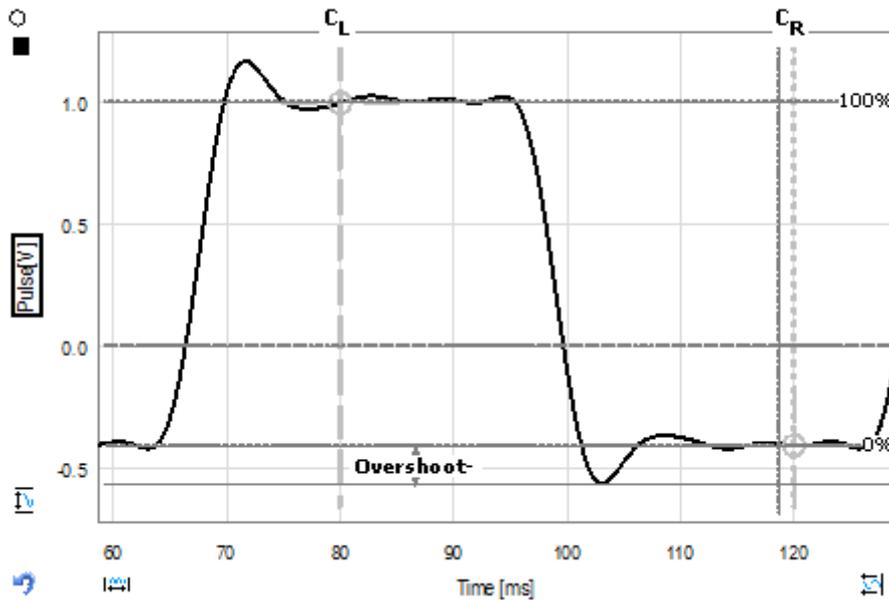
<p>Abs.Max</p>	<p>Calculates the absolute values of the maximum and the minimum and returns the higher of the two.</p>  $Abs.Max = \begin{cases} Maximum & \text{if } Maximum \geq Minimum \\ Minimum & \text{if } Minimum > Maximum \end{cases}$ <p>C_L: Cursor left C_R: Cursor right</p>
<p>Delta</p>	<p>Calculates the difference of the cursor readout from cursor C1 of the primary trace to the cursor readout from cursor C2 of the reference trace.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p>i The Delta function max be combined with a pair of cursors from different traces.</p> </div>

	 <p>$\Delta = y_{C1 \text{ Primary Trace}} - y_{C2 \text{ Reference Trace}}$</p> <p>$C_1$: Selected Cursor 1 C_2: Selected Cursor 2</p>
<p>Frequency at Maximum (FFT)</p>	<p>Finds the maximum between Cursor A and Cursor B and returns the frequency at the maximum.</p> <div style="border: 1px solid gray; padding: 5px; background-color: #f0f0f0;">  This is only for scalar spectra (FFT) are available </div>
<p>Maximum</p>	 <p>C_L: Cursor left</p>

	<p>C_R: Cursor right</p>
Mean	<p>Calculates the average value of the waveform.</p> $Mean = \frac{1}{n \cdot \Delta t} \cdot \sum_{i=C_L}^{C_R} y_i \cdot \Delta t$ <p> C_L: Cursor left C_R: Cursor right n: # of Samples y_i: y-value at position i Δt: Sampling Interval </p>
Mean Periodic	<p>Searches for the baseline level crossing points of the signal and calculates the mean value of completed periods.</p> $Mean_{Periodic} = \frac{1}{n \cdot \Delta t} \cdot \sum_{i=LC_L}^{LC_R} y_i \cdot \Delta t$ <p> LC_L: Level crossing left LC_R: Level crossing right n: # of Samples y_i: y-value at position i Δt: Sampling Interval </p>
Minimum	<p>Measures the lowest value in the waveform relativ to the signal zero-line.</p>  <p> C_L: Cursor left C_R: Cursor right </p>
Overshoot-	<p>Returns the negative overshoot.</p>



Note that the intersection point of the left cursor is the top value (100%) and the intersection point of the right cursor is the base value (0%) of the waveform.



