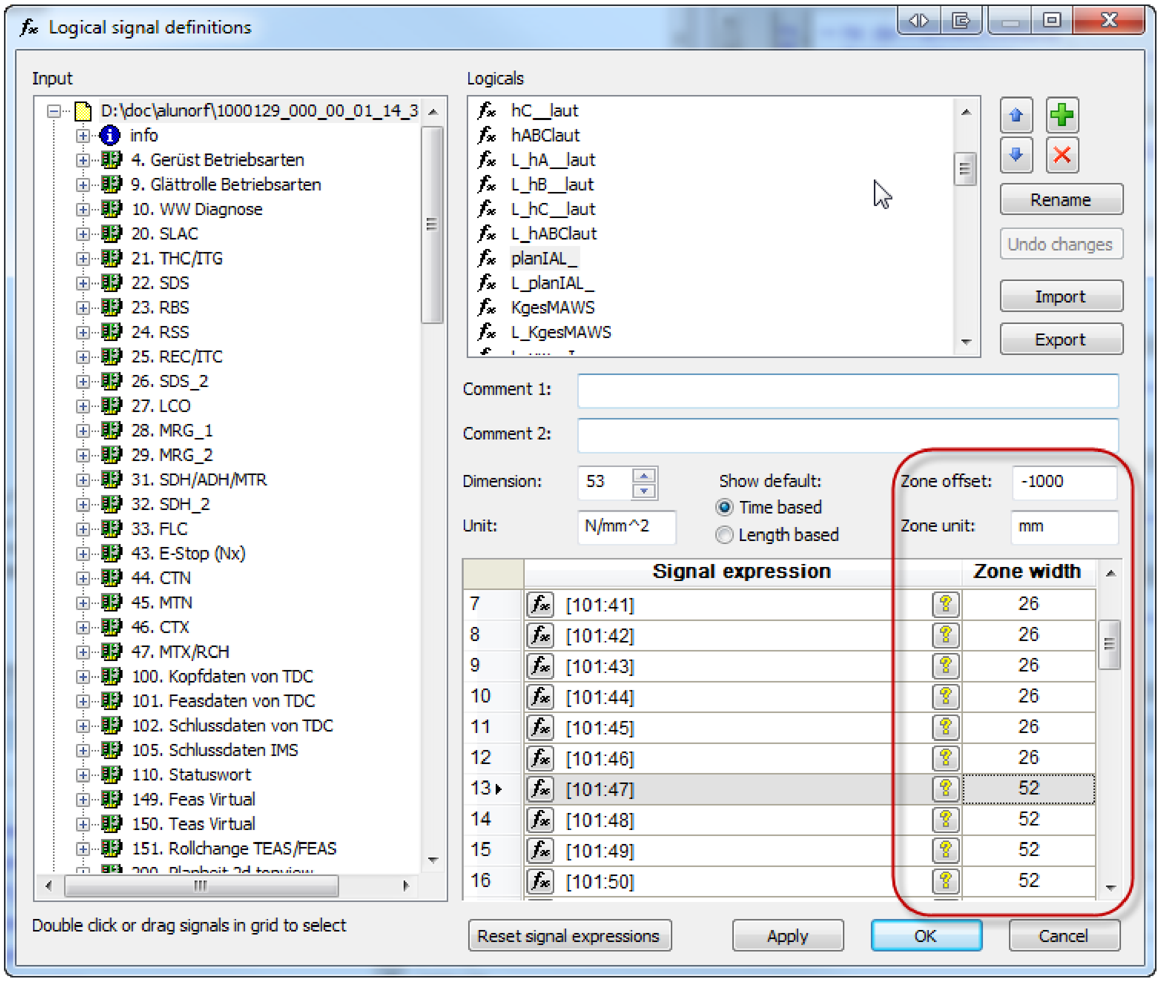
New features in ibaAnalyzer v6.1.0

1 Vectors with zone widths

In older versions of ibaAnalyzer, multidimensional signals (vectors) were shown both in the 2D top view and 3D views with each sub signal at an equidistant space from its neighboring sub signals. Also in the 2D top view, the Y axis showed the index number of the signal (i.e. simply 0, 1, 2, 3 …) rather than any actual distance between the sub signals, while in the 3D view, no axis was shown at all.
In the current version of ibaAnalyzer, the sub signals are considered centered in a 'zone'. For each zone a width can be specified and for the first sub signal, an offset to the center of its corresponding zone can be specified.
This allows for correctly annotating the Y axis in the 2D top view and adding the same axis in 3D views. Additionally the zones do not need to be equidistantly spaced, this allows one to create profile plots that more accurately depict measurements where the measuring locations along the profile are not equidistantly spaced either.

1.1 Specifying zone widths

In the logicals dialog, a new column is added to the expression grid where one can specify the zone widths for each sub signal. Also the offset to the 1st zone center can be specified, as well as the unit of the offset and widths.



Input:
- D:\doc\alunorf\1000129_000_00_01_14_3
  - info
  - 4. Gerüst Betriebsarten
  - 9. Glättrolle Betriebsarten
  - 10. WW Diagnose
  - 20. SLAC
  - 21. THC/ITG
  - 22. SDS
  - 23. RBS
  - 24. RSS
  - 25. REC/ITC
  - 26. SDS_2
  - 27. LCO
  - 28. MRG_1
  - 29. MRG_2
  - 31. SDH/ADH/MTR
  - 32. SDH_2
  - 33. FLC
  - 43. E-Stop (Nx)
  - 44. CTN
  - 45. MTN
  - 46. CTX
  - 47. MTX/RCH
  - 100. Kopfdaten von TDC
  - 101. Feasdaten von TDC
  - 102. Schlussdaten von TDC
  - 105. Schlussdaten IMS
  - 110. Statuswort
  - 149. Feas Virtual
  - 150. Teas Virtual
  - 151. Rollchange TEAS/FEAS

Logicals:
- hC__laut
- hABClaut
- L_hA__laut
- L_hB__laut
- L_hC__laut
- L_hABClaut
- planIAL_
- L_planIAL_
- KgesMAWS
- L_KgesMAWS

Comment 1:
Comment 2:
Dimension: 53
Unit: N/mm^2
Show default: [✓ Time based] Length based
Zone offset: -1000
Zone unit: mm

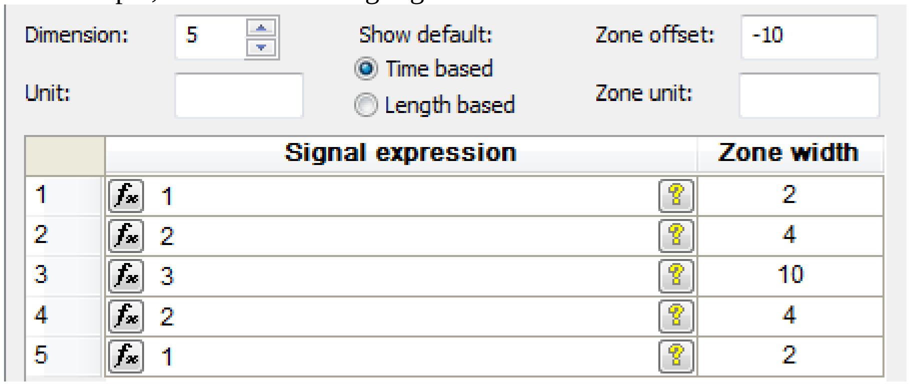
	Signal expression	Zone width
7	[101:41]	26
8	[101:42]	26
9	[101:43]	26
10	[101:44]	26
11	[101:45]	26
12	[101:46]	26
13	[101:47]	52
14	[101:48]	52
15	[101:49]	52
16	[101:50]	52

Double click or drag signals in grid to select

The zone widths need to be positive numeric constants; attempts to type negative values, zero or a text that doesn't parse as a number will be rejected.
The offset to the first zone center needs to be a numeric constant as well but can be negative.
By default, the offset is 0 and all widths are 1. If these values are not altered, they will cause ibaAnalyzer to draw a 2D top view plot of the logical entirely identically as in previous versions of ibaAnalyzer, while a 3D plot will be identical as well with the exception of a new axis that will denote the indices of the subsignals (0, 1, 2, 3 …).
If a large number of consecutive identical widths need to be entered, one can speed up the process by clicking the 'Zone width' header. This causes the zone width of the current selected row in the expression grid to be copied in all rows below it.

