ibaAnalyzer

Ausdruckseditor

Handbuch Teil 3

Ausgabe 8.3

Messsysteme für Industrie und Energie

www.iba-ag.com

Hersteller

iba AG

Königswarterstraße 44

90762 Fürth

Deutschland

Kontakte

Zentrale	+49 911 97282-0
Support	+49 911 97282-14
Technik	+49 911 97282-13
E-Mail	iba@iba-ag.com
Web	www.iba-ag.com

Version	Datum	Revision	Autor	Version SW
8.3	06-2025	Funktionen aktualisiert gem. v8.3.0	mm	8.3.0

Inhalt

1 Zu dieser Dokumentation

Diese Dokumentation beschreibt die Funktion und die Anwendung der Software *ibaAnalyzer*.

1.1 Zielgruppe

Diese Dokumentation wendet sich an ausgebildete Fachkräfte, die mit dem Umgang mit elektrischen und elektronischen Baugruppen sowie der Kommunikations- und Messtechnik vertraut sind. Als Fachkraft gilt, wer auf Grund der fachlichen Ausbildung, Kenntnisse und Erfahrungen sowie Kenntnis der einschlägigen Bestimmungen die übertragenen Arbeiten beurteilen und mögliche Gefahren erkennen kann.

Diese Dokumentation wendet sich insbesondere an Personen, die mit der Auswertung von Mess- und Prozessdaten befasst sind. Da die Bereitstellung der Daten mit anderen iba-Produkten erfolgt, sind für die Arbeit mit *ibaAnalyzer* folgende Vorkenntnisse erforderlich bzw. hilfreich:

- Betriebssystem Windows
- *ibaPDA* (Entstehung und Struktur der Messdateien)

1.2 Schreibweisen

In dieser Dokumentation werden folgende Schreibweisen verwendet:

Aktion	Schreibweise
Menübefehle	Menü *Funktionsplan*
Aufruf von Menübefehlen	*Schritt 1 – Schritt 2 – Schritt 3 – Schritt x* Beispiel: Wählen Sie Menü *Funktionsplan – Hinzufügen – Neuer Funktionsblock*
Tastaturtasten	<Tastenname> Beispiel: <Alt>; <F1>
Tastaturtasten gleichzeitig drücken	<Tastenname> + <Tastenname> Beispiel: <Alt> + <Strg>
Grafische Tasten (Buttons)	<Tastenname> Beispiel: <OK>; <Abbrechen>
Dateinamen, Pfade	`Dateiname`, `Pfad` Beispiel: `Test.docx`

1.3 Verwendete Symbole

Wenn in dieser Dokumentation Sicherheitshinweise oder andere Hinweise verwendet werden, dann bedeuten diese:

Gefahr!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die unmittelbare Gefahr des Todes oder der schweren Körperverletzung!

- Beachten Sie die angegebenen Maßnahmen.

Warnung!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die mögliche Gefahr des Todes oder schwerer Körperverletzung!

- Beachten Sie die angegebenen Maßnahmen.

Vorsicht!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die mögliche Gefahr der Körperverletzung oder des Sachschadens!

- Beachten Sie die angegebenen Maßnahmen.

Hinweis

Hinweis, wenn es etwas Besonderes zu beachten gibt, wie z. B. Ausnahmen von der Regel usw.

Tipp

Tipp oder Beispiel als hilfreicher Hinweis oder Griff in die Trickkiste, um sich die Arbeit ein wenig zu erleichtern.

Andere Dokumentation

Verweis auf ergänzende Dokumentation oder weiterführende Literatur.

1.4 Aufbau der Dokumentation

In dieser Dokumentation wird umfassend die Funktionalität der Software *ibaAnalyzer* beschrieben. Sie ist als Leitfaden zur Einarbeitung wie auch als Nachschlagedokument angelegt.

Ergänzend zu dieser Dokumentation können Sie für aktuelle Informationen zur installierten Programmversion die Versionshistorie im Hauptmenü *Hilfe – Versionshistorie* heranziehen (Datei `versions.htm`). In dieser Datei wird neben der Aufzählung behobener Programmfehler auch auf Erweiterungen und Verbesserungen der Software stichwortartig hingewiesen.

Außerdem wird mit jedem Software-Update, das nennenswerte neue Features enthält, eine spezielle Dokumentation "NewFeatures…" ausgeliefert, die eine ausführliche Beschreibung der neuen Funktionen bietet.

Der Stand der Software, auf den sich der jeweilige Teil dieser Dokumentation bezieht, ist in der Revisionstabelle auf Seite 2 aufgeführt.

Die Dokumentation von *ibaAnalyzer* (PDF-Ausgabe) ist in sechs separate Teile gegliedert. Jeder Teil hat seine eigene bei 1 beginnende Kapitel- und Seitennummerierung und wird unabhängig aktualisiert.

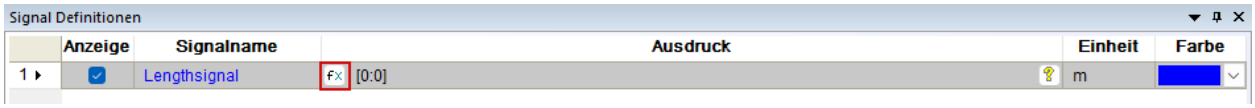
Teil	Titel	Inhalt
Teil 1	Einführung und Installation	Allgemeine Hinweise, Lizenzen und Add-ons Installation und Programmstart Bedienoberfläche
Teil 2	Arbeiten mit *ibaAnalyzer*	Arbeiten mit Messdatei und Analyse, Darstellungsfunktionen, Makrokonfiguration, Filterdesign, Voreinstellungen, Drucken, Export, Schnittstellen zu *ibaHD-Server*, *ibaCapture* und Reportgenerator
Teil 3	Ausdruckseditor	Verzeichnis aller Berechnungsfunktionen im Ausdruckseditor, inkl. Erklärung
Teil 4	Datenbank-Schnittstelle	Arbeiten mit Daten aus Datenbanken, Verbindung zur Datenbank, Schreiben von iba-Messdaten in Datenbanken, Extraktion der Daten aus der Datenbank und Analyse der Daten
Teil 5	Schnittstelle für Datei-Extraktion	Funktionen und Einstellungen zur Extraktion von Daten aus iba-Messdateien in externe Dateiformate
Teil 6	Anwendungsbeispiele	*In Vorbereitung*

2 Funktionsweise des Ausdruckseditors

In *ibaAnalyzer* können Sie Formeln für die Analyse der Messwerte nutzen. Diese Formeln können Sie von Hand eingeben, z. B. in der Signaltabelle im Register *Signaldefinitionen*, oder Sie nutzen den Ausdruckseditor.

Der Ausdruckseditor ist ein Hilfsmittel zur Eingabe von (mathematischen) Formeln oder Ausdrücken. Er erleichtert die Eingaben und bietet auch eine ausführliche Liste der möglichen Operationen und ihrer Syntax.

Ausdruckseditor öffnen
Den Ausdruckseditor können Sie über den <fx>-Button in jeder Tabellenzeile aus öffnen, in die Sie ein Signal eingeben können.

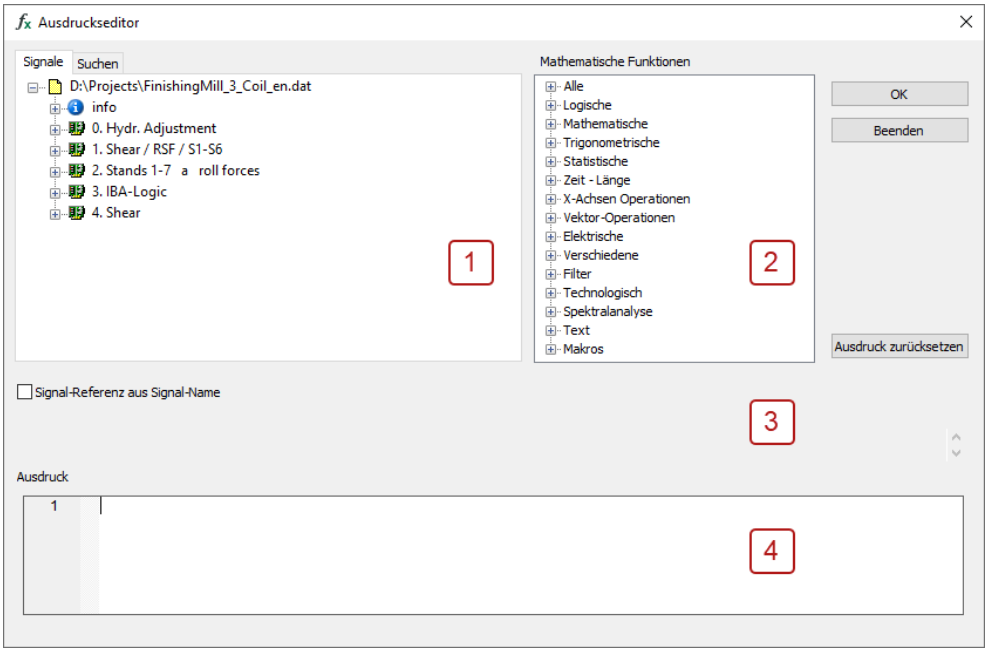


Hinweis

Das Icon f_x in der Symbolleiste öffnet nicht den Ausdruckseditor, sondern den Dialog für die logischen Ausdrücke, siehe *Logische Ausdrücke* in Teil 2 des Handbuchs.

2.1 Oberfläche des Ausdruckseditors

Der Ausdruckseditor besteht aus 4 Bereichen.



1	Signalbaum	3	Hinweise zur Funktion
2	Funktionsbaum	4	Eingabefeld

Im Signalbaum links (1) finden Sie zusätzlich zu den Originalsignalen aus der Messdatei auch alle erstellten Ausdrücke aus dem Ausdruckseditor. Wählen Sie hier die Signale oder Ausdrücke aus, mit denen Sie rechnen möchten.

Der Funktionsbaum rechts (2) zeigt alle verfügbaren mathematischen Operationen und andere Funktionen nach Themen geordnet an.

Wenn Sie eine Funktion im Funktionsbaum auswählen, werden darunter Hinweise zur Funktion (3) angezeigt, z. B. die Syntax.

In das Eingabefeld unten (4) tragen Sie den gewünschten Ausdruck ein. Ein Tooltip zur Funktion erscheint, wenn die Funktion im Eingabefeld markiert ist.

Der Button <Ausdruck zurücksetzen> löscht alle Eintragungen aus der Befehlszeile.

Wenn Sie die Option *Signalreferenz aus Signalnamen* aktivieren, werden die Signalnamen statt der üblichen Signalkennung [Modulnummer:Signalnummer] im Ausdruck verwendet.

Hinweis

Stellen Sie bei der Verwendung der Signalnamen als Signalreferenz sicher, dass die Signalnamen eindeutig sind.

2.2 Signale und Platzhalter in Ausdrücken

ibaAnalyzer kennzeichnet Signale in Ausdrücken mit eckigen Klammern "[ ]". Jedes verfügbare Signal hat eine Kennung, die aus einer Modulnummer und einer Signalnummer besteht. Je nach Signaltyp werden Modulnummer und Signalnummer mit unterschiedlichen Trennzeichen getrennt.

- Analogsignale und Textsignale: Trennung durch einen Doppelpunkt ":"
- Digitalsignale: Trennung durch einen einfachen Punkt "."

In einigen Fällen ist auch eine Untersignalnummer vorhanden, die wiederum durch das gleiche Trennzeichen (je nach Signaltyp) getrennt ist.

Bei extrahierten Dateien von *ibaAnalyzer* können Sie die Ergebnisse der Berechnungen über ihren Namen referenzieren.

Beispiele

[0:0]	Analogsignal oder Textsignal aus Modul 0 und Signal 0
[1.2]	Digitalsignal aus Modul 1 und Signal 2
[3:2:1]	Untersignal eines Analogsignals aus Modul 3, Signal 2 und Untersignal 1
[Ausdruck]	Ausdruck mit dem Namen "Ausdruck"

Dateiindizes bei mehreren Dateien

Wenn mehrere Dateien parallel geöffnet werden, ist der Dateiindex ebenfalls Teil der Kennung. Die erste Datei hat den Index 0, den Sie auch weglassen können.

Beispiele

[0_0:0]	= [0:0], Analogsignal aus der ersten Datei, Modul 0, Signal 0
[2_1.0]	Digitalsignal aus der Datei mit dem Index 2 (dritte Datei), Modul 1, Signal 0

Platzhalter bei Signalangaben

Mit Hilfe von Platzhaltern können Sie mehrere Signale auf einmal adressieren. Das Ergebnis ist ein Vektor. Auf folgende Weise können Sie jeden Teil der Kennung ersetzen (Dateiindex, Modulnummer, Signalnummer, Untersignalnummer):

- Bereichsangabe mit Bindestrich, z. B. 3-6
- Stern "*" für alle verfügbaren Nummern

Beispiele

[*_0:0]	liefert das Signal [0:0] für alle parallel geöffneten Dateien
[0:3-5]	liefert die Signale [0:3], [0:4], und [0:5]
[0-2_3:4]	liefert das Signal [3:4] der ersten 3 parallel geöffneten Dateien
[*.0]	liefert das erste Digitalsignal (mit der Nummer 0) von allen verfügbaren Modulen
[*_*:*]	liefert alle Analogsignale und Textsignale aller geöffneten Dateien

2.3 Bedienung des Ausdruckseditors

Sie haben verschiedene Möglichkeiten Operationen, Signale und Ausdrücke in das Eingabefeld einzufügen, z. B. per Drag & Drop oder per Doppelklick. Auf diese Weise vermeiden Sie Tippfehler und können schneller Arbeiten.

Eingabe mit Doppelklick

Positionieren Sie den Cursor an der Stelle, wo Sie den Operanden oder die Operation einfügen wollen. Klicken Sie dann doppelt auf das gewünschte Element im Funktionsbaum oder Signalbaum.

Hinweis

Das Übernehmen von Signalen und Ausdrücken per Doppelklick funktioniert nur im Ausdruckseditor, nicht in der Signaltabelle im Register *Signaldefinitionen*.

Eingabehilfe Intellisense
Die *Intellisense*-Eingabehilfe erleichtert die manuelle Eingabe von Ausdrücken in der Signaltabelle und der Befehlszeile des Ausdruckseditors. Bei der Eingabe erscheinen automatisch Vorschläge für Funktionen, Parameter, Signale oder virtuelle Ausdrücke aus der Messdatei.

Wählen Sie Vorschläge mit einem Klick oder mit den Pfeiltasten aus. Übernehmen Sie diese mit <Enter>. Wenn Sie weiterschreiben, zeigt die Liste fortlaufend passende Vervollständigungen an, bis der Befehl vollständig ist.