C_L : Cursor left (Top at 100%)

C_R : Cursor right (Base at 0%)

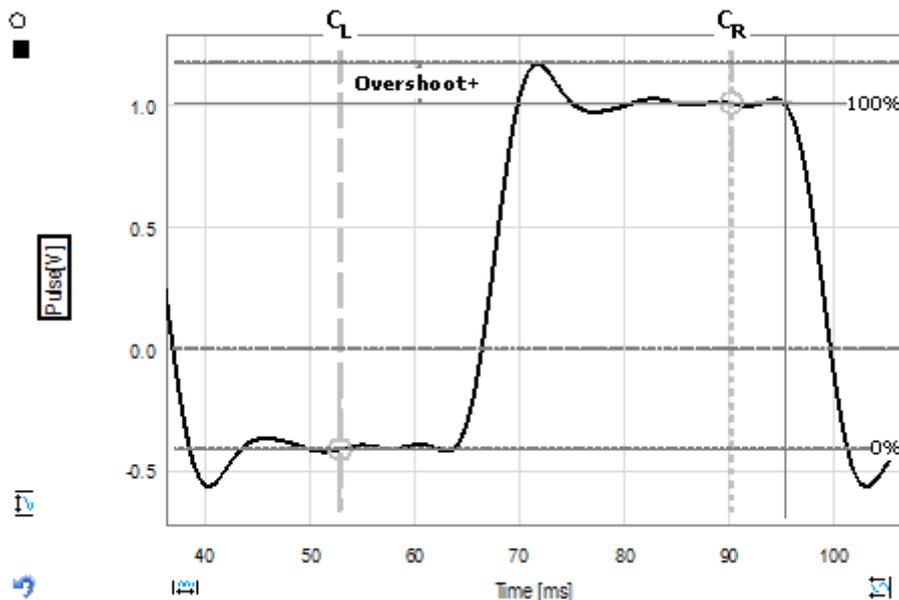
Overshoot-: Negative overshoot [%]

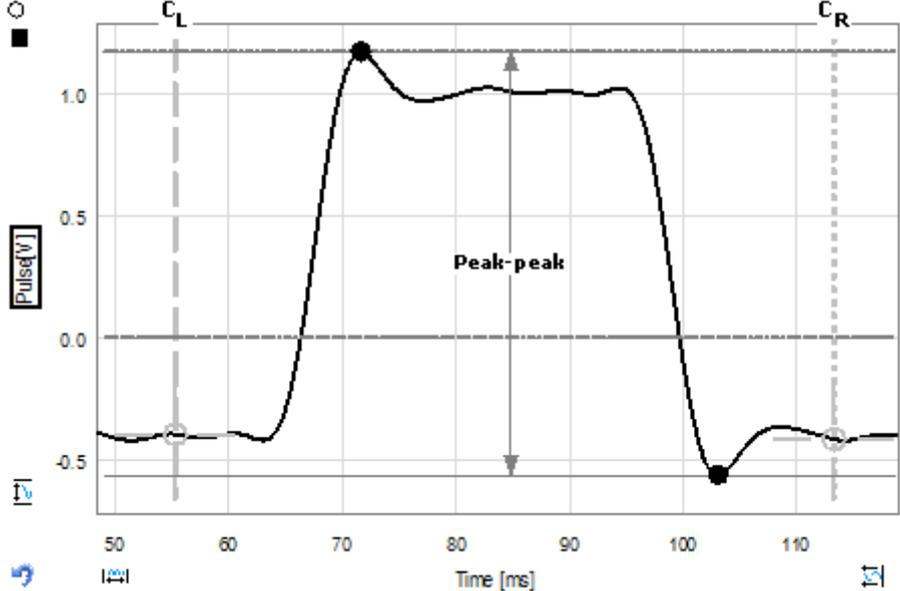
Overshoot+

Returns the positive overshoot.



Note that the intersection point of the left cursor is the base value (0%) and the intersection point of the right cursor is the top value (100%) of the waveform.