1.2 2D top view

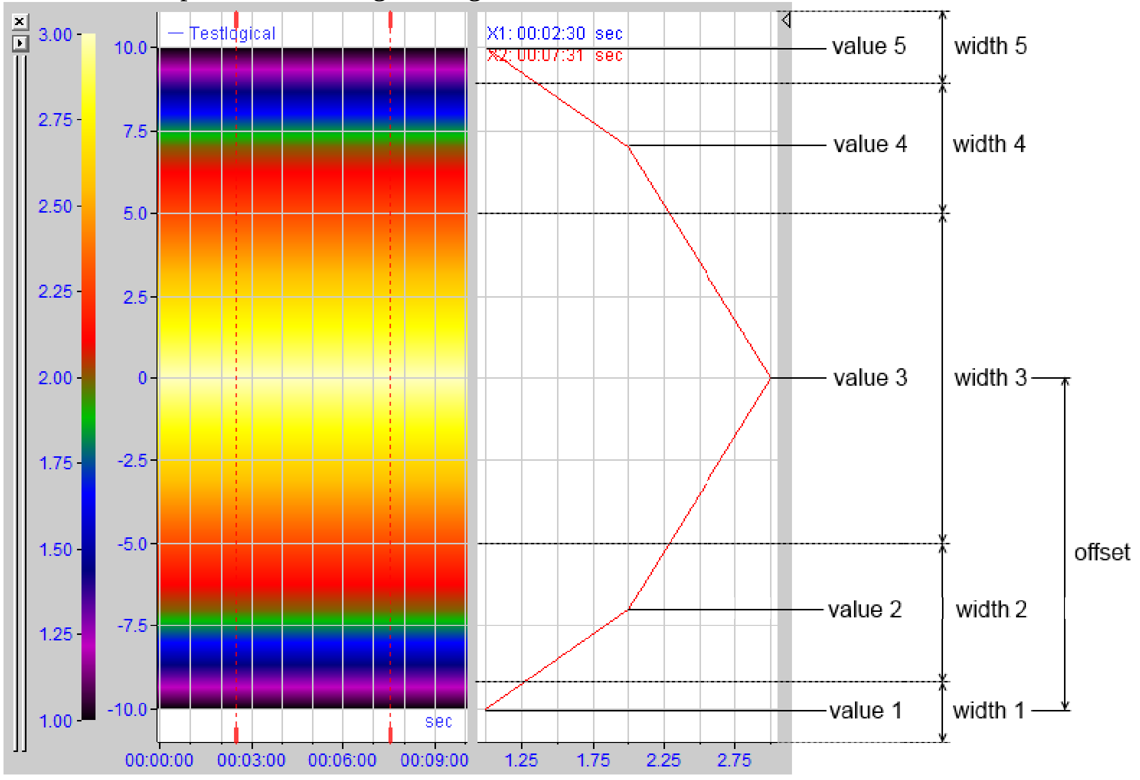
The effect of the zone widths is easiest to explain for the 2D top view.
For example, take the following logical:



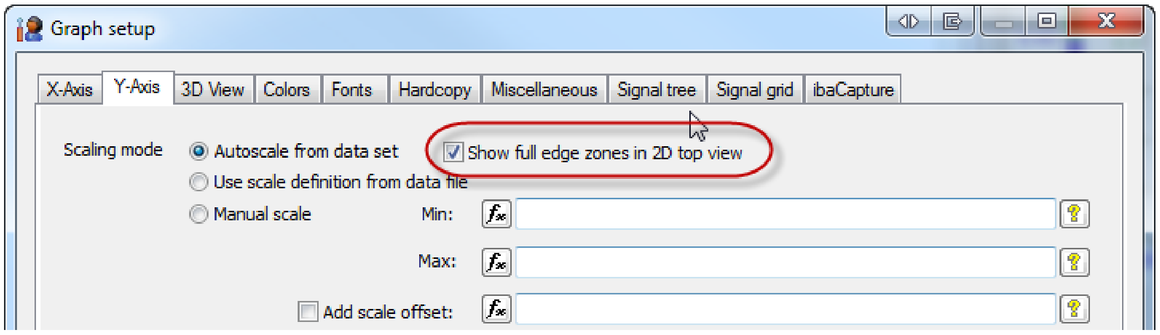
Dimension: 5
Show default: [✓ Time based] Length based
Zone offset: -10
Unit:
Zone unit:

	Signal expression	Zone width
1	1	2
2	2	4
3	3	10
4	2	4
5	1	2

Then the 2D top view of that logical might look like this:



The first signal has as position the offset specified in the logical dialog, the positions of the other signals are determined so that the distance between two sub signals is the average of the corresponding zone widths of those sub signals. As before, ibaAnalyzer draws linearly interpolated color values between the signals.
Note that for the outer halves of the outer zones, there is no signal to interpolate to; hence those half zones are left blank. In the Y-axis settings (and corresponding preferences), one can chose whether or not the blank half-zones are visible when auto scaling (by default this is disabled).



Scaling mode [✓ Autoscale from data set] Use scale definition from data file Manual scale

[✓] Show full edge zones in 2D top view

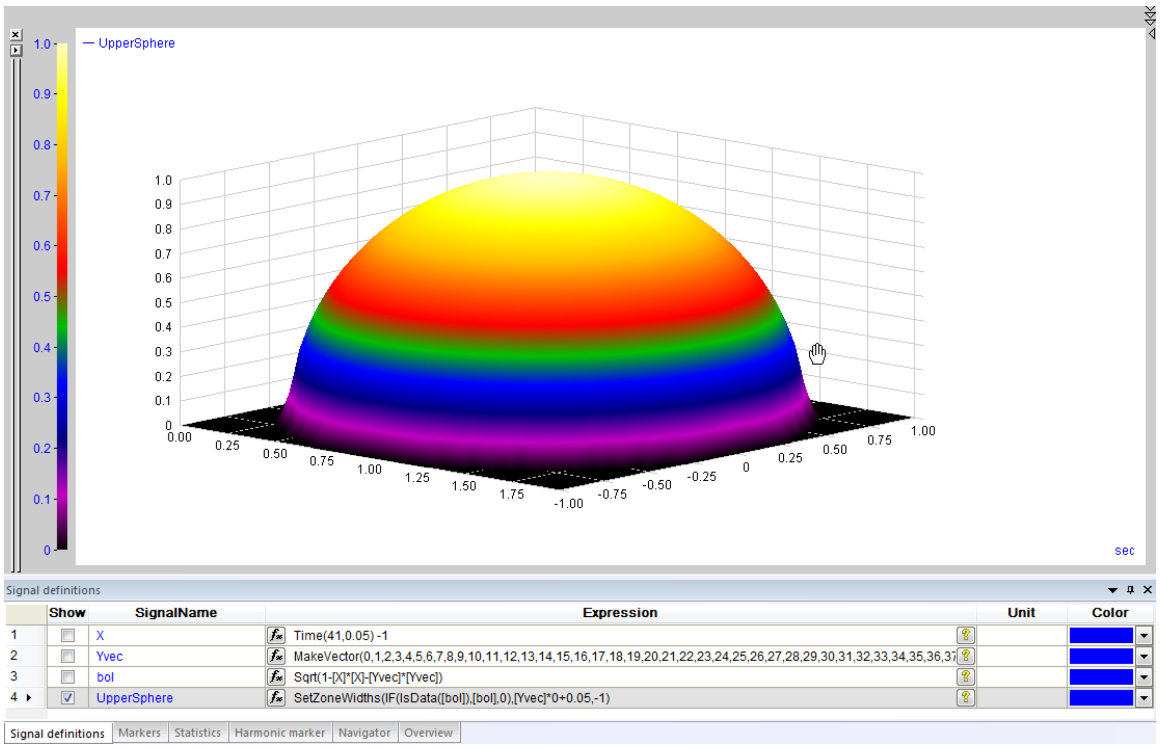
Min:

Max:

Add scale offset:

1.3 3D views

In 3D views of logicals/vectors, the specified zone widths are respected as well. Additionally a third axis is drawn which corresponds with the Y-axis of 2D top view plots.



Signal definitions

	Show	SignalName	Expression	Unit	Color
1	☐	X	Time(41,0.05) -1		🟦
2	☐	Yvec	MakeVector(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37		🟦
3	☐	bol	Sqrt(1-[X]*[X]-[Yvec]*[Yvec])		🟦
4	✓	UpperSphere	SetZoneWidths(IF(IsData([bol]),[bol],0),[Yvec]*0+0.05,-1)		🟦

1.4 Expressions

Most ibaAnalyzer functions can be applied on vectors where they are applied element by element. For functions taking multiple arguments, it is assumed the dimensions of the arguments are the same. If this is not the case, the result will have as dimension the largest dimension of the arguments and the arguments with smaller dimension will be rescaled. Note that this is currently simply a rescaling of indices and that zone widths will not be respected.
When doing calculations on vectors with zone widths, the zone widths of the result will be equal to the zone widths of the first argument that has a full dimension (i.e. a dimension equal to the maximum of all argument dimensions).

1.4.1 GetRows function

The *GetRows* function will return a subset of a vector. The sub signals of the resulting vector will have the same zone widths as the corresponding sub signals of the original vector. The offset to the first zone will be such that the position of the first sub signal in the returned vector matches the position of the corresponding sub signal in the original vector. The positions of the other returned sub signals will match as well unless a ‘step’ parameter different from 1 is used. In the latter case the position of the sub signal in the returned vector will be equal to the position of the corresponding sub signal minus the zone widths of the skipped sub signals.

1.4.2 MakeVector function

If the *MakeVector* function has arguments that are themselves vectors, the resulting vector will have a dimension that is the sum of the dimensions of the arguments. The sub signals in the resulting vector will consist out of the sub signals of the first argument, followed by the sub signals of the second argument and so on. The zone widths of the sub signals in the resulting vector will correspond with the zone widths of the original sub signals. The offset to the first zone center of the first argument will be used as the offset to the first zone center of the resulting vector.

1.5 Additional ibaAnalyzer functions

Some additional functions have been implemented in ibaAnalyzer to manipulate zone widths of vectors.

1.5.1 SetZoneWidths

This function creates a vector with zone widths where the values are taken from one vector and the widths from another vector. Since the vector with the widths can consist out of expressions, this allows for creating vectors with zone widths dependent on the loaded data. If the width vector consists out of expressions, the expressions are expected to evaluate to a constant (i.e. they should not vary over time for the same loaded data). If this is not the case, their average over the entire time range is used.
The *SetZoneWidths* function takes the following arguments:

- vector, the vector with the values.
- widths, the vector with the widths.
- offset, an optional offset to the first zone center, if omitted the offset of 'vector' is used.

E.g.: The following expression:

```
SetZoneWidths(MakeVector(1,2,3,2,1),MakeVector(2,4,10,4,2),-10)
```

creates the same vector as the one created by the logical in paragraph **1.2**

1.5.2 GetZoneWidths

This function returns a vector containing the widths of a vector. The returned vector itself has standard widths and offset (1 and 0). The *GetZoneWidths* function takes as only argument the vector to take the widths from.

1.5.3 GetZoneOffset

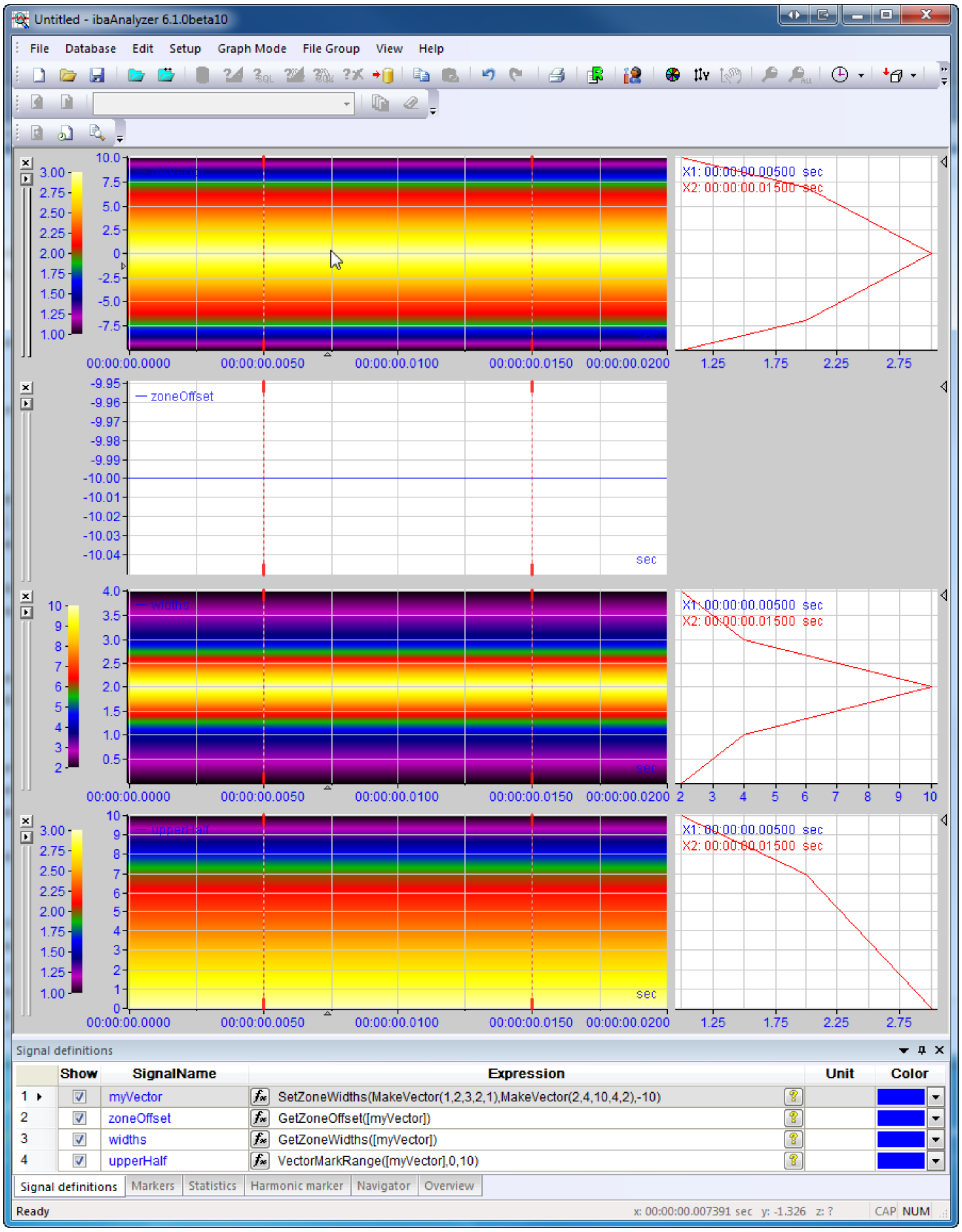
This function returns the offset to the first zone of a vector. The *GetZoneOffset* function takes as only argument the vector to take the offset to the first zone from.