Der Ausdruckseditor fügt automatisch alle notwendigen Klammern ein.

Navigation in komplexen Ausdrücken
Um die Übersicht über komplexe Ausdrücke nicht zu verlieren, nutzen Sie die Tastenkombination <Strg> + <B>, um zwischen zusammengehörigen Klammerpaaren hin- und herzuspringen.

Ausdruckseditor schließen und Ausdruck übernehmen
Wenn Sie den Ausdruckseditor mit <OK> schließen, erscheint der erstellte Ausdruck in der entsprechenden Tabellenzeile.

Der Ausdruck wird automatisch als Signalname übernommen. Sie können ihn jedoch manuell durch einen Klartext ersetzen. Bei komplexen, kaskadierten Ausdrücken empfiehlt es sich, kurze und eindeutige Namen zu wählen, um die Übersichtlichkeit zu bewahren.

Für Informationen zum Vorgehen bei fehlerhaften Ausdrücken siehe ↗ *Diagnose und Syntaxfehler-Erkennung*, Seite 14.

2.4 Diagnose und Syntaxfehler-Erkennung

ibaAnalyzer bietet verschiedene Möglichkeiten, die Richtigkeit der erstellten Ausdrücke zu prüfen, z. B. bei Verlassen des Ausdruckseditors oder über die Diagnosefunktion in der Signaltabelle.

Prüfung bei Verlassen des Ausdruckseditors
Wenn Sie einen fehlerhaften Ausdruck im Ausdruckseditor erstellt haben und den Ausdruck mit <OK> übernehmen möchten, erhalten Sie eine Meldung mit der Möglichkeit, den Ausdruck zu korrigieren. Wenn Sie auf <Ja> klicken, zeigt der Cursor direkt die Stelle an, an welcher der Fehler im Ausdruck vermutet wird.

Tipp

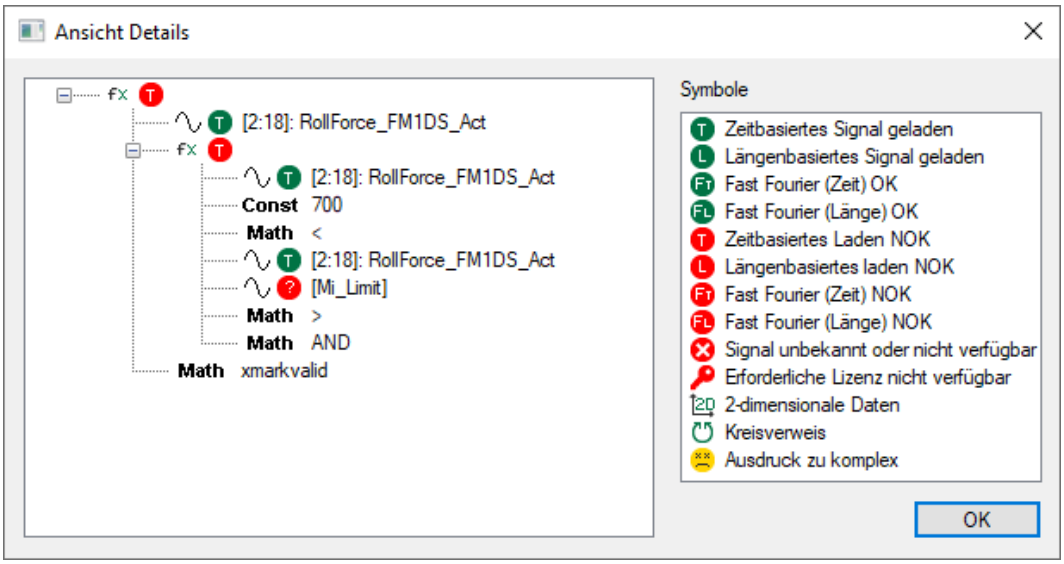
Ausdrücke im Ausdruckseditor können Sie auch manuell auf Fehler überprüfen mit der Tastenkombination <Strg> + <E>.

Diagnose in der Signaltabelle

Wenn Sie einen fehlerhaften Ausdruck manuell in die Signaltabelle eingegeben haben oder die Meldung beim Schließen des Ausdruckseditors ignoriert haben, wird der fehlerhafte Ausdruck in der Signaltabelle rot dargestellt. Auf diese Weise können Sie formale oder syntaktische Fehler erkennen, mit denen eine Berechnung nicht möglich ist.

Avg	Avg([3:16]
Max_Limit	700
Min_Limit	500
Valid	XMarkValid([2:18], [2:18] < [Max_Limit] AND [2:18] > [Mi_Limit])

Wenn Sie das Signal falsch referenziert haben, wird der Ausdruck normal dargestellt. Aber Sie können die Diagnose mit dem <?>-Button in der betreffenden Zeile öffnen, um genauere Informationen über die Fehlerursache zu erhalten.



Das fehlerhafte Signal ist mit einem roten Fragezeichen markiert.

2.5 Referenzen der logischen Ausdrücke

Im Register *Referenzen* der Signaltabelle können Sie die Verwendung und Referenzierung von Eingangssignalen und logischen Ausdrücken nachvollziehen. Das Register zeigt eine Liste aller Eingangssignale und Ausdrücke und deren Verwendung in Ausdrücken und Berechnungen. In der Tabelle *Ausdrücke referenzieren* können Sie die Ausdrücke direkt bearbeiten.

Weitere Informationen siehe *ibaAnalyzer*-Handbuch Teil 1, Kapitel *Register Referenz*.

3 Logische Funktionen

3.1 Vergleichsfunktionen

```
z. B. 'Expression1' < 'Expression2'
```

>	größer als
>=	größer als oder gleich
<	kleiner als
<=	kleiner als oder gleich
<>	ungleich
=	gleich

Beschreibung
Mit den Vergleichsoperationen >, >=, <, <=, <> und = können Sie die Werte zweier Ausdrücke (Operanden) miteinander vergleichen. Als Operanden können Sie Originalsignale, berechnete Ausdrücke oder einfach konstante Werte eintragen.

Die Operationen liefern als Ergebnis jeweils den booleschen Wert TRUE oder FALSE. Das Ergebnis können Sie als neuen Ausdruck wie ein Signal darstellen und auswerten. Auf diese Weise können Sie leicht binäre Signale bilden, die Sie wieder als Bedingungen für andere Funktionen verwenden können.

Hinweis

Wenn die Kreuzung zweier Kurven zwischen zwei Messpunkten liegt, dann wird das Ergebnis der Vergleichsoperation der letzten beiden Messwerte bis zum nächsten Messpunkt gehalten. Das heißt, dass ein Wechsel von TRUE nach FALSE (oder umgekehrt) stets im Raster der Messpunkte eingetragen wird.

Die Verbindungslinie zwischen zwei Messpunkten bei der Darstellung von Analogwerten ist nur eine grafische Näherung.

3.2 Boolesche Funktionen

```
z. B. 'Expression1' AND 'Expression2'
```

AND	Logisches UND
OR	Logisches ODER
XOR	Logisches Exklusiv-ODER
NOT	Logisches NOT, Negation

Beschreibung
Mit den booleschen Funktionen AND, OR, NOT und XOR können Sie binäre Ausdrücke miteinander verknüpfen, wie z. B. Digitalsignale. Als Parameter können Sie Digitalsignale, berechnete (binäre) Ausdrücke oder die Zahlenwerte 0 bzw. 1 eintragen.

Entsprechend den Regeln der booleschen Logik liefern die Funktionen als Ergebnis jeweils den Wert TRUE oder FALSE. Das Ergebnis können Sie als neuen Ausdruck wie ein Signal darstellen und auswerten. Auf diese Weise können Sie leicht binäre Signale bilden, die Sie wieder als Bedingungen für andere Funktionen verwenden können.

Logische Funktionen, Wahrheitstabelle:

A	B	A AND B	A OR B	A XOR B	NOT A
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	
1	1	1	1	0	

3.3 Boolesche Funktionen (bitweise)

```
z. B. bw_NOT ('Expression')
```

bw_AND	Bitweise UND
bw_OR	Bitweise ODER
bw_XOR	Bitweise Exklusiv-ODER
bw_NOT	Bitweise NOT

Beschreibung
Diese Funktionen dienen dem bitweisen Verknüpfen von zwei Analogwerten auf Basis der booleschen Algebra. Die Funktionen liefern als Ergebnissignal ein 32-Bit-Integer. Als Argumente werden 32-Bit-Integers erwartet.

Wenn die Argumente keine Integer sind, dann wird zunächst der Dezimalteil verworfen, bevor die Operation durchgeführt wird. Wenn die Argumente zu groß sind, dass ihr Absolutwert nicht in ein 32-Bit-Integer passt, wird die Operation nur auf den 32 niederwertigeren Bits ausgeführt.

Bei der Verknüpfung zweier Analogwerte mit einer bw-Funktion werden die einzelnen Bits beider Werte logisch verknüpft. Das Ergebnis ist wieder ein gleichartiger Analogwert mit einem Bitmuster entsprechend der logischen Verknüpfung.

Beispiel
Für 2 Analogwerte W1 = 15 und W2 = 2 ergeben sich folgende Ergebnisse:

	Dez.-Wert	Bits	Hex	Ergebniswert
Ausgangswert W1	15	…1111	0x0000000F	
Ausgangswert W2	2	…0010	0x00000002	
W1 bw_AND W2		…0010	0x00000002	2
W1 bw_OR W2		…1111	0x0000000F	15
W1 bw_XOR W2		…1101	0x0000000D	13
bw_NOT (W1)		…0000	0xFFFFFFF0	-16

3.4 Verzweigungen

3.4.1 If

```
If('Condition','IF-True','IF-False')
```

Argumente

'Condition'	Bedingung als Operation mit den booleschen Ergebnissen TRUE oder FALSE
'IF-True'	Operation wird ausgeführt, wenn 'Condition' gleich TRUE
'IF-False'	Operation wird ausgeführt, wenn 'Condition' gleich FALSE

Beschreibung
Die *If*-Funktion dient zur bedingten Ausführung weiterer Berechnungen. Abhängig vom booleschen Ergebnis einer Bedingung ('Condition'), die selbst eine Operation sein kann, wird beim Ergebnis TRUE die Operation 'IF-True' ausgeführt, beim Ergebnis FALSE entsprechend die Operation 'IF-False'.

Damit lassen sich unterschiedliche Berechnungen prozessgesteuert durchführen. Sie können die Funktion auch verschachtelt nutzen, um weitere Verzweigungen zu realisieren.

Hinweis

Wenn Sie Signale mit verschiedenen Abtastraten kombinieren, werden die Signale vor der Berechnung automatisch neu abgetastet. Die Abtastrate (XBase) des Signals 'Condition' ist ausschlaggebend.

Um ein Ergebnis mit einer bestimmten Abtastrate zu erhalten, können Sie das Signal 'Condition' mit der Funktion *Resample* anpassen.

Tipp

Wenn Sie für 'Condition' ein Analogsignal eintragen, wird als Bedingung abgefragt, ob die Größe größer als (TRUE) oder kleiner als (FALSE) 0,5 ist.

3.4.2 Switch

```
Switch ('Selector_Expression','Case_1_Expression','Value_1_Expression',
                              'Case_2_Expression','Value_2_Expression',
                              …
                              'Case_n_Expression','Value_n_Expression',
                              'Default_Value_Expr')
```

Argumente

'Selector_Expression'	Ausdruck, welcher auf verschiedene Bedingungen überprüft wird
'Case_n_Expression'	Ausdruck, welcher mit 'Selector_Expression' verglichen wird
'Value_n_Expression'	Ergebnis, falls 'Selector_Expression' und 'Case_n_Expression' übereinstimmen
'Default_Value_Expr'	Ergebnis, falls keiner der 'Case_n_Expression' mit 'Selector_Expression' übereinstimmt

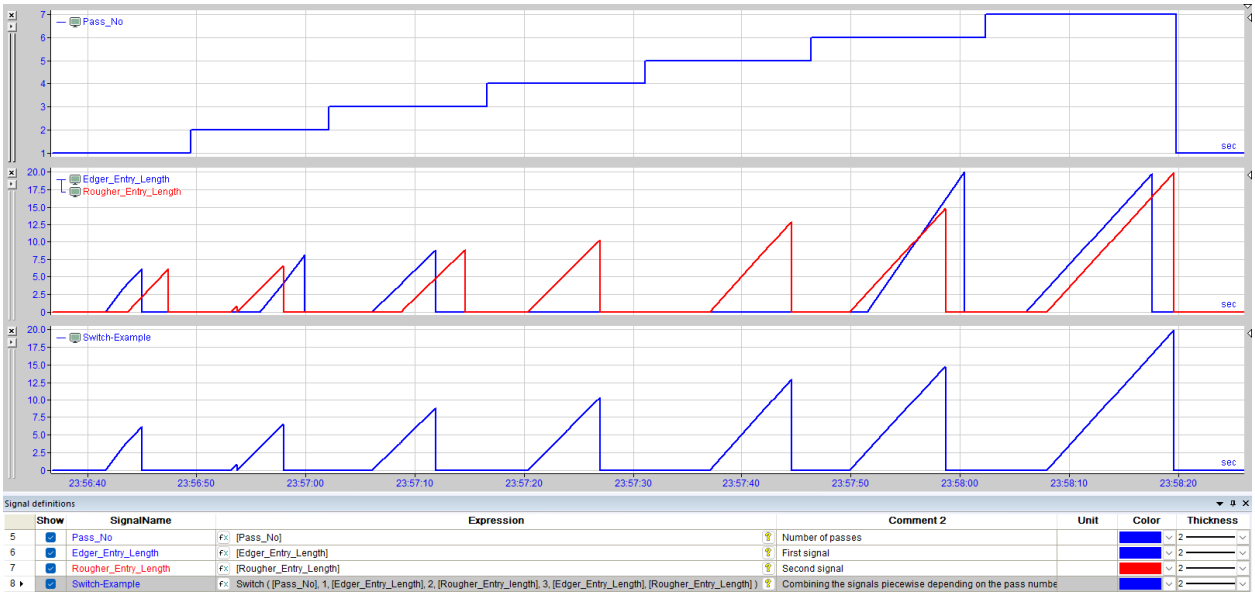
Beschreibung
Diese Anweisung vergleicht eine eingehende 'Selector_Expression' mit beliebig vielen 'Case_n_Expression', angelehnt an das SQL Statement CASE. Es werden mindestens 3 Argumente benötigt. Bei einer geraden Anzahl von Argumenten wird automatisch das letzte als 'Default_Value_Expr' interpretiert, welches herangezogen wird, wenn keine der 'Case_n_Expression' zur 'Selector_Expression' passt.

Falls 'Selector_Expression' und 'Case_n_Expression' zusammenpassen, wird die dazugehörige 'Value_n_Expression' zurückgegeben. Wenn mehrere 'Case_n_Expression' zum Eingangssignal passen, wird automatisch die erste ausgewählt.