	<p>C_L: Cursor left (Base at 0%)</p> <p>C_R: Cursor right (Top at 100%)</p> <p>Overshoot+: Positive overshoot [%]</p>
Peak-Peak	<p>Calculates the difference between the maximum and minimum value in the waveform.</p>  <p>Peak-Peak = Maximum – Minimum</p> <p>C_L: Cursor left</p> <p>C_R: Cursor right</p>
Rectified Mean	<p>Calculates the rectified average of the signal.</p> $Rect = \frac{1}{n \cdot \Delta t} \cdot \sum_{i=C_L}^{C_R} y_i \cdot \Delta t$ <p>C_L: Cursor left</p> <p>C_R: Cursor right</p> <p>n: # of Samples</p> <p>y_i: y-value at position i</p> <p>Δt: Sampling Interval</p>
Rectified Mean Periodic	<p>Searches for the baseline crossing points of the signal and calculates the rectified average value of completed cycles.</p> $Rect_{Periodic} = \frac{1}{n \cdot \Delta t} \cdot \sum_{i=LC_L}^{LC_R} y_i \cdot \Delta t$ <p>LC_L: Level Crossing left</p> <p>LC_R: Level Crossing right</p> <p>n: # of Samples</p>

	<p>y_i: y-value at position i Δt: Sampling Interval</p>
Rightmost Value	<p>Returns the rightmost value of the current trace in the data acquisition memory. This Function is especially useful in the continuous data acquisition mode to display steadily the actual value of the current measurement.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p> This Scalar is not depending on the cursor position.</p> <p> This Scalar is updated even if the recording is active.</p> </div>
RMS	<p>Calculates the root mean square of the signal.</p> $RMS = \sqrt{\frac{1}{n \cdot \Delta t} \cdot \sum_{i=C_L}^{C_R} y_i^2 \cdot \Delta t}$ <p> C_L: Cursor left C_R: Cursor right n: # of Samples y_i: y-value at position i Δt: Sampling Interval </p>
RMS Periodic	<p>Searches for the baseline level crossing points of the signal and calculates the RMS value of completed cycles.</p> $RMS_{Periodic} = \sqrt{\frac{1}{n \cdot \Delta t} \cdot \sum_{i=LC_L}^{LC_R} y_i^2 \cdot \Delta t}$ <p> LC_L: Level crossing left LC_R: Level crossing right n: # of Samples y_i: y-value at position i Δt: Sampling Interval </p>
StdDev	<p>Calculates the Standard Deviation of the signal.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p> Standard Deviation is similar to the calculation of the RMS value with the offset removed.</p> </div> $SDev = \sqrt{\frac{1}{n \cdot \Delta t} \cdot \sum_{i=C_L}^{C_R} (y_i - mean)^2 \cdot \Delta t}$ <p>C_L: Cursor left</p>

	<p>C_R: Cursor right n: # of Samples y_i: y-value at position i Δt: Sampling Interval</p>
<p>StdDev Periodic</p>	<p>Searches for the baseline level crossing points of the signal and calculates the standard deviation of completed cycles.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> Standard Deviation Periodic is similar to the calculation of the RMS Periodic value with the offset removed.</p> </div> $SDev_{Periodic} = \sqrt{\frac{1}{n \cdot \Delta t} \cdot \sum_{i=LC_L}^{LC_R} (y_i - mean)^2 \cdot \Delta t}$ <p>LC_L: Level Crossing left LC_R: Level Crossing right n: # of Samples y_i: y-value at position i Δt: Sampling Interval</p>

36 Miscellaneous

Further information and description of functions and features:

36.1 ActiveX/COM- Interface



Active-X/COM - Interface is a software option

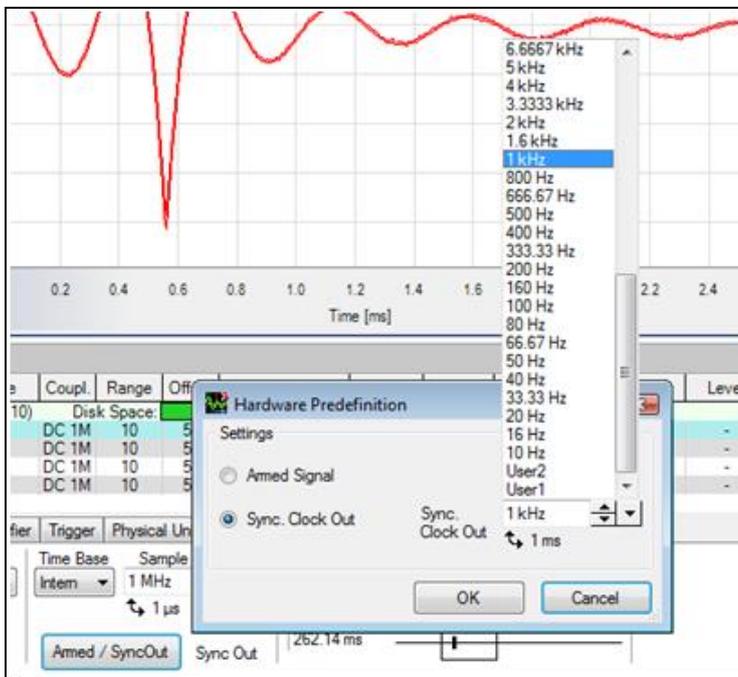
Microsoft created an Interface called COM (Component Object Model), which gives applications and components the possibility to communicate with each other. This feature can be used for remote control TranAX with other applications (e.g. BallAX or Excel Visual Basic).



Using and settings of the ActiveX/COM parameters are described in the corresponding manual.

36.2 Sync.Clock Out

From Star Hub firmware version 4 upwards, the **Sync.Clock Out** (Synchronization Clock Output) can be used for **synchronizing** external devices like high-speed cameras.



The frequency for the Sync.Clock output can be selected from a **dropdown list** or entered manually in the Main Section of the Control Panel. The entered value will be rounded to the next fitting value.

Acceptable clock frequencies run from **10Hz up to 10MHz**. The two entries "User 1" and "User 2" at the beginning of the list are reserved entries and can be **custom configured** at factory.

Click the button "**Armed/SyncOut**" in the **Control Panel** tab "**Main**" to open the "**Hardware**

predefinition" Dialog. Two radio buttons allow to switch between the common settings (**Armed signal**) or the function **Sync.Clock.Out**. Normally the Sync.Clock.Out respectively the Armed signal is available at the 25pol D-Sub connector. Please see the **hardware manual** for more detailed information.



For using **Sync.Clock.Out functionality**, the installed software may need to be upgraded. The following **versions** are **prerequisites**:

TranAX:	3.2.1.702	(Menu " Help " / " About ")
TPC-Server:	1.3.5	(Control Panel / )
Star Hub:	4	(TraNetConfiguration.exe / Show Logfile / Server)

An update of the Star Hub firmware has to be done by manufacturer!



For **higher frequencies** (above 1MHz) the cable length needs to be taken into account (< 2m at 10MHz). Or else the signals may not be used reliably with the synchronized device.



The frequencies for "User 1" and "user 2" have to be programmed to the Star Hub with **customer specific software** and can't be changed ad hoc.

For Changes (also for updates for devices delivered before 2012) the Star Hub has to be sent back to the manufacturer or its representative.

36.3 Command line parameter

TranAX can also be started in a batch file. The existing parameters are meant for practiced users and give a lot of opportunities and possibilities.



This section is for advanced users and requires substantial awareness in the usage of the DOS prompt and batch files. This is a coarse overview about the possible commands. Depending on the Windows version in use, some differences are possible.

Refer to your internal IT department for support and more information about the usage of batch files.

To get a list of all available parameters, open DOS, by going to the directory of TranAX (usually "C:\Program Files\Elsys\TranAX_x.x") and enter ***TranAX.exe -help***

The table below shows the published and available parameters. These will be independent of the language settings of TranAX, i.e., always be written in English.