1.5.4 VectorMarkRange

This function returns similar as the *GetRows* function a subset of a vector. However rather than specifying row indices, two *positions* must be specified. All sub signals of the vector which are positioned between the two specified positions are retained. The sub signals in the resulting vector have the same zone widths as the corresponding sub signals in the original vector and the offset to the first zone of the resulting vector is such that all sub signals in the resulting vector have the same position as the corresponding sub signals in the original vector.
The *VectorMarkRange* function takes the following arguments:

- Vector, the vector to take the subset from
- PositionFrom, the smallest of the two edge positions.
- PositionTo, the largest of the two edge positions.

PostionFrom and PositionTo can be expressions themselves and hence be dependent on the loaded data, however they are expected to evaluate to a constant value (i.e. they should not vary over time for the same loaded data). If this is not the case, their average over the entire time range is used.



Signal definitions

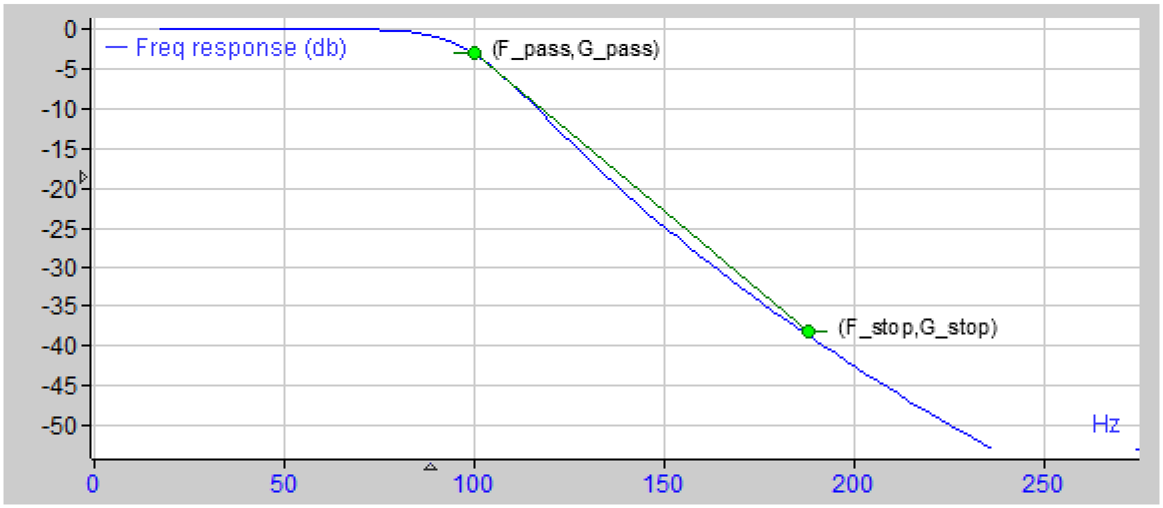
	Show	SignalName	Expression	Unit	Color
1	✓	myVector	SetZoneWidths(MakeVector(1,2,3,2,1),MakeVector(2,4,10,4,2),-10)		🟦
2	✓	zoneOffset	GetZoneOffset([myVector])		🟦
3	✓	widths	GetZoneWidths([myVector])		🟦
4	✓	upperHalf	VectorMarkRange([myVector],0,10)		🟦

Ready x: 00:00:00.007391 sec y: -1.326 z: ?

2 Adapted filter functions

In ibaAnalyzer one can create digital filters in the filter design dialog (button on the toolbar). In this dialog one can, depending on the selected type of filter, specify the gains and frequencies of the start and/or stop bands. One does so by dragging green dots in the graphical representation of the filter, or by right-clicking on the graphical representation, selecting 'manual filter setup' from the context menu and typing the desired values in the next dialog. After having designed a filter in the dialog, one has a new function available in ibaAnalyzer with the same name as specified in the dialog that takes as only argument an input signal and the result of that function is the filter applied to the input signal.
Unfortunately, this approach does not allow for frequency and gain parameters that are dependent on the loaded data. In the current version of ibaAnalyzer, each filter function takes additional optional parameters that can be used to override the frequencies and gains specified in the filter design dialog. These arguments can be expressions themselves (and hence be dependent on the loaded data) but are expected to evaluate to a constant (i.e. they should not vary over time for the same loaded data). If this is not the case, their average over the entire time range is used.

2.1 Low-pass filter

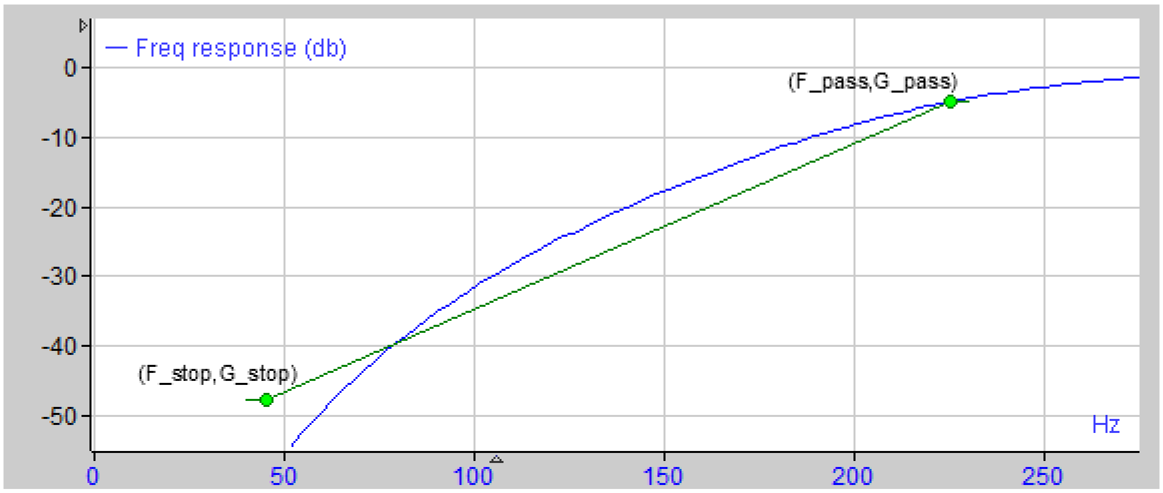


A low-pass digital filter takes as first mandatory parameter the input signal to be filtered and then takes as optional arguments:

- F_pass: The passband edge frequency, i.e. the rightmost point of the passband.
- F_stop: The stopband edge frequency, i.e. the leftmost point of the stopband. It must be larger than F_pass.
- G_pass: gain of the passband in dB, must be between -0.5 and -5 dB.
- G_stop: gain of the stopband in dB, must be smaller than G_pass minus 0.1 dB.