Als 'Selector_Expression' sind folgende Signale zulässig:

- eine numerische Konstante
- eine Textkonstante
- ein äquidistant oder nicht-äquidistant gesampeltes Signal
- ein Textsignal

Grundsätzlich müssen die Typen der Vergleichswerte zusammenpassen, ansonsten wird der entsprechende Fall nicht ausgewählt.



Signal definitions

	Show	SignalName	Expression	Comment 2	Unit	Color	Thickness
5	☑	Pass_No	[Pass_No]	Number of passes			2
6	☑	Edger_Entry_Length	[Edger_Entry_Length]	First signal			2
7	☑	Rougher_Entry_Length	[Rougher_Entry_Length]	Second signal			2
8	☑	Switch-Example	Switch ([Pass_No], 1, [Edger_Entry_Length], 2, [Rougher_Entry_length], 3, [Edger_Entry_Length], [Rougher_Entry_Length])	Combining the signals piecewise depending on the pass numbe			2

3.5 Flankenerkennung

3.5.1 OneShot

```
OneShot('Expression')
```

Beschreibung
Diese Funktion liefert das Ergebnis TRUE, wenn der aktuelle Messwert von 'Expression' ungleich dem vorigen ist.

Die Funktion liefert das Ergebnis FALSE, wenn der aktuelle Messwert gleich dem vorigen ist.

Tipp

Die Funktion arbeitet auch mit nicht-äquidistanten Messwerten.

3.5.2 SetReset

```
SetReset('Set','Reset','SetDominant=1')
```

Argumente

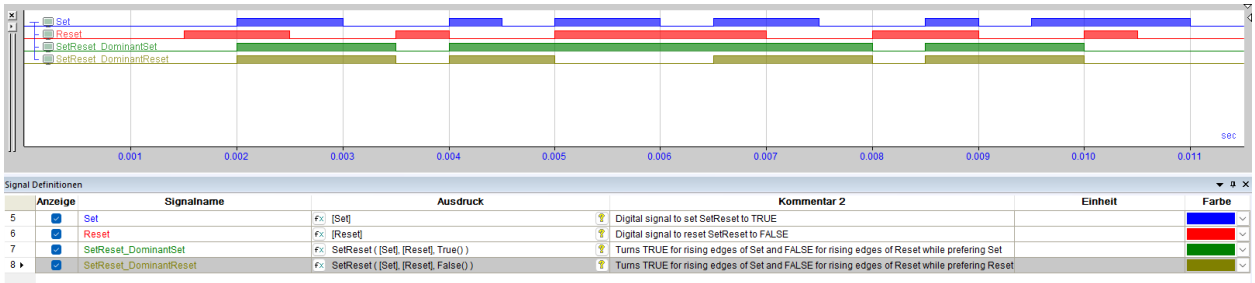
'Set'	positive Flanke setzt Funktion auf TRUE	
'Reset'	positive Flanke setzt Funktion auf FALSE	
'SetDominant'	Optionaler Parameter (Voreinstellung = 1), der steuert, welches Eingangsargument dominant ist, wenn beide Argumente gleichzeitig eine positive Flanke erhalten.	
	'SetDominant' = 1	Set hat Vorrang gegenüber Reset
	'SetDominant' = 0	Reset hat Vorrang gegenüber Set

Beschreibung
Mit dieser Funktion können Sie ein binäres Ergebnis (TRUE/FALSE) mithilfe von positiven Flanken (Übergang von 0 zu 1) der Argumente 'Set' und 'Reset' steuern.

Eine positive Flanke des Operanden 'Set' gibt als Ergebnis ein statisches TRUE wieder. Eine positive Flanke des Operanden 'Reset' setzt das Ergebnis auf FALSE zurück. Das Argument 'SetDominant' ist optional und bestimmt die Dominanz von 'Set' bzw. 'Reset'.

Tipp

Bei einem analogen Signal entspricht das Überschreiten des Werts 0,5 einer positiven Flanke.



3.5.3 F_Trig und R_Trig

F_Trig

```
F_Trig('Expression')
```

Beschreibung

Diese Funktion gibt TRUE für ein Sample zurück, wenn in 'Expression' ein Übergang von TRUE zu FALSE (fallende Flanke) stattfindet.

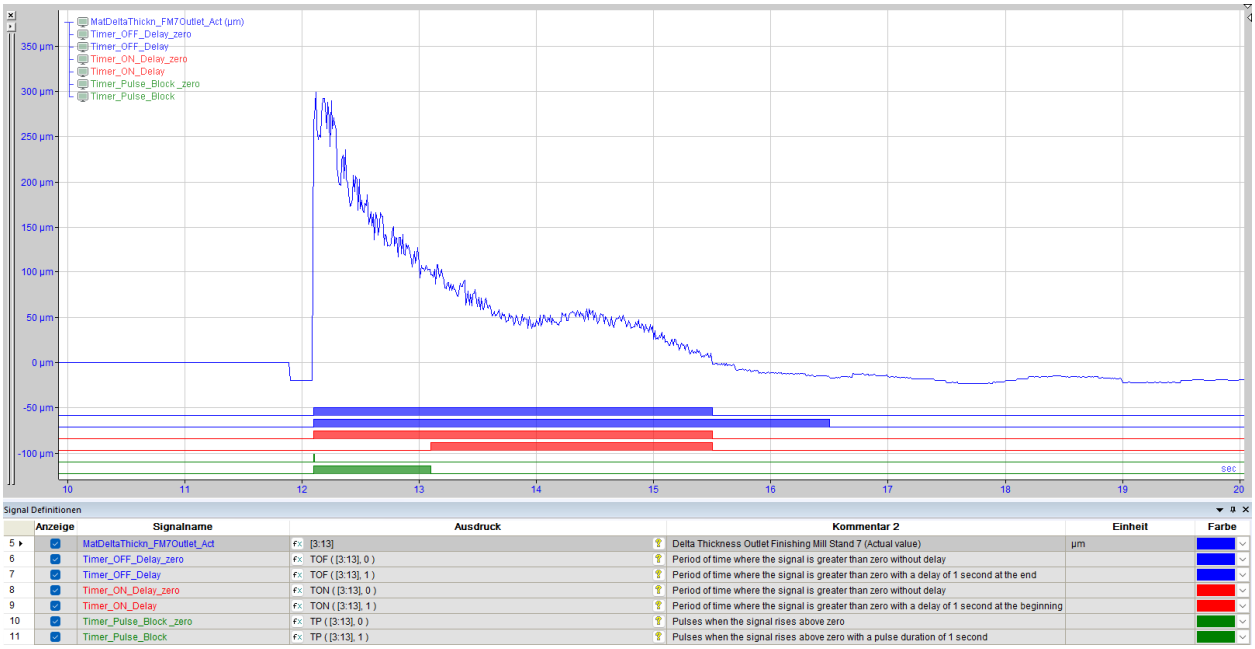
R_Trig

```
R_Trig('Expression')
```

Beschreibung

Diese Funktion gibt TRUE für ein Sample zurück, wenn in 'Expression' ein Übergang von FALSE nach TRUE (steigende Flanke) stattfindet.

3.6 Timer-Funktionen (IEC 61131-3)



Signal Definitionen

	Anzeige	Signalname	Ausdruck	Kommentar 2	Einheit	Farbe
5	☑	MatDeltaThickn_FM7Outlet_Act	[3:13]	Delta Thickness Outlet Finishing Mill Stand 7 (Actual value)	µm	
6	☑	Timer_OFF_Delay_zero	TOF ([3:13], 0)	Period of time where the signal is greater than zero without delay		
7	☑	Timer_OFF_Delay	TOF ([3:13], 1)	Period of time where the signal is greater than zero with a delay of 1 second at the end		
8	☑	Timer_ON_Delay_zero	TON ([3:13], 0)	Period of time where the signal is greater than zero without delay		
9	☑	Timer_ON_Delay	TON ([3:13], 1)	Period of time where the signal is greater than zero with a delay of 1 second at the beginning		
10	☑	Timer_Pulse_Block _zero	TP ([3:13], 0)	Pulses when the signal rises above zero		
11	☑	Timer_Pulse_Block	TP ([3:13], 1)	Pulses when the signal rises above zero with a pulse duration of 1 second		

TOF

```
TOF('in','pt')
```

Beschreibung

Ausschaltverzögerung. Der Ausgang wird 'pt' Sekunden nach dem Ausschalten des Eingangs 'in' ausgeschaltet.

TON

```
TON('in','pt')
```

Beschreibung

Einschaltverzögerung. Der Ausgang wird 'pt' Sekunden nach dem Einschalten des Eingangs 'in' eingeschaltet.

TP

```
TP('in','pt')
```

Beschreibung

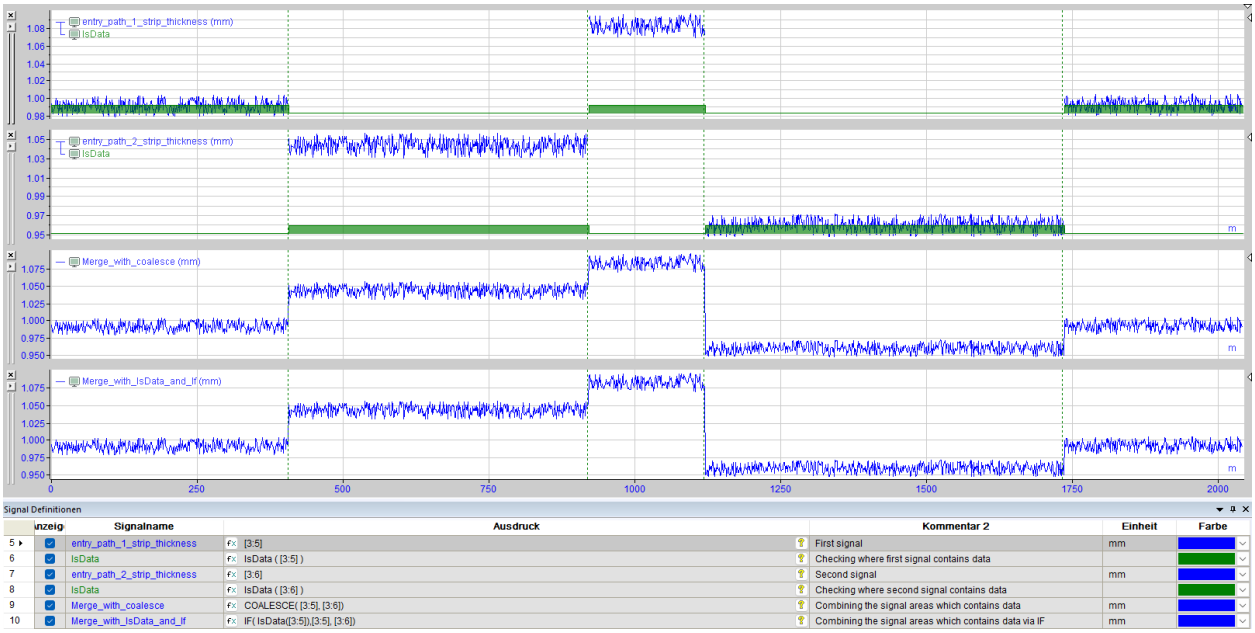
Impuls-Funktion. Der Ausgang wird für 'pt' Sekunden nach steigender Flanke an Eingang 'in' eingeschaltet.

Hinweis

Durch die verzögerte Ausschaltung können mehrere Impulse verschmelzen, wenn das verzögerte Ende mit dem Beginn eines anderen Impulses überlappt.

Eine weitere steigende Flanke während des Ausgangs-Impulses verlängert die Schaltung nicht und startet den Impuls nicht neu.

3.7 IsData und Coalesce



IsData

```
IsData('Expression','End')
```

Argumente

'Expression'	Eingangssignal oder Ausdruck
'End'	Länge des Ergebnissignals

Beschreibung

Das Ergebnis dieser Operation ist TRUE, wenn für 'Expression' Messwerte vorhanden sind. Das Ergebnis ist FALSE, wenn Messwerte fehlen bzw. Signale leer sind. Diese Funktion können Sie beispielsweise als Bedingung für andere Berechnungen verwenden.

Optional können Sie den Parameter 'End' angeben. Dieser Parameter kann das Ergebnissignal der Funktion verkürzen oder verlängern, sodass das Ergebnissignal mit anderen Signalen übereinstimmt und für weitere Verknüpfungen genutzt werden kann. Geben Sie den Parameter 'End' nicht an, dann entspricht die Länge des Ergebnissignals der des Eingangssignals (inkl. ungültiger Samples).

Coalesce

```
Coalesce('Candidate1','Candidate2',…)
```

Beschreibung

Inspiriert von der entsprechenden SQL-Abfrage gibt die Funktion *Coalesce* das erste ihrer Argumente zurück, welches Daten enthält. Ab *ibaAnalyzer* v8.3.0 oder höher arbeitet diese Funktion Sample-weise.

Sie können diese Funktion beispielsweise nutzen, um eine Absicherung gegen fehlende Signale in einer DAT-Datei zu schaffen.

3.8 IsNE

```
IsNE('Expression')
```

Beschreibung
Die Funktion zeigt an, ob ein Signal nicht-äquidistant abgetastete Daten enthält. Sie gibt TRUE zurück, wenn die durch die Auswertung von 'Expression' geladenen Daten nicht-äquidistant abgetastet sind, andernfalls FALSE.

Diese Funktion ist besonders nützlich bei der Verarbeitung von nicht-äquidistant abgetasteten HD-Daten, sie funktioniert jedoch auch mit anderen Datenquellen, z. B. Signalen aus Messdateien oder Ergebnissen der *XY*-Funktion.

4 Mathematische Funktionen

4.1 Grundrechenarten

4.1.1 Grundrechenarten +, -, *, /

```
z. B. 'Expression1' + 'Expression2'
```

Beschreibung
Sie können alle Signale und Ausdrücke mit den Grundrechenarten verwenden (Addition, Subtraktion, Multiplikation und Division).

Wenn Sie digitale Signale oder Ausdrücke als Operanden mit den Grundrechenarten verwenden, dann übersetzt *ibaAnalyzer* die Werte TRUE mit 1.0 und FALSE mit 0.0.

Das Ergebnis einer Grundrechenoperation ist immer ein analoger Ausdruck.

4.1.2 Abs

```
Abs('Expression')
```

Beschreibung
Die Absolutfunktion gibt den Absolutwert (= |Betrag|) von 'Expression' zurück.

Tipp

Interpolierte Werte bei einem Vorzeichenwechsel zwischen zwei Messpunkten können sich im Betrag unterscheiden.