Parameter	Description and example
-allsettings	Loads all settings from the specific path. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE: TranAX.exe -allsettings="C:\YOURPATH\MyAllSettings.lay"
-autoseq	Loads the autosequence from the specific path and starts it. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE: TranAX.exe -autoseq="C:\YOURPATH\MyAutoSeq.aut"
-cachesize	Limits the data cache to the specified amount of bytes (default 150'000'000) EXAMPLE: TranAX.exe -cachesize=200000000
-device	Sets url of device(s) to use EXAMPLE: TranAX.exe -device=192.168.0.102:10010
-experiment	Opens the given Experiment. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE: TranAX.exe -experiment="C:\YOURPATH\Experiment.exp"
-experimentSet	Opens the given experiment set. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE:

	TranAX.exe -experiment="C:\YOURPATH\Experiment.exp\Experiment.zip"
-formula	Loads the formula file from the specific path. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE: TranAX.exe -formula="C:\YOURPATH\MyFormula.for"
-help	Shows all commands EXAMPLE: TranAX.exe -help
-info	Shows metadata of a TPC5-File. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE: TranAX.exe -info=C:\YOURPATH\test.tpc5
-layout	Loads the layout from a specific path. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE: TranAX.exe -layout="C:\YOURPATH\MyLayout.lay"
-recording	Loads the recording settings from the specific path. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE: TranAX.exe -recording="C:\YOURPATH\MyRecording.tps.xml"
-server	Acts as remote server (on port 12668) EXAMPLE: TranAX.exe -server
-serverport	Acts as remote server with a defined port EXAMPLE: TranAX.exe -serverport=12000
-title	Sets the title of the TranAX program. This could be necessary if you open several TranAX program EXAMPLE: TranAX.exe -title=VoltageMeasurement
-version	Shows the current build version of TranAX EXAMPLE: TranAX.exe -version
-view	Adds a file to the signal source browser. If the file name or directory path has blanks you have to write the whole path with quotes EXAMPLE: TranAX.exe -view="C:\YOURPATH\test.tpc5"
-offline	Starts TranAX without connecting to a device EXAMPLE: TranAX.exe -offline
-viewer	Starts TranAX as a viewer. The user can not manipulate measurement configurations (Start, stop, etc.) EXAMPLE: TranAX.exe -viewer
-scope	Starts TranAX in fullscreen mode with a Scope-Display EXAMPLE: TranAX.exe -scope
-fullscreen	Starts TranAX in fullscreen mode EXAMPLE: TranAX.exe -fullscreen

36.4 Create shortcuts

You also may run TranAX via (Desktop) shortcuts or a batch file. To load TranAX directly with an existing Experiment, add ***-experimentset="experimentset name"*** to the command line. If it is required that an auto sequence runs directly after load, then the auto sequence name can be added: ***-autosequence="autosequence name"***

Example file "startTranAX.bat":

```
set TRANAX="C:\Program Files\Elsys\TranAX_4.1\TranAX.exe"  
set EXPSET=". \myExperiment.exp\myExperimentSet.zip"  
  
start "TranAX" %TRANAX% -experimentset=%EXPSET%
```

36.5 Limitations

The following discussion covers settings which are subject to certain limitations.

36.5.1 Digital inputs (markers)

Digital inputs are only available when that **optional hardware is installed**. A marker signal always corresponds to an analog channel name. Channels equipped with the option have 2 markers each.

With the 16-bit modules, markers can only be recorded if the ADC is set to 14-bit (see [Averaging](#)). Marker signals cannot be used as trigger, however one external trigger input per module or per instrument is available. External trigger is controlled by settings on the [Trigger](#) tab.

36.5.2 Differential inputs

With the single ended modules switchable to differential inputs (e.g. TPCX-2014-**8S**) the following channel pairing arrangements are used for **differential input** connections: (1,2), (3,4), (5,6), (7,8). In this case the even numbered channels are not programmable. With the differential modules (e.g. TPCX-2014-**8D**) all channels are equipped with two BNC connectors; therefore the even numbered channels also have real differential inputs. These settings are described in the [Input Amplifier](#) section.