G_pass and G_stop can be omitted, in which case the gains specified in the filter dialog are used, One can also omit F_stop and only specify F_pass; in that case F_stop will be calculated so that the difference between F_pass and F_stop is the same as specified in the filter design dialog.

2.2 High-pass filter

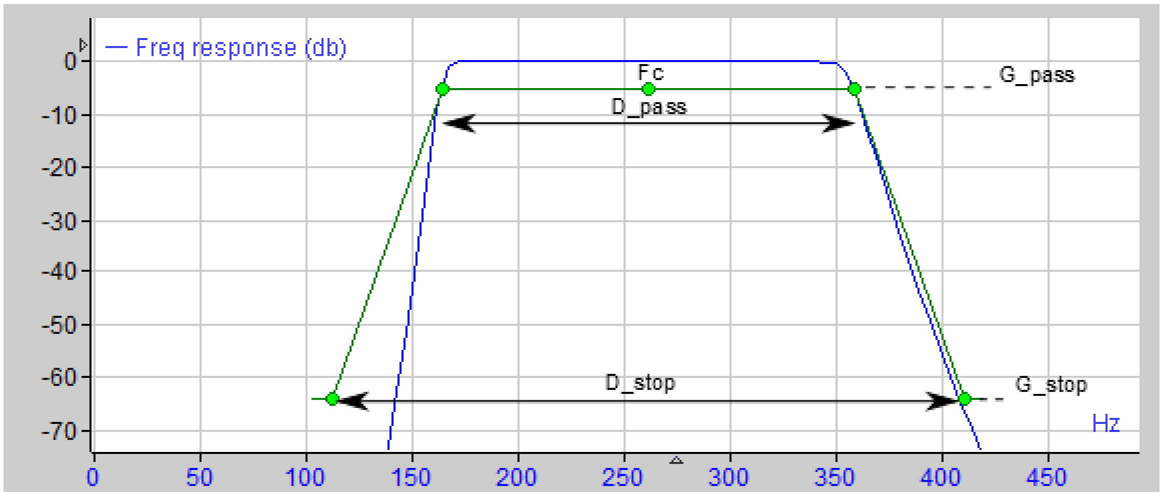


A high-pass digital filter takes as first mandatory parameter the input signal to be filtered and then takes as optional arguments:

- F_stop: The stopband edge frequency, i.e. rightmost point of the stopband
- F_pass: The passband edge frequency, i.e. leftmost point of the passband. It must be larger than F_stop.
- G_stop: gain of the stopband in dB, must be smaller than G_pass minus 0.1 dB.
- G_pass: gain of the passband in dB, must be between -0.5 and -5 dB.

G_stop and G_pass can be omitted, in which case the gains specified in the filter dialog are used. One can also omit F_pass and only specify F_stop; in that case F_pass will be calculated so that the difference between F_stop and F_pass is the same as specified in the filter design dialog.

2.3 Band-pass filter



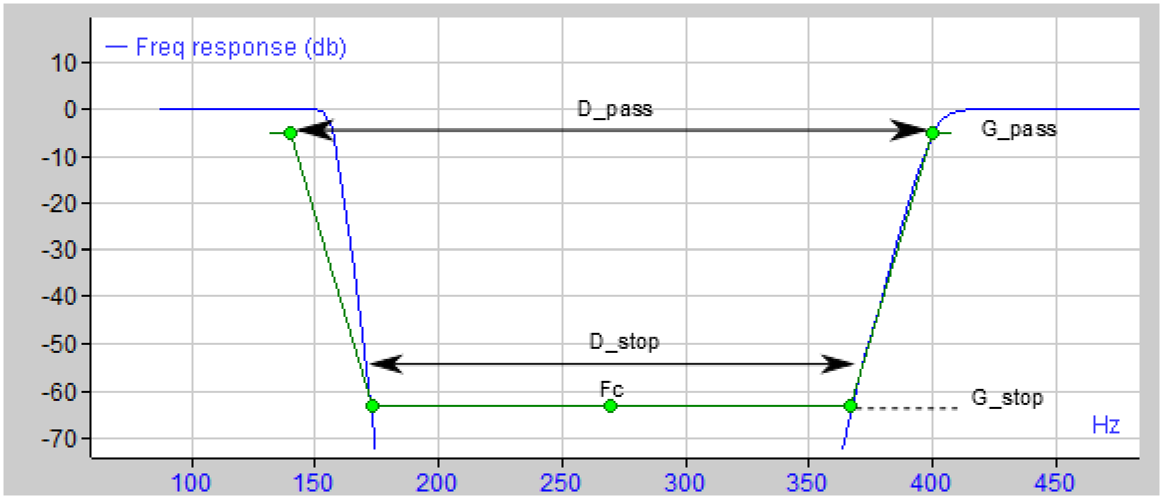
A band-pass digital filter takes as first mandatory parameter the input signal to be filtered and then takes as optional arguments:

- Fc: The frequency at the center of the passband.
- D_pass: Width of the passband (i.e. filter bandwidth); the passband outer frequencies will be symmetric around Fc with a distance between them determined by this argument.

- D_stop: Distance between first and second stopband, the stopband frequencies will also be symmetric around Fc. D_stop needs to be larger than D_pass.
- G_pass: Passband gain in dB, must be between -0.5 and -5 dB.
- G_stop: Stopband gain in dB, must be smaller than G_pass minus 0.1 dB

D_pass, D_stop, G_pass and G_stop can be omitted in which case the parameters specified in the filter design dialog are used. One can also specify only Fc and D_pass; in that case D_stop will be calculated so that the difference between D_stop and D_pass is the same as specified in the filter design dialog.

2.4 Band-stop filter



A band-stop digital filter takes as first mandatory parameter the input signal to be filtered and then takes as optional arguments:

- Fc: The frequency at the center of the stopband.
- D_stop: Width of the stopband; the stopband outer frequencies will be symmetric around Fc width a distance between them determined by this argument.
- D_pass: Distance between first and second passband, the passband frequencies will also be symmetric around Fc. D_pass needs to be larger than D_stop.
- G_stop: Stopband gain in dB, must be smaller than G_pass minus 0.1 dB.
- G_pass: Passband gain in dB, must be between -0.5 and -5 dB.

D_stop, D_pass, G_stop and G_pass can be omitted in which case the parameters specified in the filter design dialog are used. One can also specify only Fc and D_stop; in that case D_pass will be calculated so that the difference between D_pass and D_stop is the same as specified in the filter design dialog.