4.1.3 Mod

```
Mod('Dividend','Divider')
```

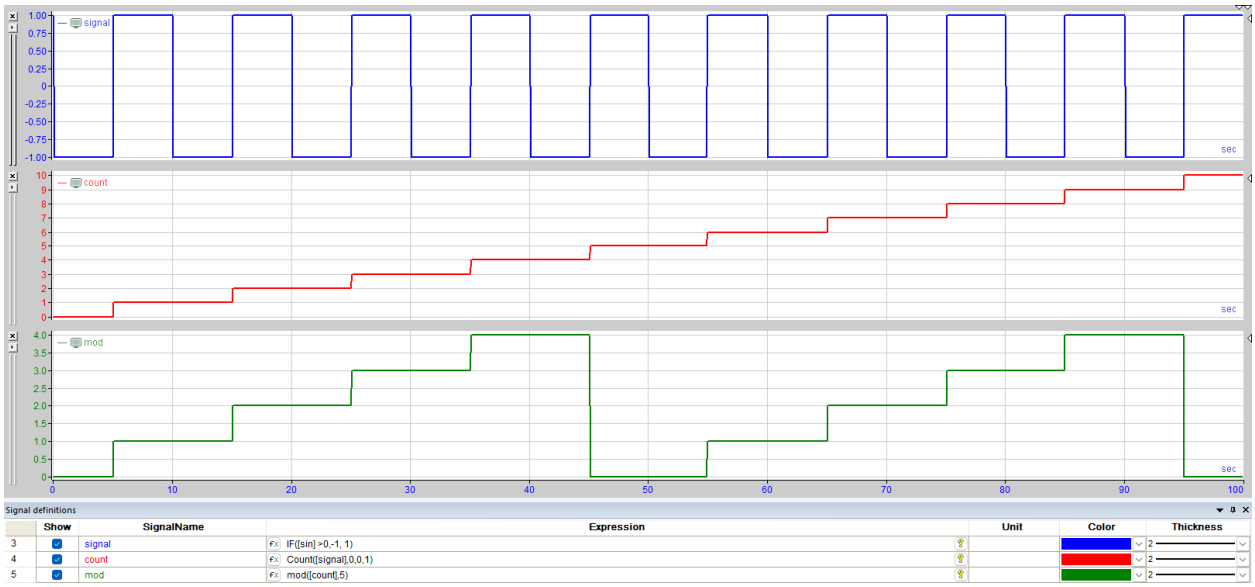
Beschreibung
Diese Funktion liefert als Ergebnis den Modulo von 'Dividend' und 'Divider'. Die Funktion verwendet intern die C-Funktion fmod, die es gestattet, Gleitkommawerte für 'Dividend' und 'Divider' zu verwenden.

Der Modulo r ist der Rest der Division 'Dividend'/'Divider', so dass umgekehrt gilt:

Dividend = Divider * x + r, wobei x eine ganze Zahl (Integer) ist.

Modulo r hat stets das gleiche Vorzeichen wie 'Dividend' und der Absolutwert von r ist stets kleiner als der Absolutwert von 'Divider'.

Wenn 'Dividend' < 'Divider' ist, dann liefert die Funktion den Wert 'Dividend' als Ergebnis. Der Rest kann in der mathematischen Sprache auch als "Dividend modulo Divider" bezeichnet werden.



4.1.4 Ceiling, Floor und Round

Ceiling

```
Ceiling('Expression')
```

Beschreibung

Diese Funktion gibt den kleinsten ganzzahligen Wert wieder, der größer oder gleich 'Expression' ist.

Floor

```
Floor('Expression')
```

Beschreibung

Diese Funktion gibt den größten ganzzahligen Wert wieder, der kleiner oder gleich 'Expression' ist.

Round

```
Round('Expression')
```

Beschreibung

Diese Funktion rundet 'Expression' auf die nächste Ganzzahl (Integerwert) auf oder ab.

Tipp

Mit folgendem Ausdruck können Sie auf 2 Nachkommastellen runden:

```
ROUND(100 * [Signal]) / 100
```

4.2 Integral- und Differenzialrechnung

4.2.1 Int

```
Int('Expression','Reset')
```

Argumente

'Expression'	Messwert	
'Reset'	Optionaler binärer Parameter zum Zurücksetzen des Integrals bzw. Unterdrücken des Integrationsvorgangs. 'Reset' kann selbst auch ein Ausdruck sein.	
	'Reset' > 0	Integral wird zurückgesetzt.
	'Reset'= 0	Integration freigegeben. (Voreinstellung)

Beschreibung
Diese Funktion liefert als Ergebnis das Integral von 'Expression' zurück. Sie können mit dem Parameter 'Reset' das Integrals bzw. Unterdrücken des Integrationsvorgangs zurücksetzen, z. B. um bei periodischen oder reversierenden Vorgängen dasselbe Signal mehrfach zu integrieren.
'Reset' kann selbst auch ein Ausdruck sein.

4.2.2 Diff (ehemals Dif)

```
Diff('Expression','dy'=FALSE)
```

Beschreibung
Diese Funktion liefert als Ergebnis die Ableitung (oder das Differential) von 'Expression' zurück.

Wenn Sie den optionalen Parameter 'dy' auf TRUE() setzen, wird statt dem Differential nur die Differenz zwischen den Messwerten berechnet.

Beispiel
Wenn 'Expression' ein längenbasiertes Signal ist, dann können Sie daraus mit der *Diff*-Funktion den Geschwindigkeitsverlauf ermitteln.

4.3 Potenzen und Wurzeln

4.3.1 Pow

```
Pow('Expression1','Expression2')
```

Argumente

'Expression1'	Basis
'Expression2'	Exponent

Beschreibung
Diese Funktion potenziert 'Expression1' (Basis) mit 'Expression2' (Exponent).

Beispiel
Berechnung einiger wichtiger Potenzen

(2)^0 = Pow(2, 0) = 1

(2)^-2 = Pow(2, -2) = 0,25

(-2)^2 = Pow(-2, 2) = 4

(10)^(lg 2) = Pow(10, Log10(2)) = 2

(0)-1 = Pow(0, -1) = +∞ (unendlich)

Hinweis

Die Potenzierung von 0 mit -1 führt zwar nicht zu einer Fehlermeldung, liefert jedoch auch kein Ergebnis.

4.3.2 Sqrt

```
Sqrt('Expression')
```

Beschreibung
Diese Funktion liefert als Ergebnis die Quadratwurzel von 'Expression'.

Hinweis

Negative Werte für 'Expression' führen zwar nicht zu einer Fehlermeldung, liefern aber auch kein Ergebnis.

4.4 e-Funktion und Logarithmen

4.4.1 Exp

```
Exp('Expression')
```

Beschreibung
Diese Funktion berechnet den Ausdruck $e^{'Expression'}$.

4.4.2 Log

```
Log('Expression')
```

Beschreibung
Diese Funktion liefert als Ergebnis den natürlichen Logarithmus von 'Expression'.

Hinweis

Negative Werte für 'Expression' führen zwar nicht zu einer Fehlermeldung, liefern aber auch kein Ergebnis.

4.4.3 Log10

```
Log10('Expression')
```

Beschreibung
Diese Funktion liefert als Ergebnis den dekadischen Logarithmus von 'Expression'.

Hinweis

Negative Werte für 'Expression' führen zwar nicht zu einer Fehlermeldung, liefern aber auch kein Ergebnis.

4.5 Pi

```
Pi()
```

Beschreibung
Die Zahl Pi ist als Konstante (π = 3.1415927…) für diverse Berechnungen im System hinterlegt. Mit dieser Funktion fügen Sie die Zahl π in die Berechnung ein.

4.6 Sum

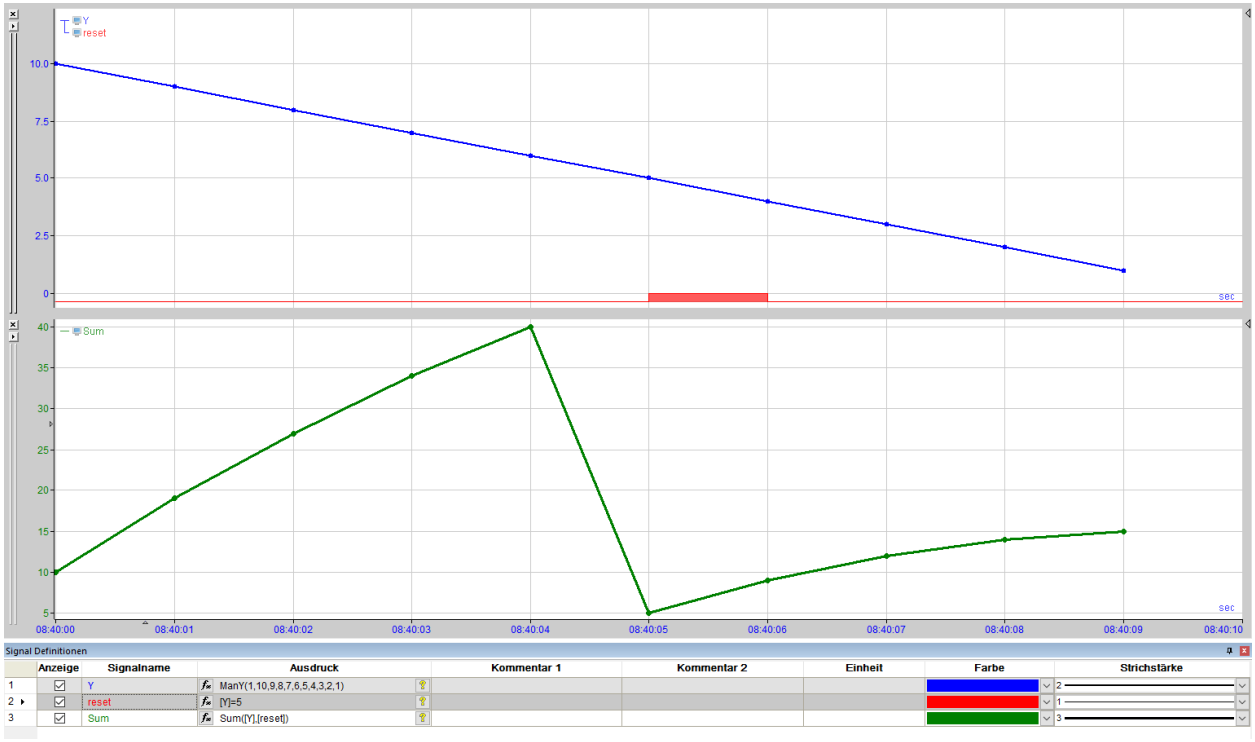
```
Sum('Expression','Reset'=FALSE)
```

Beschreibung

Diese Operation summiert alle Signalpunkte einer Funktion Punkt für Punkt. Wenn die Summenbildung durch einen Reset-Wert unterbrochen wird, beginnt anschließend die Summenbildung erneut.

Beispiel

Die Summenbildung beginnt mit dem Signalpunkt 10 + 9 + 8 + … 6. Hier erfolgt durch 'Reset' = TRUE() eine Unterbrechung und die Funktion wird auf Null gesetzt. Anschließend beginnt die Summenbildung erneut.



5 Trigonometrische Funktionen

`z. B. Sin('Expression')`

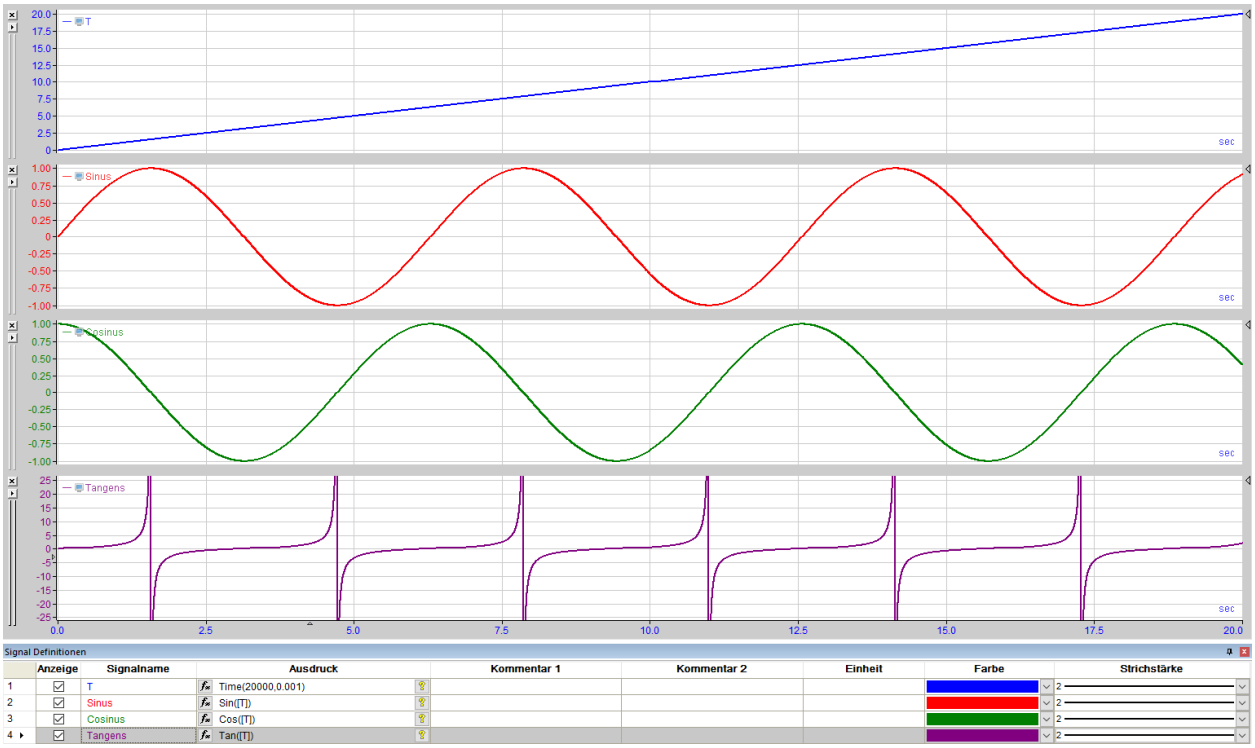
Sin('Expression')	Diese Funktion liefert als Ergebnis den Sinus von 'Expression' in rad.
Cos('Expression')	Diese Funktion liefert als Ergebnis den Cosinus von 'Expression' in rad.
Tan('Expression')	Diese Funktion liefert als Ergebnis den Tangens von 'Expression' in rad.
Asin('Expression')	Diese Funktion liefert als Ergebnis den Arkussinus von 'Expression' in rad.
Acos('Expression')	Diese Funktion liefert als Ergebnis den Arkuscosinus von 'Expression' in rad.
Atan('Expression')	Diese Funktion liefert als Ergebnis den Arkustangens von 'Expression' in rad.
Atan2('X','Y')	Diese Funktion liefert als Ergebnis den Arkustangens von 'Y'/'X'.