36.5.3 Maximum Sample rate

The **16-bit amplitude resolution** (with the 16-bit modules such as TPCX-4016-4D) is only applicable at **sample rates 1/4 of the maximum sample rate or slower**.

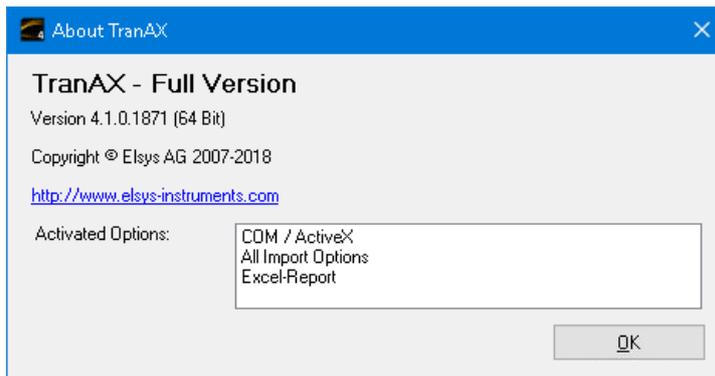
Example: With a TPCX-4016-4x module (40 MHz), the 16-bit resolution only works for sample rates equal or slower than 10 MHz. Above 10 MHz to 40 MHz, the ADC converts only 14-bit.

37 Trouble Shooting

To solve failures and other performance issues, technical support must be provided with most adequate and detailed information. This chapter describes how problems can be solved itself and what data should be given for any inquiries.

37.1 TranAX Software version

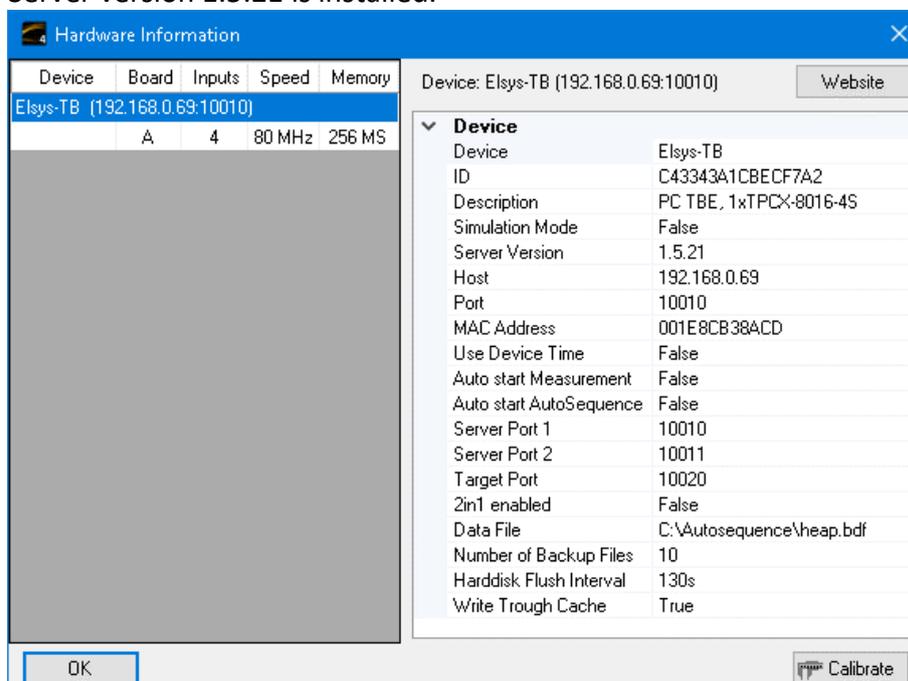
To capture a measurement in TranAX, several software components interact with each other. TranAX communicates with the TPC-Server, the TPC-Server with the driver and the driver works with the TPCX/TPCE modules.



To see the currently installed version of TranAX, please click the menu **"Help" / "Info"**. In the example on the left side, version 4.1.1871 is installed.

37.2 TPC-Server Version

To get the version of the actual installed TPC-Server, please click the information button  in the Control Panel. The hardware information dialog will be opened. In the example below, TPC-Server version 1.5.21 is installed.