3 Miscellaneous new features

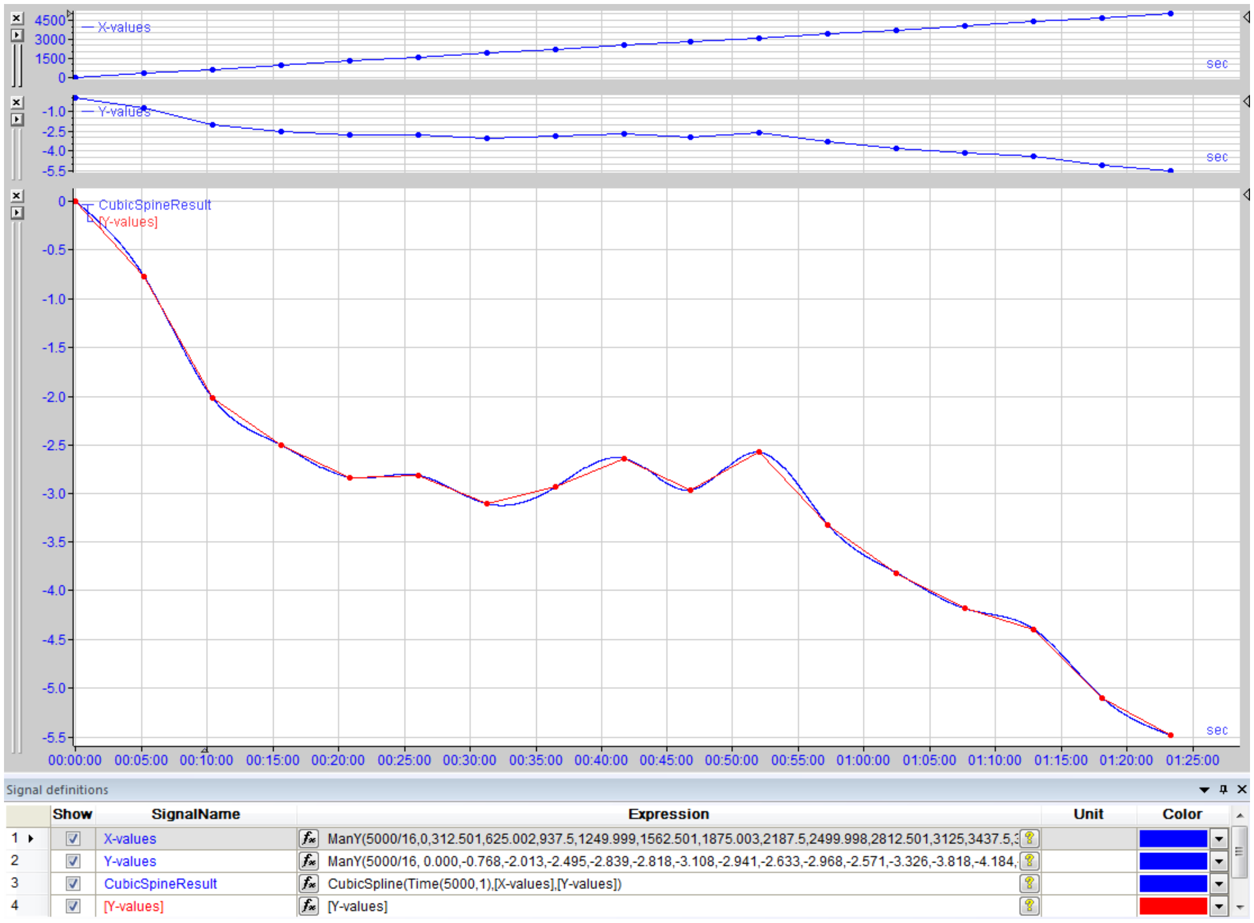
3.1 New ibaAnalyzer functions

3.1.1 CubicSpline function

The CubicSpline function creates a cubic spline for a set of points and then proceeds to evaluate an input signal along that spline.
This function can be used to smoothly interpolate a sparsely sampled channel.
The function takes three arguments:

- Expression: the values that need to be evaluated along the cubic spline.
- X, the X- coordinates of the set of points that define the cubic spline.
- Y, the Y- coordinates of the set of points that define the cubic spline.

The X coordinates do not need to be unique and sorted; however, before constructing the spline, if multiple (X, Y) pairs have the same X value only the last pair is retained. Also the remaining set of pairs is sorted on X value before constructing the spline.



	Show	SignalName	Expression	Unit	Color
1	✓	X-values	ManY(5000/16,0,312.501,625.002,937.5,1249.999,1562.501,1875.003,2187.5,2499.998,2812.501,3125,3437.5,3		🟦
2	✓	Y-values	ManY(5000/16, 0.000,-0.768,-2.013,-2.495,-2.839,-2.818,-3.108,-2.941,-2.633,-2.968,-2.571,-3.326,-3.818,-4.184,		🟦
3	✓	CubicSpineResult	CubicSpline(Time(5000,1),[X-values],[Y-values])		🟦
4	✓	[Y-values]	[Y-values]		🟥

3.1.2 HighPrecision function

Since version 5.21.0, ibaAnalyzer can perform calculations in both single (32-bit) and double (64-bit) precision. Double precision has the advantage of being more accurate while having the disadvantage of requiring twice as much memory to store the results and temporary results.
ibaAnalyzer decides itself in which mode it will operate depending on the precision of the input arguments it receives for its functions. If the signal was written to the .dat file using 16 bit integers or 32 bit floating point numbers, ibaAnalyzer will use single precision, while it will use double precision if the signals were written to the .dat file using 32 bit integers or 64 bit floating point numbers.
With the HighPrecision function, one can force ibaAnalyzer to perform double precision calculations regardless of the precision of the input arguments.
The HighPrecision function takes only one argument and returns an identical signal as its argument, with the exception that when used in further calculations, it will trigger ibaAnalyzer to do those calculations in higher precision.
It should be noted that ibaAnalyzer is optimized so it does not take an actual double precision copy of the argument of HighPrecision, rather the argument is merely earmarked to be treated as a double precision signal. Ofcourse the results and temporary results of any further calculations performed on the result of the HighPrecision function will be stored in double precision and hence take twice as much memory.

3.2 IbaCapture-CAM

3.2.1 Zooming by wheel button

By scrolling the wheel button of the mouse, one can zoom in and out.
One zooms in by wheeling forward and zooms out by wheeling backwards.
The zoom factor is the square root of two for each notch of the mouse wheel, hence two notches zooms in or out by a factor of two. The zoom is centered around the position of the mouse cursor on the video (or as close as possible if the mouse cursor is near the edge of the video frame).

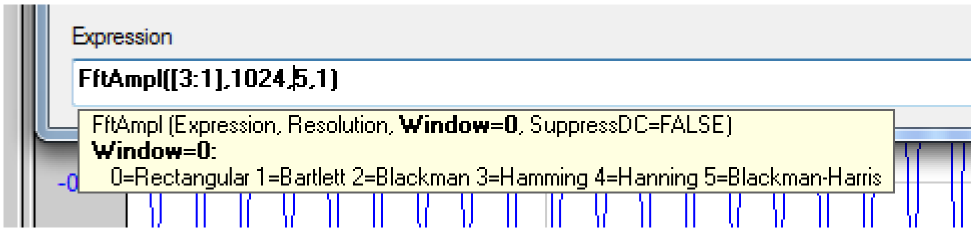
3.2.2 Panning

When zoomed in and holding the ALT key, the mouse cursor will change into a little hand. Now you can pan the video image by dragging.

3.3 FFT

3.3.1 Blackman-Harris window

The Blackman-Harris window function is added to the range of possible window functions to be applied before calculating FFTs in ibaAnalyzer.
In ibaAnalyzer FFT functions that have the 'Window' (e.g. FftAmpl) parameter, setting this parameter to the value 5 selects the Blackman-Harris window.