Beschreibung

Für die verschiedensten Berechnungen, in denen trigonometrische Funktionen benötigt werden, z. B. Leistungsermittlung in Wechselstromsystemen, stehen die Standardfunktionen und die entsprechenden Umkehrfunktionen zur Verfügung.

Beispiel

Darstellung der trigonometrischen Funktionen:



6 Statistische Funktionen

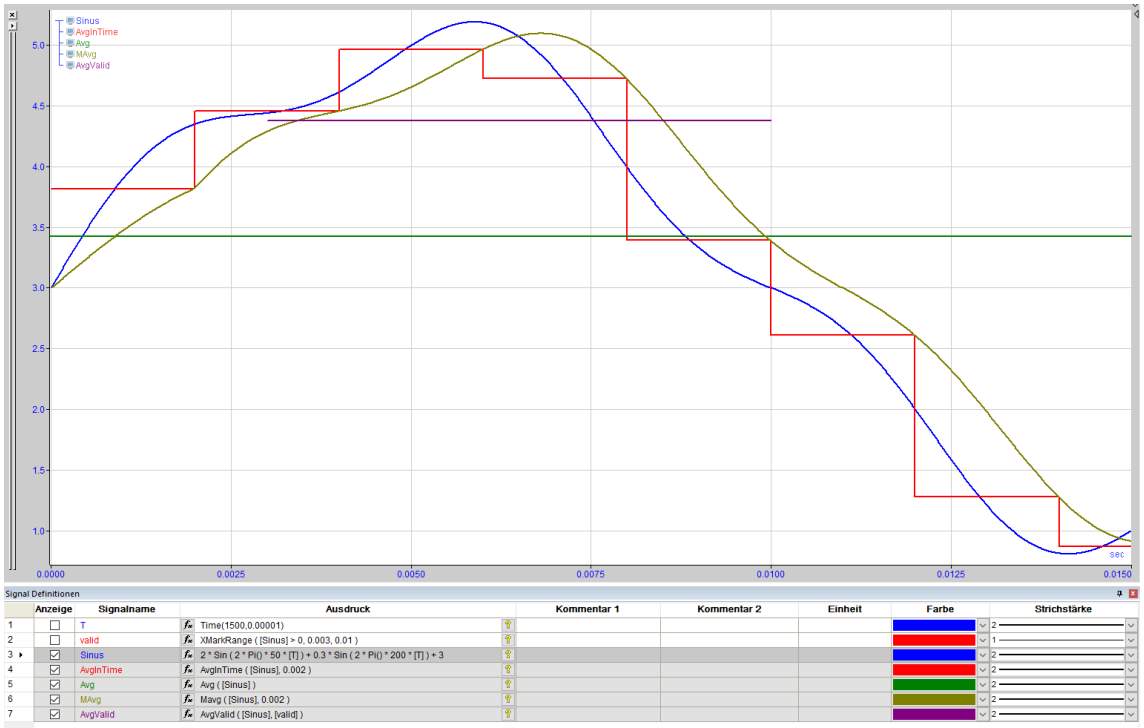
ibaAnalyzer unterstützt die Berechnung verschiedener statistischer Funktionen. Für alle Funktionen sind im Normalfall vier verschiedene Varianten verfügbar:

- Die Standardfunktion berechnet die entsprechende Größe immer über das gesamte Signal.
- Das Suffix *InTime* kennzeichnet, dass die entsprechende Größe über Intervalle einer angegebenen Länge gebildet wird.
- Das Suffix *Valid* wird benutzt, um die entsprechende Größe über Intervalle zu berechnen, die mit einem Binärsignal gekennzeichnet werden.
- Das Präfix *M* kennzeichnet, dass die entsprechende Größe über gleitende Intervalle einer angegebenen Breite gebildet wird.

Die Vektorstatistiken finden Sie bei den Vektorfunktionen, siehe ↗ *Vektor-Operationen*, Seite 71.

Die Funktion RMS finden Sie bei den elektrischen Funktionen, siehe ↗ *RMS und Eff*, Seite 81.

6.1 Mittelwert (Avg)



Avg

```
Avg('Expression')
```

Beschreibung

Diese Funktion liefert als Ergebnis den Mittelwert von 'Expression'. Das Ergebnis wird als konstanter Wert (horizontale Linie) im Signalstreifen angezeigt.

AvgInTime

```
AvgInTime('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den der Mittelwert gebildet wird
'Interval'	Länge des Intervalls, über das der Mittelwert berechnet wird

Beschreibung

Diese Funktion liefert als Ergebnis den Mittelwert je Zeitabschnitt der Länge 'Interval' von 'Expression'.

AvgValid

```
AvgValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, für den der Mittelwert gebildet wird
'Valid'	Kontrollsignal

Beschreibung

Diese Operation liefert als Ergebnis den Mittelwert von 'Expression' für das Intervall (Zeit oder Länge) zurück, in dem ein zugehöriges Kontrollsignal TRUE ist.

MAvg

```
MAvg('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den der Mittelwert gebildet wird
'Interval'	Länge des Intervalls, über das der Mittelwert gebildet wird

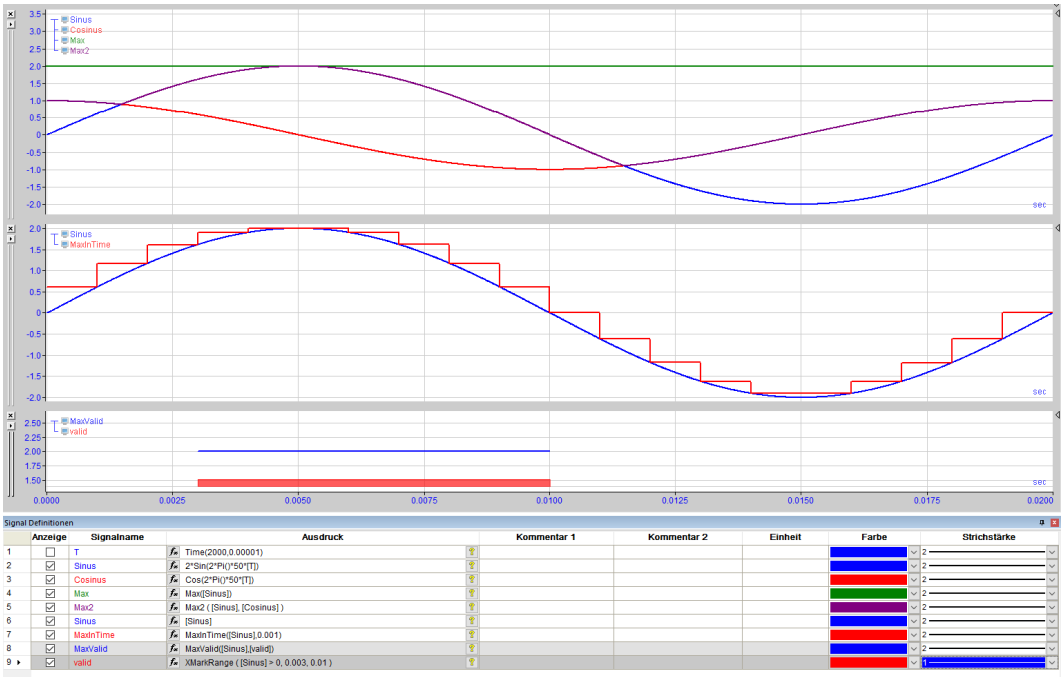
Beschreibung

Diese Funktion liefert als Ergebnis den gleitenden arithmetischen Mittelwert von 'Expression' berechnet über ein 'Interval' in Sekunden.

Tipp

Es können auch Signale oder Ausdrücke mit diesen Funktionen verarbeitet werden, die nicht zeitbasiert sind, d. h. die die Basis Länge, Frequenz oder 1/Länge haben. Anstelle von Sekunden ist dann der X-Achsen-Abschnitt entsprechend der Basis in m, Hz oder 1/m anzugeben.

6.2 Maxima (Max)



Max

`Max('Expression')`

Beschreibung

Diese Funktion liefert als Ergebnis den Maximalwert von 'Expression'. Das Ergebnis wird als konstanter Wert (horizontale Linie) im Signalstreifen angezeigt.

Max2

`Max2('Expression1','Expression2',…)`

Beschreibung

Diese Funktion liefert als Ergebnis das Maximum von allen angegebenen Signalen 'Expression1', 'Expression2' usw. Alle Signale werden Messwert für Messwert verglichen und der jeweils größere Wert wird als Ergebnis übergeben.

MaxInTime

`MaxInTime('Expression','Interval')`

Argumente

'Expression'	Signal oder Ausdruck, für den das Maximum gebildet wird
'Interval'	Länge des Intervalls, über das das Maximum berechnet wird

Beschreibung

Diese Funktion liefert als Ergebnis das Maximum von 'Expression' innerhalb jedes Intervalls der Länge 'Interval'. Die Funktion können Sie sowohl für zeitbasierte Signale ('Interval' in Sekunden) als auch für längenbasierte Signale ('Interval' in Metern) verwenden.

MaxValid

```
MaxValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, für den das Maximum gebildet wird
'Valid'	Kontrollsignal

Beschreibung

Diese Operation liefert als Ergebnis das Maximum von 'Expression' für das Intervall (Zeit oder Länge) zurück, in dem ein zugehöriges Kontrollsignal TRUE ist.

MMax

```
MMax('Expression','Interval')
```

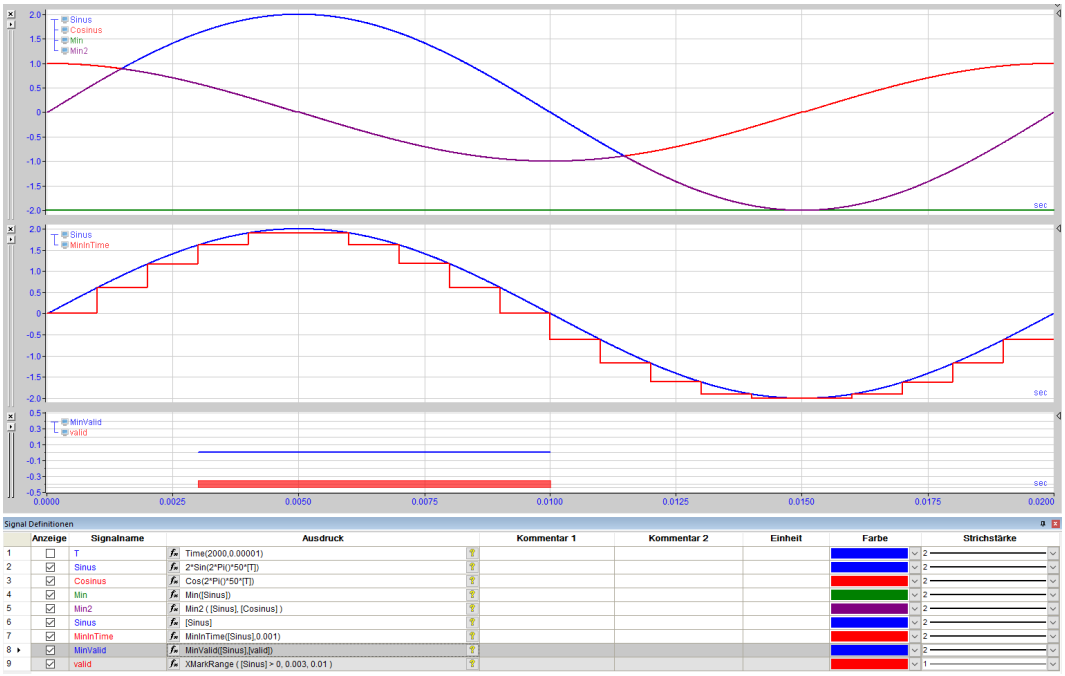
Argumente

'Expression'	Signal oder Ausdruck, für den das Maximum gebildet wird
'Interval'	Länge des Intervalls, über das das Maximum berechnet wird

Beschreibung

Diese Funktion liefert das Maximum von 'Expression' innerhalb eines gleitenden X-Achsenintervalls der Länge 'Interval', fortschreitend um jeweils einen Messpunkt.

6.3 Minima (Min)



Min

```
Min('Expression')
```

Beschreibung

Diese Funktion liefert als Ergebnis den Minimalwert des Signals 'Expression'. Das Ergebnis wird als konstanter Wert (horizontale Linie) im Signalstreifen angezeigt.

Min2

```
Min2('Expression1','Expression2',…)
```

Beschreibung

Diese Funktion liefert als Ergebnis das Minimum von allen angegebenen Signalen 'Expression1', 'Expression2' usw. Alle Signale werden Messwert für Messwert verglichen und der jeweils kleinere Wert wird als Ergebnis übergeben.

MinInTime

```
MinInTime('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den das Minimum gebildet wird
'Interval'	Länge des Intervalls, über das das Minimum berechnet wird

Beschreibung

Diese Funktion liefert als Ergebnis das Minimum von 'Expression' innerhalb jedes Intervalls der Länge 'Interval'. Die Funktion können Sie sowohl für zeitbasierte Signale ('Interval' in Sekunden) als auch für längenbasierte Signale ('Interval' in Metern) verwenden.

MinValid

```
MinValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, für den das Minimum gebildet wird
'Valid'	Kontrollsignal

Beschreibung

Diese Operation liefert als Ergebnis das Minimum von 'Expression' für das Intervall (Zeit oder Länge) zurück, in dem ein zugehöriges Kontrollsignal TRUE ist.

MMin

```
MMin('Expression','Interval')
```

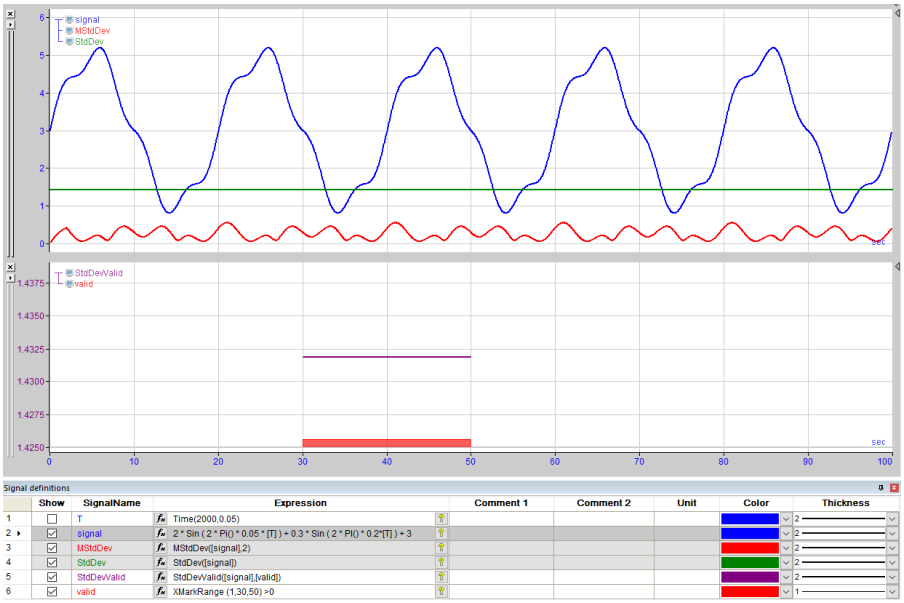
Argumente

'Expression'	Signal oder Ausdruck, für den das Minimum gebildet wird
'Interval'	Länge des Intervalls, über das das Minimum berechnet wird

Beschreibung

Diese Funktion liefert das Minimum von 'Expression' innerhalb eines gleitenden X-Achsenintervalls der Länge 'Interval', fortschreitend um jeweils einen Messpunkt.