37.3 Driver and Firmware Version

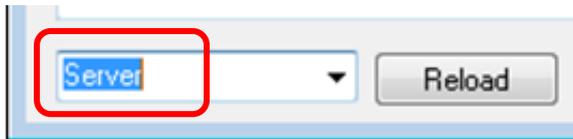
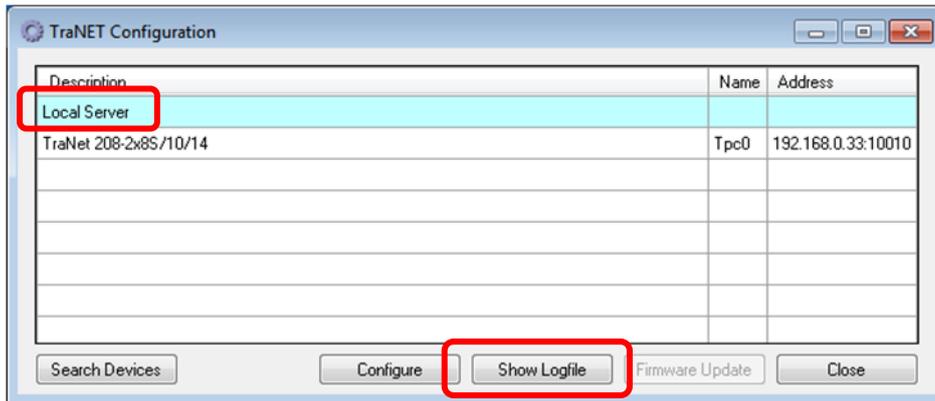
To get the version of the driver, Firmware of TPCX/ TPCE modules and of the Starhub, the application TraNetConfiguration.exe has to be started. Please double click the icon on the desktop.



Select the device in the TraNET Configuration dialog "Local Server" or one of the listed TraNET FE devices. Then click the button "Show Logfile".



There are only TraNET FE devices and local devices listed. To get the information from other TraNET EPC or PPC devices, please check the version on these systems directly.



Please make sure that "Server" is selected in the dropdown menu. In the Textbox above all the information from the selected TraNET system are listed.



An update of the **TPC-Server** includes the **TPC-Server software itself, the Firmware of the TPCX/TPCE module and the driver for Windows.**

An update of TranAX includes just the Analytic software TranAX itself.



In case of questions and troubles please save the server settings by clicking the "Save" button and send this file together with a short description of the problem to technical support



Generally, it is recommended to install **all of the software on a CD** to prevent possible incompatibilities between different software versions.

37.3.1 Example Windows 7

```
Server number 0 started
=====
Operation System: Microsoft Windows 7
Server version: 10305
=====
Board number 0 found
Driver Version 2.3.3 found
I2C Delay Factor: 4
Firmware "Version A.1.6.2 Overload Reset changed" found
No Synchronisation found
```

This example is a Windows 7 System with the same installed software as the example above with Windows XP. The message "**No Synchronisation found**" means that there is either **no Starhub installed**, or **not connected properly**, or maybe **damaged**.

37.3.2 TraNET FE

```
Server number 0 started
=====
Model Type: TraNET-FE
Operation System: Linux
Server version: 10305
=====
Board number 0 found
Driver Version 2.3.2 found
I2C Delay Factor: 10
Firmware "Version A.1.6.2 Overload Reset changed" found
Starhub Version 1 found
```

In the third line **Model Type: TraNET FE** is written. The installed operating system is **Linux**. Overall, the information is the same, except that here the **Starhub** has firmware **version 1**.