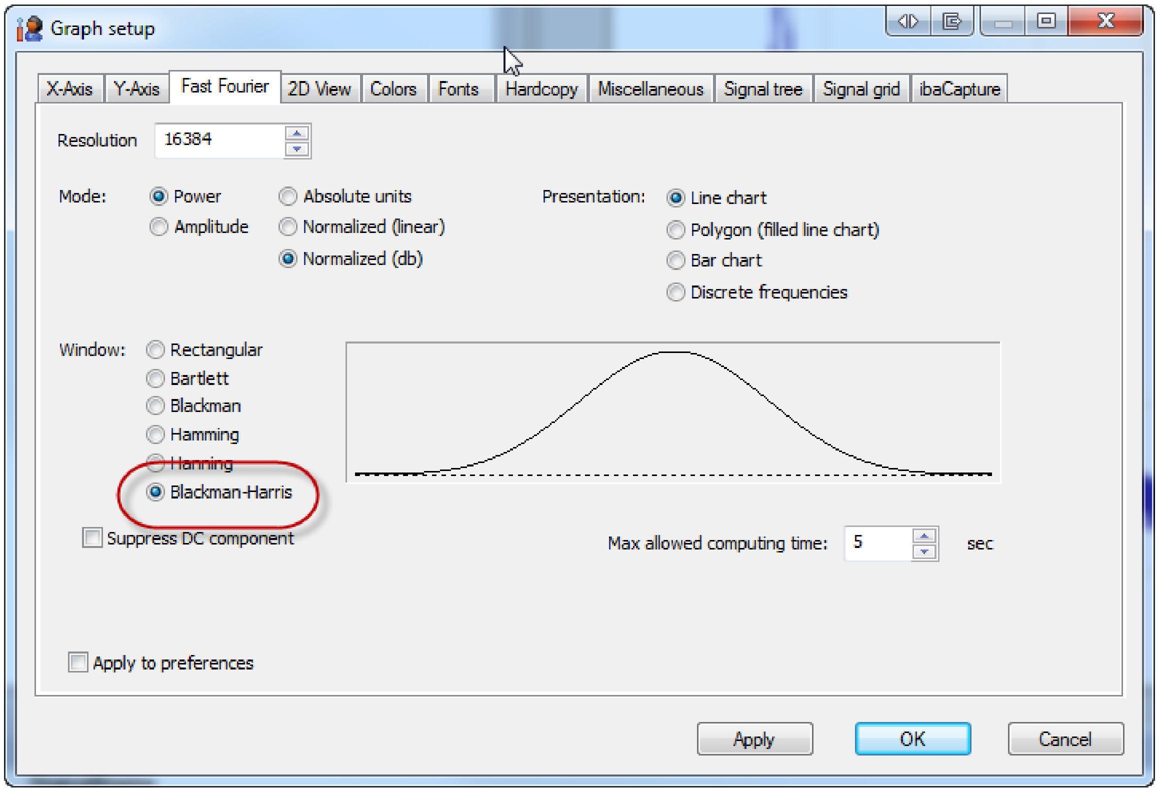


Expression

```
FftAmpl([3:1],1024,5,1)
```

FftAmpl (Expression, Resolution, **Window=0**, SuppressDC=FALSE)
Window=0:
0=Rectangular 1=Bartlett 2=Blackman 3=Hamming 4=Hanning 5=Blackman-Harris

For the FFT views in ibaAnalyzer (selecting by clicking the little button on the top left of graphs), the parameter can be set in the usual FFT settings. An additional radio button is added to select the Blackman-Harris window from the list of supported window functions.



Resolution: 16384

Mode: [✓ Power] Amplitude

Absolute units / Normalized (linear) / [✓ Normalized (db)]

Presentation: [✓ Line chart] Polygon (filled line chart) / Bar chart / Discrete frequencies

Window: Rectangular / Bartlett / Blackman / Hamming / Hanning / [✓ Blackman-Harris]

☐ Suppress DC component

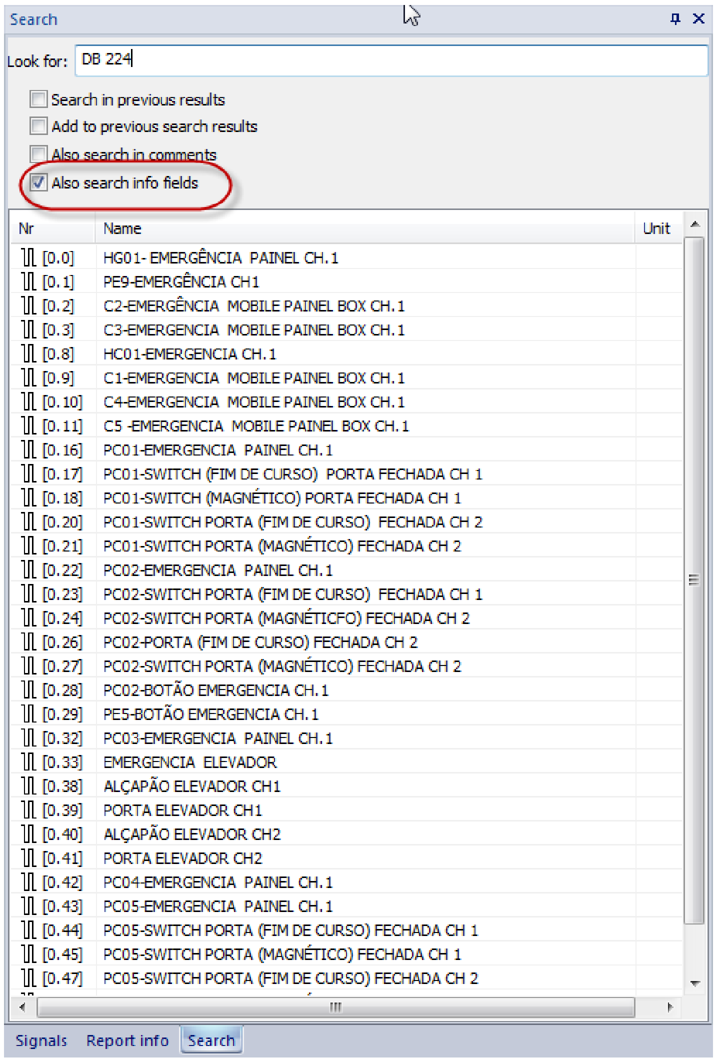
Max allowed computing time: 5 sec

☐ Apply to preferences

3.4 Search dialog

3.4.1 Search in info fields

The option to search into other channel info fields than the comment fields for the search string has been added to the signal search dialog.