6.4 Standardabweichung (StdDev)



StdDev

```
StdDev('Expression')
```

Beschreibung

Diese Funktion liefert als Ergebnis die Standardabweichung von 'Expression'.

Die Berechnung der Standardabweichung erfolgt entsprechend der Formel:

$$s_x = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$$

s_x = Standardabweichung

$\bar{x}$ = Mittelwert

n = Anzahl Messungen

StdDevInTime

```
StdDevInTime('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Standardabweichung gebildet wird
'Interval'	Länge des Intervalls, über das die Standardabweichung berechnet wird

Beschreibung

Diese Funktion liefert als Ergebnis die Standardabweichung von 'Expression' über jedes Zeitintervall der Länge 'Interval'.

Hinweis

Die Standardabweichung wird immer für das vorhergehende Intervall angegeben.

StdDevValid

```
StdDevValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Standardabweichung gebildet wird
'Valid'	Kontrollsignal

Beschreibung

Diese Funktion liefert als Ergebnis die Standardabweichung von 'Expression' für das Intervall (Zeit oder Länge) zurück, in dem ein zugehöriges Kontrollsignal TRUE ist.

MStdDev

```
MStdDev('Expression','Interval')
```

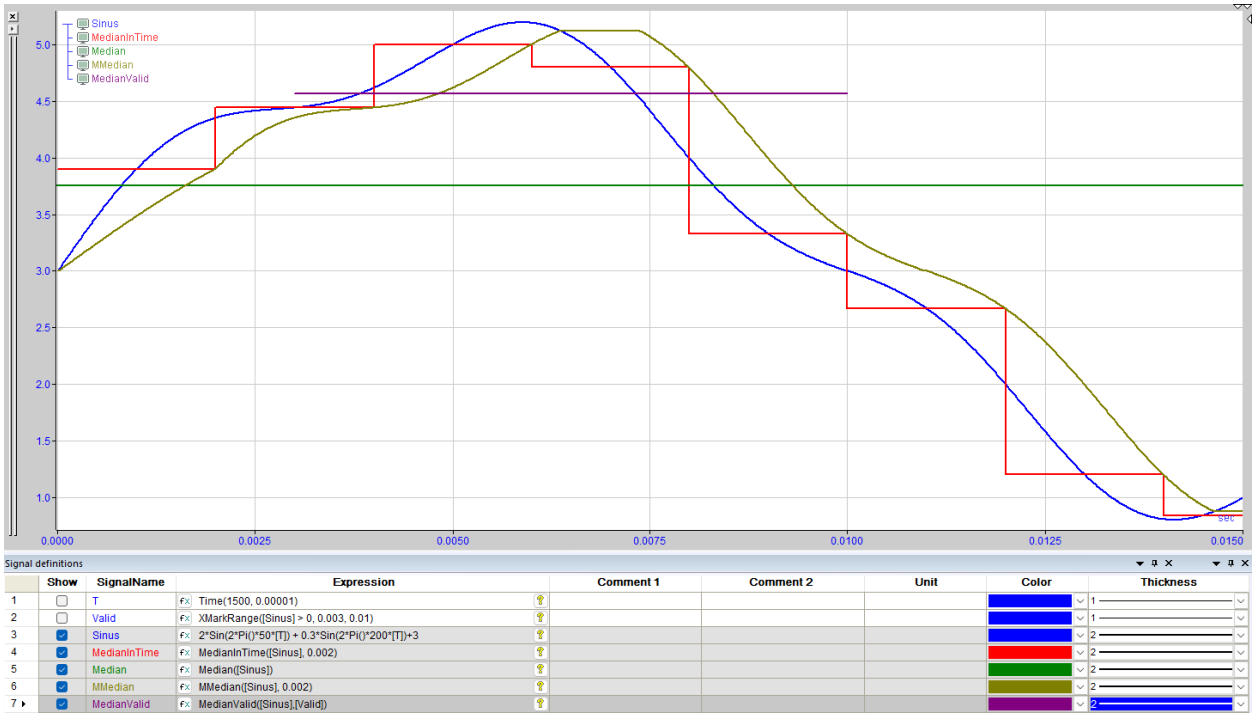
Argumente

'Expression'	Signal oder Ausdruck, für den die Standardabweichung gebildet wird
'Interval'	Länge des Intervalls, über das die Standardabweichung berechnet wird

Beschreibung

Diese Funktion liefert als Ergebnis die gleitende Standardabweichung von 'Expression' über jedes Zeitintervall der Länge 'Interval'. Die Funktion können Sie sowohl für zeitbasierte Signale ('Interval' in Sekunden) als auch für längenbasierte Signale ('Interval' in Metern) verwenden.

6.5 Median



Signal definitions

	Show	SignalName	Expression	Comment 1	Comment 2	Unit	Color	Thickness
1	☐	T	Time(1500, 0.00001)					1
2	☐	Valid	XMarkRange([Sinus] > 0, 0.003, 0.01)					1
3	☑	Sinus	2*Sin(2*Pi()*50*[T]) + 0.3*Sin(2*Pi()*200*[T])+3					2
4	☑	MedianInTime	MedianInTime([Sinus], 0.002)					2
5	☑	Median	Median([Sinus])					2
6	☑	MMedian	MMedian([Sinus], 0.002)					2
7	☑	MedianValid	MedianValid([Sinus],[Valid])					2

Median

```
Median('Expression')
```

Beschreibung

Diese Funktion liefert als Ergebnis den Median von 'Expression'. Der Median ist der Messwert, für den gilt, dass 50 % aller Messwerte kleiner und 50 % größer sind. Das Ergebnis wird als konstanter Wert (horizontale Linie) im Signalstreifen angezeigt.

MedianInTime

```
MedianInTime('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den der Median gebildet wird
'Interval'	Länge des Intervalls, über das der Median berechnet wird

Beschreibung

Diese Funktion liefert als Ergebnis den Median von 'Expression' innerhalb jedes Intervalls der Länge 'Interval'.

MedianValid

```
MedianValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, für den der Median gebildet wird
'Valid'	Kontrollsignal

Beschreibung

Diese Operation liefert als Ergebnis den Median von 'Expression' für das Intervall (Zeit oder Länge) zurück, in dem ein zugehöriges Kontrollsignal TRUE ist.

MMedian

```
MMedian('Expression','Interval')
```

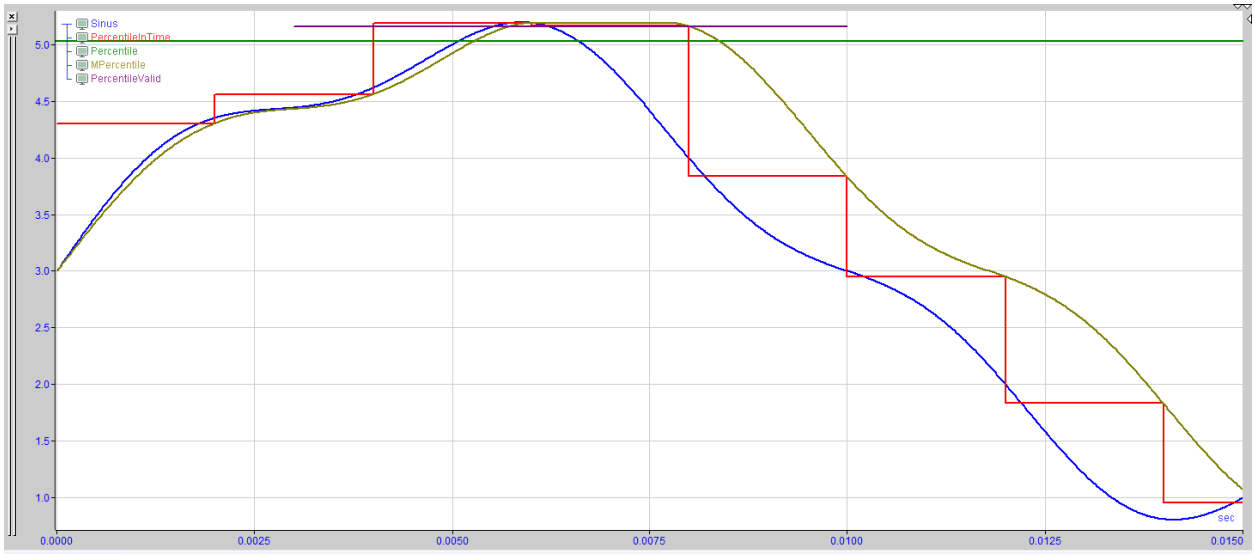
Argumente

'Expression'	Signal oder Ausdruck, für den der Median gebildet wird
'Interval'	Länge des Intervalls, über das der Median berechnet wird

Beschreibung

Diese Funktion liefert als Ergebnis den gleitenden Median von 'Expression' berechnet über ein 'Interval' in Sekunden.

6.6 Perzentile (Percentile)



Signal definitions

	Show	SignalName	Expression	Comment 1	Comment 2	Unit	Color	Thickness
1	☐	T	Time(1500, 0.00001)					1
2	☐	Valid	XMarkRange([Sinus] > 0, 0.003, 0.01)					1
3	☑	Sinus	2*Sin(2*Pi()*50*[T]) + 0.3*Sin(2*Pi()*200*[T])+3					2
4	☑	PercentileInTime	PercentileInTime([Sinus], 0.002, 0.9)					2
5	☑	Percentile	Percentile([Sinus], 0.9)					2
6	☑	MPercentile	MPercentile([Sinus], 0.002, 0.9)					2
7	☑	PercentileValid	PercentileValid([Sinus],[Valid], 0.9)					2

Percentile

```
Percentile('Expression','Percentile'=0.5)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Perzentile berechnet wird
'Percentile'	Perzentile

Beschreibung

Diese Funktion liefert als Ergebnis die definierte Perzentile von 'Expression'.

Die Angabe der p%-Perzentile bezeichnet den kleinsten Wert einer Menge von Messwerten, der größer ist als p% der Anzahl der Messwerte.

Eine typische Perzentile ist die 50%-Perzentile, der so genannte Median. Der Median teilt die Menge der Messwerte in zwei gleiche Hälften: 50 % aller Messwerte sind kleiner als der Median-Wert, die anderen 50 % sind größer oder gleichgroß. Weitere gebräuchliche Perzentilen sind 25 % und 75 %, die zusammen mit dem Median die Unterteilung einer Menge von Messwerten in vier Gruppen, die so genannten Quartile, erlauben. (< 25%, < 50%, < 75%, ≥ 75%).

Die Funktion *Percentile* ermittelt den Perzentilen-Wert aus der Gesamtmenge der Messpunkte eines Signals. Die Angabe der Perzentile muss als Dezimalwert erfolgen:

- 50 % --> 'Percentile' = 0.5 (Standardwert)
- 75 % --> 'Percentile' = 0.75
- 95,9 % --> 'Percentile' = 0.959

Diese Funktion ist insbesondere dafür geeignet, z. B. die Qualität eines Produktes zu beurteilen, indem eine bestimmte Produkteigenschaft einer festgelegten Klassifizierung genügen muss.

PercentileInTime

```
PercentileInTime('Expression','Interval','Percentile'=0.5)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Perzentile berechnet wird
'Interval'	Länge des Intervalls, über das die Perzentile gebildet wird
'Percentile'	Perzentile

Beschreibung

Diese Funktion liefert als Ergebnis die Perzentile von 'Expression' über jedes Zeitintervall der Länge 'Interval'.

PercentileValid

```
PercentileValid('Expression','Valid','Percentile'=0.5)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Perzentile berechnet wird
'Valid'	Länge des Intervalls, über das die Perzentile gebildet wird
'Percentile'	Perzentile

Beschreibung

Diese Funktion liefert als Ergebnis die Perzentile von 'Expression' für jedes Intervall (Zeit oder Länge), für das ein zugehöriges Kontrollsignal 'Valid' TRUE ist.

MPercentile

```
MPercentile('Expression','Interval','Percentile'=0.5)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Perzentile berechnet wird
'Interval'	Länge des gleitenden Intervalls, über das die Perzentile gebildet wird
'Percentile'	Perzentile

Beschreibung

Diese Funktion liefert als Ergebnis die gleitende Perzentile von 'Expression' über jedes Intervall der Länge 'Interval'. Die Funktion können Sie sowohl für zeitbasierte Signale ('Interval' in Sekunden) als auch für längenbasierte Signale ('Interval' in Metern) verwenden.

6.7 Korrelation und Kovarianz (Correl, CoVar)

Correl

```
Correl('Expression1','Expression2')
```

Argumente

'Expression1/2'	Signale oder Ausdrücke, für die der Korrelationskoeffizient berechnet wird

Beschreibung

Diese Funktion berechnet den Korrelationskoeffizienten zwischen 'Expression1' und 'Expression2'. Betrachtet wird die gesamte Aufzeichnungslänge. Die Funktion liefert einen konstanten Wert zurück.

Mcorrel

```
Mcorrel('Expression1','Expression2','Interval')
```

Argumente

'Expression1/2'	Signale oder Ausdrücke, für die der Korrelationskoeffizient berechnet wird
'Interval'	Länge des Intervalls, über das der Korrelationskoeffizient gebildet wird

Beschreibung

Diese Funktion berechnet den Korrelationskoeffizienten zwischen 'Expression1' und 'Expression2' über gleitende Intervalle der Größe 'Interval' gemessen in s, m, Hz oder 1/m.

CoVar

```
CoVar('Expression1','Expression2')
```

Argumente

'Expression1/2'	Signale oder Ausdrücke, für die die Kovarianz berechnet wird

Beschreibung

Diese Funktion berechnet die Kovarianz zwischen 'Expression1' und 'Expression2'. Betrachtet wird die gesamte Aufzeichnungslänge. Die Funktion liefert einen konstanten Wert zurück.