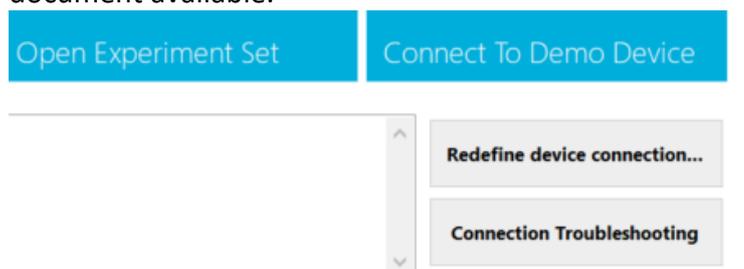
37.4 Error Messages

37.4.1 TranAX

The following error messages can appear in the Control Panel of TranAX:

Message	Reason	Solution
<i>"Hardware failure"</i>	no TPCX/TPCE module found	Install a TPCX/TPXE module or change the settings of the TPC-Server to Demo Mode.
	TPCX/TPCE- modules not properly installed	Check if the TPCX/TPCE module is mounted correctly and is installed tight into the PCI/PCIe slot. The computer has to be powered off for mounting a module.
	Driver not installed	Install the whole TPC-Server software. There is a CD in the cover of the manual. A reboot if the system will be required.
	PCI/PCIe slot damaged	Switch the PCI/PCIe slot of the module to be sure the used slot is not damaged.
	TPCX/TPCE module damaged	In case that none of the upper solutions are working, the TPCX/TPCE module has to be sent back to manufacturer for further analyses.
<i>"Network Error"</i>	Ethernet cable not connected	plug in the network cable, depending on the kind of connection maybe a crossed Ethernet cable has to be used.
	TraNET FE or EPC not started	Plug in the power cord and start the system up. TraNET FE signals a running system when the green LED "Ready" on the front panel is flashing.
	Wrong IP Address	Make sure that TranAX is connected to the correct device or local address.
	Wrong IP Port	Make sure that the correct IP-Port is selected. TranAX has to connect to the same Port as the TraNET device provides, check also the settings of the TraNET device.
	Network collisions	Make sure that every device in the network has its own IP-Address which is not used by any other device in the local network. You may have to contact your internal IT support to assign an address to the TraNET device.

In the Startup Page of TranAX, there is a Connection Troubleshooting Guide available as a PDF document available.



37.4.2 TraNET Config Logfile

More important information about the TraNET system can be found in the logfile of the TPC-Server. This can be opened with the program "TraNetConfigurator.exe".

Message	Reason	Solution
<i>"No Synchronisation found"</i>	No Star hub installed	No Error
	Star hub not connected	Check the cables between the Star hub and the TPCX/TPCE modules.
	Star hub damaged	In case none of the upper solutions work, the Star hub has to be sent back to manufacturer for repair.
No entries found like: <i>"Board number x found"</i>	There is no TPCX/TPCE module installed	Install a TPCX/TPXE module or change the settings of the TPC-Server to Demo Mode.
	TPCX/TPCE- modules not properly installed	Check if the TPCX/TPCE module is mounted correctly and is installed tight into the PCI/PCIe slot. The computer has to be powered off prior to mounting a module.
	Driver not installed	Install the entire TPC-Server software. There is a CD in the cover of the manual. A reboot of the system will be required.
	PCI/PCIe slot damaged	Switch to a different PCI/PCIe slot for the module to see if the failure is persistent.
	TPCX/TPCE module damaged	In case that none of the upper solutions are working, the TPCX/TPCE module has to be sent back to manufacturer for further analyses.
No entries found like: <i>"Server number 0 started"</i>	TPC-Server not installed or a faulty installation was made	Install the entire TPC-Server software. There is a CD in the cover of the manual. A reboot of the system will be required.

37.4.3 TranAX firewall and port settings

TranAX communicates over TCP/IP with the TPCServer. This is running on Windows based like EPC/PPC, as also on external devices like TraNET FE.

- Device Finder of TranAX uses Port 10020 UDP for searching for devices
- TranAX uses Port 10010 TCP for communication and data Transfer with devices



Make sure that **port 10020 UPD** and **port 10010 TPC** are activated for TranAX software in **both directions**. Windows standard firewall will be configured automatically.

Document information

TranAX version: 4.1.2.2320
Filename: TranAX_4.1.2_Manual_EN.docx
Status: Released

Last modification

Date: 20.05.2020
Author: MM/TBE