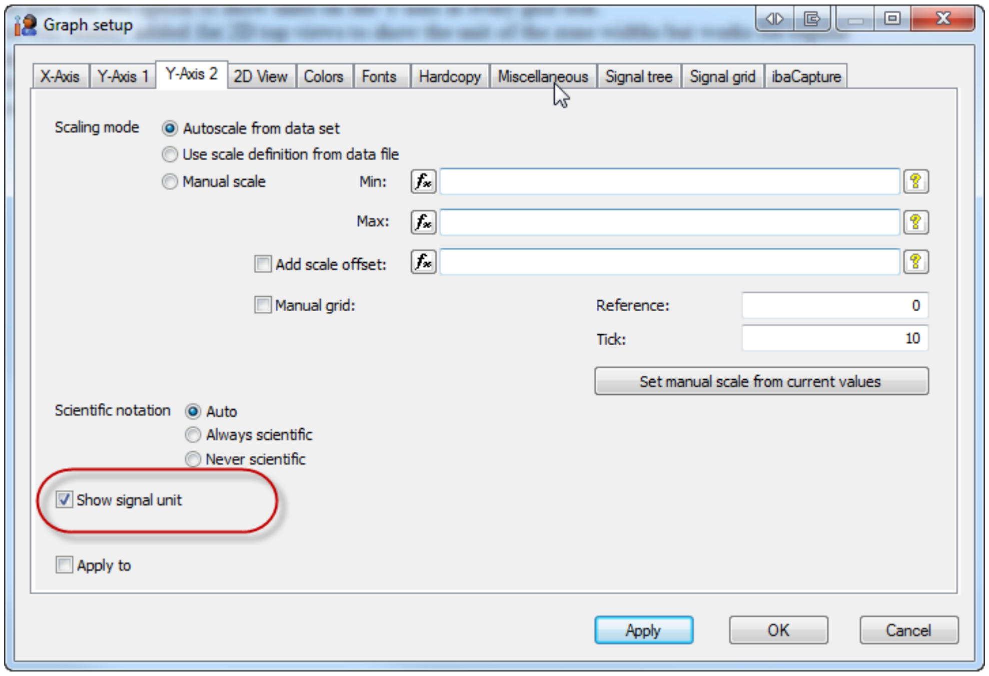
Look for: DB 224

[ ] Search in previous results
[ ] Add to previous search results
[ ] Also search in comments
[✓] Also search info fields

Nr	Name	Unit
[0.0]	HG01- EMERGÊNCIA PAINEL CH.1	
[0.1]	PE9-EMERGÊNCIA CH1	
[0.2]	C2-EMERGÊNCIA MOBILE PAINEL BOX CH.1	
[0.3]	C3-EMERGENCIA MOBILE PAINEL BOX CH.1	
[0.8]	HC01-EMERGENCIA CH.1	
[0.9]	C1-EMERGENCIA MOBILE PAINEL BOX CH.1	
[0.10]	C4-EMERGENCIA MOBILE PAINEL BOX CH.1	
[0.11]	C5 -EMERGENCIA MOBILE PAINEL BOX CH.1	
[0.16]	PC01-EMERGENCIA PAINEL CH.1	
[0.17]	PC01-SWITCH (FIM DE CURSO) PORTA FECHADA CH 1	
[0.18]	PC01-SWITCH (MAGNÉTICO) PORTA FECHADA CH 1	
[0.20]	PC01-SWITCH PORTA (FIM DE CURSO) FECHADA CH 2	
[0.21]	PC01-SWITCH PORTA (MAGNÉTICO) FECHADA CH 2	
[0.22]	PC02-EMERGENCIA PAINEL CH.1	
[0.23]	PC02-SWITCH PORTA (FIM DE CURSO) FECHADA CH 1	
[0.24]	PC02-SWITCH PORTA (MAGNÉTICFO) FECHADA CH 2	
[0.26]	PC02-PORTA (FIM DE CURSO) FECHADA CH 2	
[0.27]	PC02-SWITCH PORTA (MAGNÉTICO) FECHADA CH 2	
[0.28]	PC02-BOTÃO EMERGENCIA CH.1	
[0.29]	PE5-BOTÃO EMERGENCIA CH.1	
[0.32]	PC03-EMERGENCIA PAINEL CH.1	
[0.33]	EMERGENCIA ELEVADOR	
[0.38]	ALÇAPÃO ELEVADOR CH1	
[0.39]	PORTA ELEVADOR CH1	
[0.40]	ALÇAPÃO ELEVADOR CH2	
[0.41]	PORTA ELEVADOR CH2	
[0.42]	PC04-EMERGENCIA PAINEL CH.1	
[0.43]	PC05-EMERGENCIA PAINEL CH.1	
[0.44]	PC05-SWITCH PORTA (FIM DE CURSO) FECHADA CH 1	
[0.45]	PC05-SWITCH PORTA (MAGNÉTICO) FECHADA CH 1	
[0.47]	PC05-SWITCH PORTA (FIM DE CURSO) FECHADA CH 2	

3.5 Units on Y-axis

One now has the option to show units on the Y-axis at every grid tick.
This was mainly added for 2D top views to show the unit of the zone widths but works for regular graphs as well were the units of the signals are shown instead. If multiple signals with different units are placed on the same Y-axis, the unit of the first signal is used. One sets the option to show the units in the Y-axis preferences, by default it is disabled.


Graph setup

X-Axis | Y-Axis 1 | Y-Axis 2 | 2D View | Colors | Fonts | Hardcopy | Miscellaneous | Signal tree | Signal grid | ibaCapture

Scaling mode: [✓ Autoscale from data set] / Use scale definition from data file / Manual scale

Min:

Max:

☐ Add scale offset:

☐ Manual grid:

Reference: 0

Tick: 10

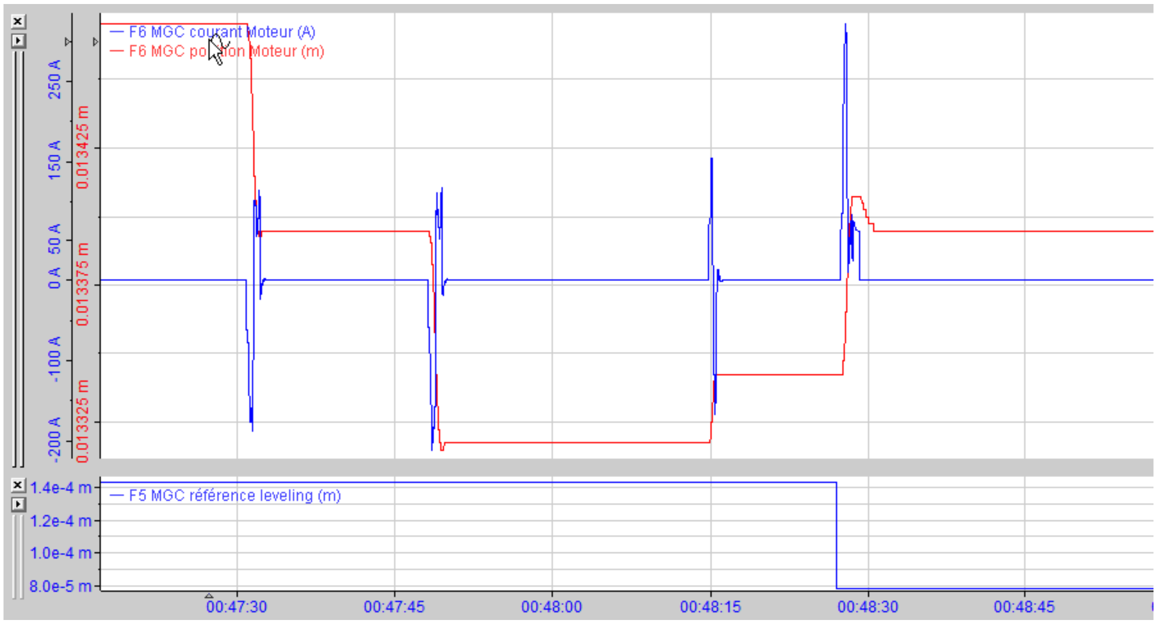
Set manual scale from current values

Scientific notation: [✓ Auto] / Always scientific / Never scientific

☑ Show signal unit

☐ Apply to

Apply | OK | Cancel


F6 MGC courant Moteur (A)
F6 MGC position Moteur (m)
250 A
150 A
50 A
0 A
-100 A
-200 A
0.013425 m
0.013375 m
0.013325 m
F5 MGC référence leveling (m)
1.4e-4 m
1.2e-4 m
1.0e-4 m
8.0e-5 m
00:47:30
00:47:45
00:48:00
00:48:15
00:48:30
00:48:45