MCoVar

```
MCoVar('Expression1','Expression2','Interval')
```

Argumente

'Expression1/2'	Signale oder Ausdrücke, für die die Kovarianz berechnet wird
'Interval'	Länge des Intervalls, über das die Kovarianz gebildet wird

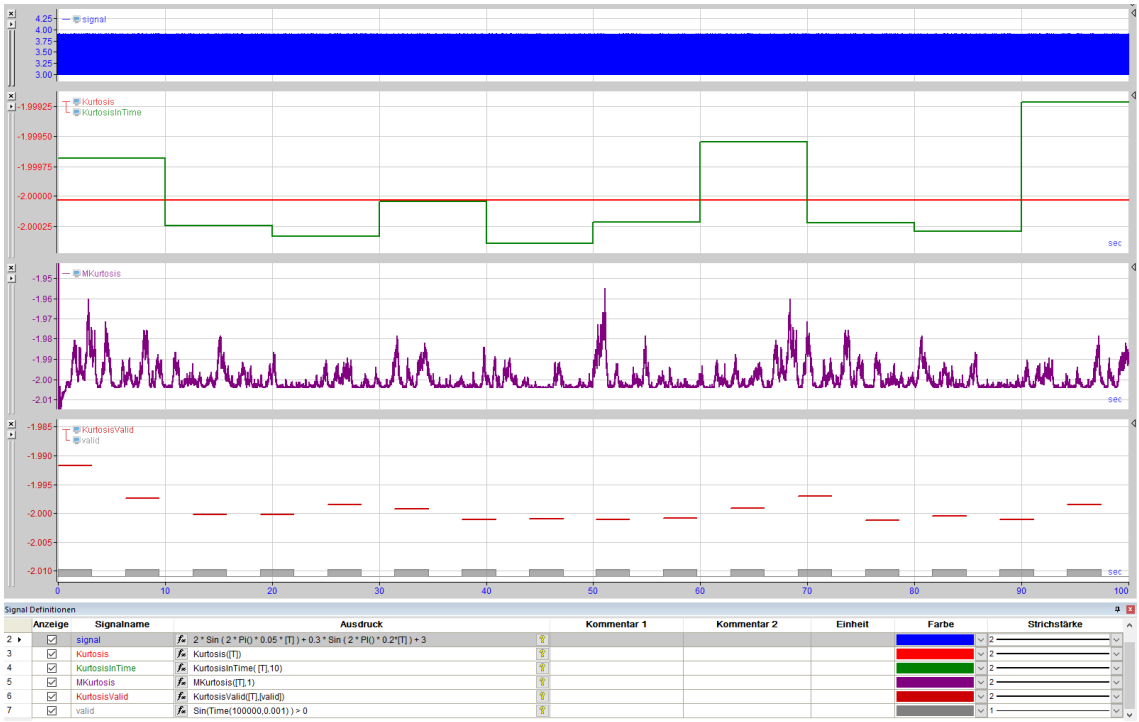
Beschreibung

Diese Funktion berechnet die Kovarianz zwischen 'Expression1' und 'Expression2' über gleitende Intervalle der Länge 'Interval', gemessen in s, m, Hz oder 1/m.

6.8 Kurtosis

Die Berechnung der *Kurtosis* wird z. B. für die Bewertung und Analyse von Schwingungen verwendet. Sie dient dazu, innerhalb eines Schwingungssignals die Anzahl von Ausreißern zu bestimmen.

Mathematisch gesehen ist die Kurtosis ein Maß für die relative "Flachheit" einer Verteilung (im Vergleich zur Normalverteilung, die eine Kurtosis von null aufweist). Eine positive Kurtosis zeigt eine spitz zulaufende Verteilung (eine so genannte leptokurtische Verteilung), wohingegen eine negative Kurtosis eine flache Verteilung (platykurtische Verteilung) anzeigt.



Diese statistische Methode eignet sich speziell zur Analyse zufälliger oder stochastischer Signale, z. B. in der zustandsorientierten Instandhaltung (Condition Monitoring) bei der Analyse von Schwingungen.

Zur Charakterisierung des Signalverlaufes werden Methoden der Wahrscheinlichkeitsdichte oder Häufigkeit verwendet. Dabei wird von der Annahme ausgegangen, dass nach dem Ausfiltern z. B. von drehfrequenten Schwingungsanteilen bei intakten Maschinen ein Rauschsignal mit einer Gauß'schen Amplitudenverteilung messbar ist. Diesem Signal überlagern sich bei auftretender Schädigung einzelne Impulssignale, welche die Verteilungsfunktion verändern. Durch die Bildung geeigneter Kennwerte, wie dem *Crest-Faktor* oder dem *Kurtosis-Faktor* kann eine Bewertung des Anlagenzustandes stattfinden.

Diese Verfahren bieten bei regelmäßigen Messungen einen Überblick über den Zustand der Maschine. Der Nachteil liegt jedoch darin, dass die Kennwerte, nachdem sie angestiegen sind, wieder sinken. Grund dafür ist, dass bei fortschreitender Schädigung die Anzahl der Impulssignale ansteigt. Das wiederum beeinflusst den Effektivwert, jedoch kaum den Spitzenwert.

Durch Stoßimpulse hervorgerufene Veränderungen des Zeitsignals bewirken eine Veränderung der sich ergebenen Verteilungsfunktion. Schädigungen mit ausgeprägtem diskretem Charakter lassen den *Kurtosis-Faktor* somit stark ansteigen. Sein Absolutwert lässt somit bereits Aussagen über eine Schädigung zu.

Die Berechnung der Kurtosis erfolgt nach ähnlichem Muster wie die Berechnung der Standardabweichung *StdDev*. *ibaAnalyzer* verwendet folgende Formel:

$$\frac{(n+1)n}{(n-1)(n-2)(n-3)} * \frac{\sum_{i=1}^{n}(x_i-\bar{x})^4}{s^4} - 3 * \frac{(n-1)^2}{(n-2)(n-3)}$$

n: Anzahl Messwerte

s: Standardabweichung

Kurtosis

```
Kurtosis('Expression')
```

Beschreibung

Diese Operation liefert als Ergebnis die Kurtosis des gewählten Zeitsignals (Ausdruck).

KurtosisInTime

```
KurtosisInTime('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Kurtosis gebildet wird
'Interval'	Länge des Intervalls, über das die Kurtosis berechnet wird

Beschreibung

Bei dieser Operation wird der ausgewählte Ausdruck in gleichlange Intervalle der Größe 'Interval' unterteilt. Für diese Intervalle erfolgt anschließend die Berechnung der Kurtosis.

KurtosisValid

```
KurtosisValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Kurtosis gebildet wird
'Valid'	Kontrollsignal

Beschreibung

Diese Operation gibt die Kurtosis für die Bereiche wieder, in denen ein zugehöriges Kontrollsignal TRUE ist.

MKurtosis

```
MKurtosis('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Kurtosis gebildet wird
'Interval'	Länge des Intervalls in Sekunden, über das die Kurtosis gebildet wird

Beschreibung

Bei dieser Operation wird die Kurtosis von 'Ausdruck' über ein festes aber gleitendes X-Achsenintervall berechnet.

6.9 Skewness

Ähnlich wie der Kurtosis-Faktor eignet sich der Skewness-Faktor zur Bewertung und Analyse von Schwingungen. Den Skewness-Faktor können Sie dann anwenden, wenn Sie z. B. die Symmetrieeigenschaften eines Schwingungssignals überprüfen wollen.

Mathematisch gesehen handelt es sich hierbei um die Beurteilung der Schiefe (Skewness) einer Verteilungsfunktion. Eine Verteilung wird rechtsschief (bzw. linkssteil) genannt, wenn der Hauptanteil der Verteilung auf der linken (bzw. rechten) Seite konzentriert ist. Der Grad der Schiefe wird durch das dritte Moment der Verteilung bestimmt.

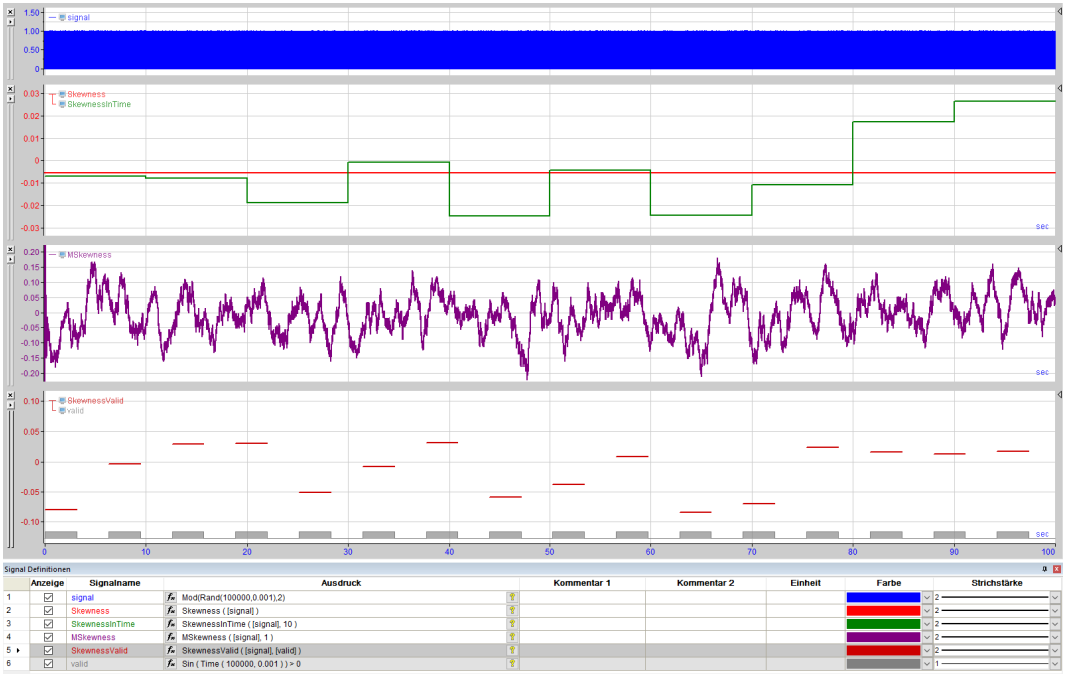
Für die Berechnung der Skewness wird ähnlich wie bei den Funktionen Kurtosis und Standardabweichung verfahren. *ibaAnalyzer* verwendet folgende Formel:

$$\frac{n}{(n-1)(n-2)} * \frac{\sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3}$$

n: Anzahl Messwerte

s: Standardabweichung

Die verschiedene Berechnungen der Skewness liefern folgende Ergebnisse:



Skewness

```
Skewness('Expression')
```

Beschreibung

Diese Operation liefert als Ergebnis die Skewness des gewählten Zeitsignals 'Expression'.

SkewnessInTime

```
SkewnessInTime('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Skewness gebildet wird
'Interval'	Länge des Intervalls, über das die Skewness berechnet wird

Beschreibung

Bei dieser Operation wird der ausgewählte Ausdruck in gleichlange Intervalle der Größe 'Interval' unterteilt. Für diese Intervalle erfolgt anschließend die Berechnung der Skewness.

SkewnessValid

```
SkewnessValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Skewness gebildet wird
'Valid'	Kontrollsignal

Beschreibung

Diese Operation gibt die Skewness für die Bereiche wieder, in denen ein zugehöriges Kontrollsignal TRUE ist.

MSkewness

```
MSkewness('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Skewness gebildet wird
'Interval'	Länge des gleitenden Intervalls, über das die Skewness berechnet wird

Beschreibung

Bei dieser Operation wird die Skewness von 'Expression' über ein festes aber gleitendes X-Achsenintervall berechnet.

6.10 CountSamples

```
CountSamples('Expression','Reset'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Anzahl der Signalpunkte ermittelt wird	
'Reset'	Optionaler binärer Parameter zum Zurücksetzen bzw. Unterdrücken des Zählvorgangs. 'Reset' kann selbst auch ein Ausdruck sein	
	'Reset'=TRUE()	Zählvorgang wird zurückgesetzt.
	'Reset'=FALSE()	Zählvorgang freigegeben. (Voreinstellung)

Beschreibung

Diese Funktion ermittelt die Anzahl der einzelnen Signalpunkte unabhängig davon, ob die Signalpunkte äquidistant sind oder nicht. Ungültige Signale werden nicht gezählt. Wenn das Eingangssignal ungültig ist, wird als Ergebnis der konstante Wert 0 ausgegeben.

Tipp

Diese Funktion können Sie beispielsweise auch in Kombination mit *XMarkValid* benutzen, siehe XMark-Funktionen ↗ *XMarkRange und XMarkValid*, Seite 63.

6.11 Sort und Xsort

Sort

```
Sort('Expression','Descending'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Samples sortiert werden
'Descending'	Optionaler binärer Parameter zum Umkehren der Sortierreihenfolge

Beschreibung

Diese Funktion sortiert alle Samples einer Kurve ('Expression') entsprechend ihrer Werte in aufsteigender Reihenfolge von links nach rechts.

Standardeinstellung: Sortierung in aufsteigender Reihenfolge ('Descending'=FALSE). Wenn Sie die Samples in absteigender Reihenfolge von links nach rechts sortieren wollen, setzen Sie 'Descending' auf TRUE.

XSort

```
XSort('Expression','Descending'=FALSE)
```

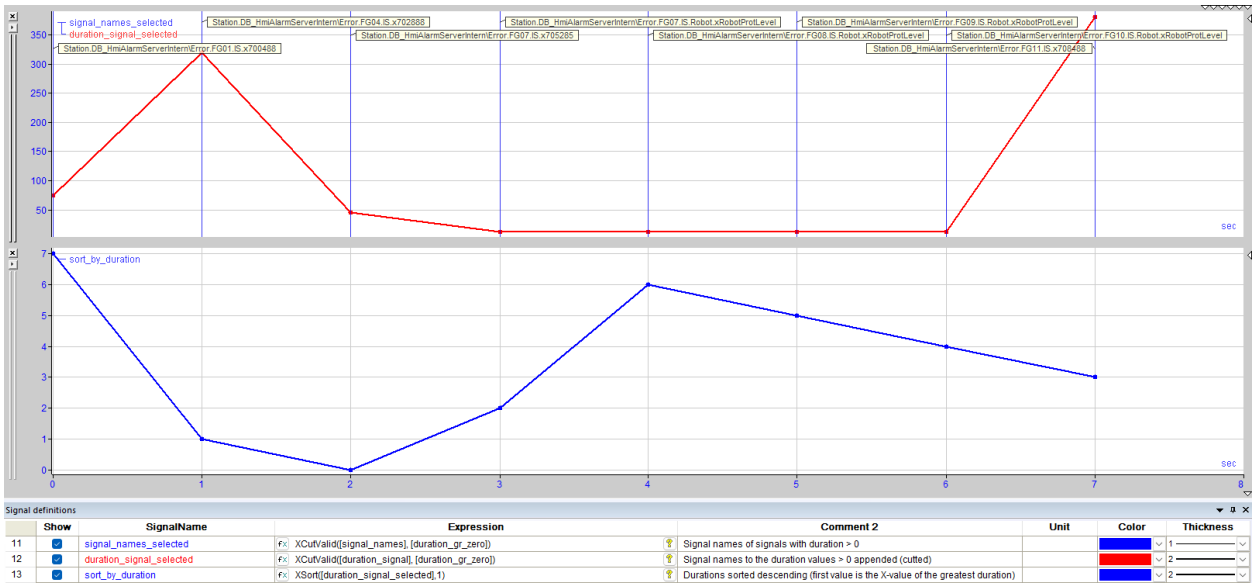
Argumente

'Expression'	Signal oder Ausdruck, für den die Samples sortiert werden
'Descending'	Optionaler binärer Parameter zum Umkehren der Sortierreihenfolge

Beschreibung

Diese Funktion sortiert alle (X,Y)-Paare einer Kurve ('Expression') entsprechend ihrer Y-Werte in aufsteigender Reihenfolge von links nach rechts und gibt die X-Werte zurück. Das heißt, der letzte Y-Wert des sortierten Signals entspricht dem X-Wert, an dem 'Expression' sein Maximum hat. Die X-Achse und Y-Achse des sortierten Signals sind gleich lang.

Standardeinstellung: Sortierung in aufsteigender Reihenfolge ('Descending'=FALSE). Wenn Sie die Paare in absteigender Reihenfolge von links nach rechts sortieren wollen (der letzte Wert entspricht dem X-Wert, an dem 'Expression' sein Minimum annimmt), setzen Sie 'Descending' auf TRUE.



7 Zeit-Länge-Funktionen

7.1 ConvertBase

```
ConvertBase('Expression','From','To')
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Basis geändert wird
'From' 'To'	Einstellung der Basis von bzw. zu der gewechselt wird 0 = Zeit 1 = Länge 2 = Frequenz 3 = inverse Länge

Beschreibung
Diese Operation wandelt einen Ausdruck von einer Basis zu einer anderen Basis. Es wird keine physikalische Umrechnung oder Skalierung vorgenommen.

Diese Funktion dient dazu, die Bezugsgröße eines Signals zu ändern. Dies kann dann von Vorteil sein, wenn für weitere Berechnungen beispielsweise längenbasierte Bezugsgrößen verwendet werden, das vorhandene Signal jedoch nur zeitbasiert vorliegt.

7.2 Neuabtasten (Resample)

```
Resample('Expression','Basis','interpolate'=TRUE)
```

Argumente

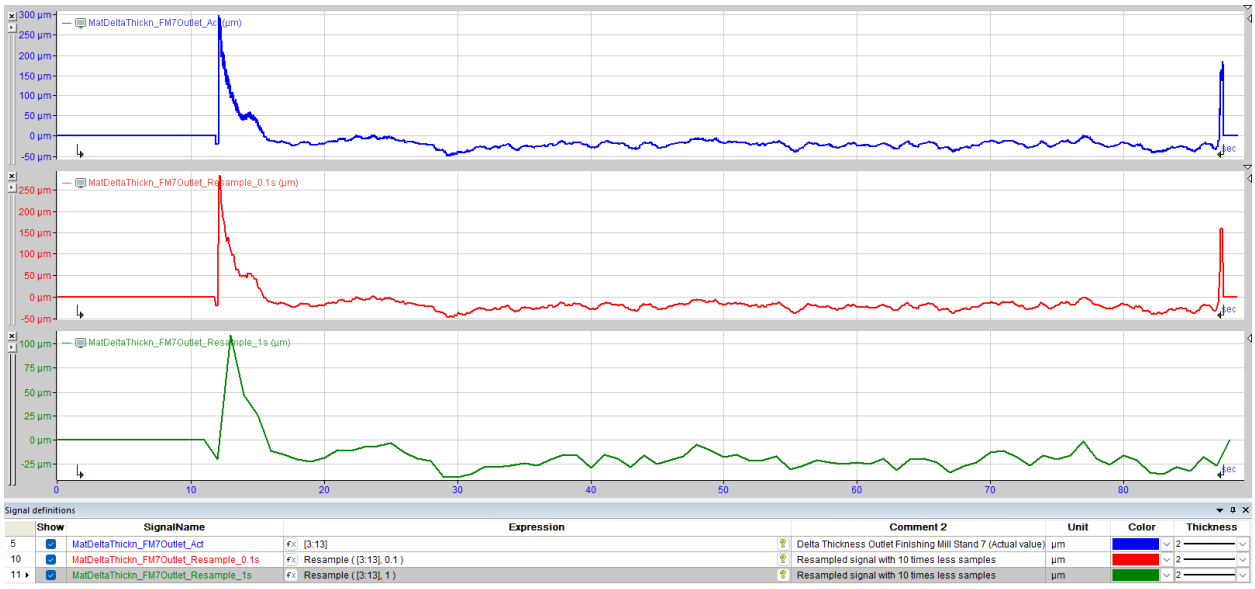
'Expression'	Signal oder Ausdruck, der neu abgetastet wird
'Basis'	Neue Abtastrate des Ergebnisses
'interpolate'	Optionaler Parameter, um die automatische Interpolation für die neuen Messwerte zu verhindern

Beschreibung

Diese Operation liefert als Ergebnis den Signalverlauf von 'Expression' auf einer neuen Zeitbasis. Dabei werden aus dem Originalverlauf die Momentanwerte entsprechend der neuen Zeitbasis zeitrichtig übertragen, so dass die Länge der neuen Kurve praktisch gleich ist. Die Funktion kann auch für längenbasierte Signale verwendet werden. Anstelle eines Zeitraums ist dann der Betrag eines Weges in m einzugeben.

Tipp

Eine Kurve lässt sich grafisch glätten, wenn in der Funktion *Resample* eine größere Zeitbasis verwendet wird, da weniger Punkte miteinander verbunden werden. Die Werte werden nicht gemittelt.



7.3 Zeitfunktionen

7.3.1 Time

```
Time('Count','Basis')
```

Argumente

'Count'	Anzahl der zu erzeugenden Messpunkte
'Basis'	Abtastrate des Ergebnissignals

Beschreibung
Diese Funktion liefert als Ergebnis ein lineares, zeitproportionales Signal mit einer Anzahl 'Count' Werte im Abstand von 'Basis'. Die Angabe der Zeitbasis versteht sich in Sekunden. Die Zeitwerte werden dabei sowohl auf der X-Achse als auch auf der Y-Achse abgetragen.

Hinweis

Für die Verwendung der Time-Funktion müssen Sie keine Messdatei laden.

7.3.2 InfoFieldTime

```
InfoFieldTime('FileIndex',"'InfoField'",'Begin'=0,'End'=Text end)
```

Argumente

'FileIndex'	Index der Messdatei
'InfoField'	Infofeld, das ausgelesen wird; in Anführungszeichen setzen.
'Begin'	Optional: Erstes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der komplette Inhalt ausgelesen.
'End'	Optional: Letztes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der Inhalt von 'Begin' bis zum letzten Zeichen ausgelesen.

Beschreibung
Diese Funktion interpretiert den Inhalt eines Infofeldes als Zeit, wenn möglich, und gibt die relative Zeit seit dem Start von *ibaAnalyzer* in Sekunden zurück. Das Ergebnis ist ein konstanter Wert im Signalstreifen.

Hinweis

Die Funktion *InfoFieldTime* erwartet, dass das angegebene Infofeld als Zeit (Datum, Stunden, Minuten, Sekunden, …) interpretiert werden kann. Wenn das nicht möglich ist, wird ein leerer Signalstreifen erzeugt.

Wenn das Infofeld als numerischer Wert bzw. als Text ausgelesen werden soll, müssen Sie die Funktionen *InfoField* bzw. *InfoFieldText* verwenden.

Tipp

Wenn Sie im Signalbaum einen Doppelklick auf das gewünschte Infofeld machen, das eine Zeit beschreibt (z. B. "starttime"), dann fügt *ibaAnalyzer* die entsprechende Funktion automatisch als neues Signal in die Signaltabelle ein. Anschließend passen Sie bei Bedarf nur noch den Signalnamen, Start und Ende an.

Diese Methode funktioniert auch im Eingabefeld des Editors. Die Funktion wird dann an der Cursorposition eingefügt.

7.3.3 AbsoluteTime und RelativeTime

AbsoluteTime

```
AbsoluteTime('Time','DoSync'=FALSE)
```

Argumente

'Time'	Relative Zeit, die konvertiert wird
'DoSync'	Optionaler Parameter, der festlegt, ob die absolute Zeit am Startzeitpunkt der aktuell angezeigten Zeitachse ausgerichtet wird

Beschreibung

Diese Funktion transformiert die relative Zeitinformation 'Time' (z. B. generiert mit *XFirst*, *XLast* oder *XValues*) in absolute Zeit. Dabei wird abhängig davon, ob die Eingangszeit konstant ist oder nicht, ein Vektor mit konstanten oder variierenden Einträgen zurückgegeben. Der optionale, binäre Parameter 'DoSync' legt fest, ob das Ergebnis am Startzeitpunkt der aktuell angezeigten Zeitachse ausgerichtet werden soll.

Das Ergebnis ist ein Vektor mit folgenden Einträgen, die mit *GetRows* ausgelesen werden können:

- Index 0: Millisekunden
- Index 1: Sekunden
- Index 2: Minuten
- Index 3: Stunden
- Index 4: Tag des Monats
- Index 5: Monat
- Index 6: Jahr
- Index 7: Tag des Jahres
- Index 8: Wochentag (1=Montag, 2=Dienstag, … , 7=Sonntag)

Hinweis

Wenn Sie für 'Time'=0 angeben, gibt die Funktion die Startzeit der Messdatei zurück oder die UNIX-Startzeit (01.01.1970; 00:00:00 Uhr), wenn keine Messdatei geladen ist.

RelativeTime

```
RelativeTime('Year','Month','Day','Hour','Minute','Second','Millisecond'=0)
```

Argumente

'Year'	Jahr der absoluten Zeit
'Month'	Monat der absoluten Zeit
'Day'	Tag der absoluten Zeit
'Hour'	Stunde der absoluten Zeit
'Minute'	Minute der absoluten Zeit
'Second'	Sekunde der absoluten Zeit
'Millisecond'	Millisekunde der absoluten Zeit

Beschreibung

Diese Funktion transformiert eine absolute Zeit (z. B. generiert mit *AbsoluteTime*) in relative Zeit, also die Zeit seit dem Start von *ibaAnalyzer* in Sekunden. Der optionale Parameter 'Millisecond' kann die Genauigkeit erhöhen. Das Ergebnis ist ein konstanter Wert im Signalstreifen.

7.4 Umrechnung von Zeit- auf Längenbezug (TimeToLength)

TimeToLength

```
TimeToLength('Expression','Speed','Precision')
```

Argumente

'Expression'	Signal oder Ausdruck, der auf Länge umgerechnet wird
'Speed'	Geschwindigkeitssignal
'Precision'	Optionaler Parameter, der die Abtastrate des Ergebnissignals in m festlegt

Beschreibung

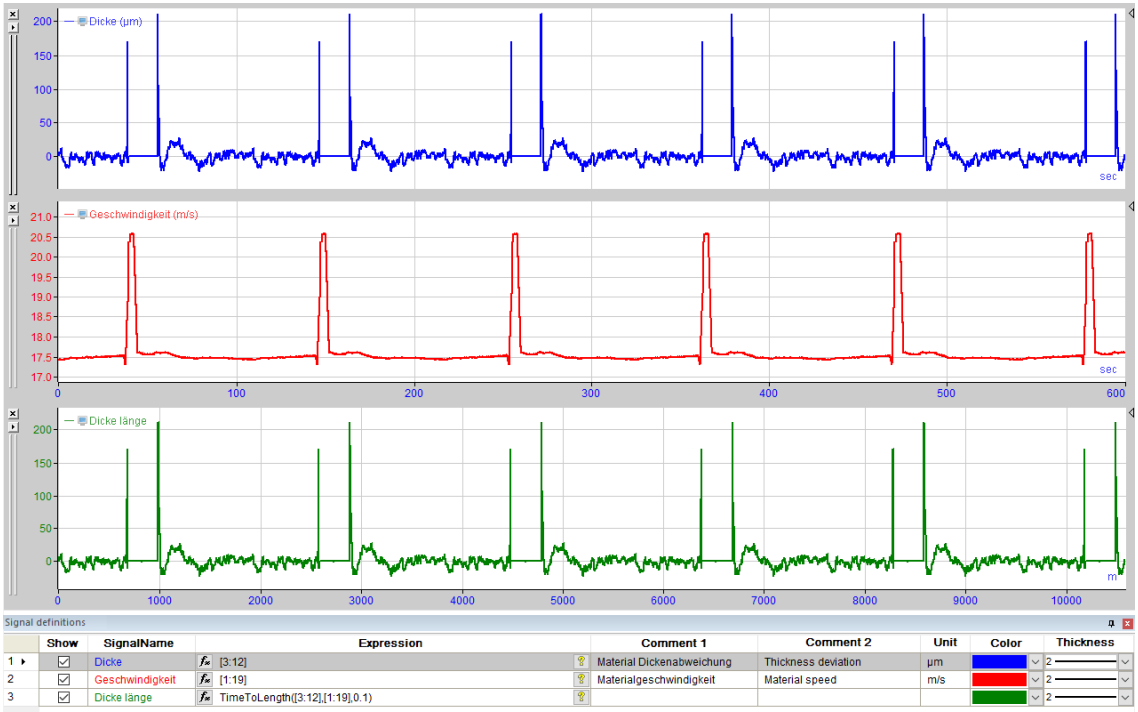
Diese Funktion wandelt das zeitbasierte Signal 'Expression' in ein längenbasiertes um, mit der Geschwindigkeit des Messobjektes 'Speed' als Vortriebsvektor in m/s.

Diese Funktion kann jedes Signal, zu dem ein passendes Geschwindigkeitssignal vorliegt, in eine längenbasierte Darstellung umrechnen. Das heißt, dass nicht nur der Zusammenhang zwischen Messwert und Zeit sondern auch zwischen Messwert und zurückgelegtem Weg dargestellt werden kann.

Am Beispiel eines Stahlbands im Walzwerk kann mit dieser Funktion die Verteilung der Messwerte entlang der Bandlänge ermittelt werden. Wenn prozesstechnisch dafür gesorgt wurde, dass Beginn und Ende der Messung exakt mit Anfang und Ende des Bandes übereinstimmen, dann lässt sich mit dieser Funktion auch die Gesamtlänge des Bandes berechnen. Der größte ermittelte Längenwert wird als Skalenendwert an der X-Achse eingetragen (Autoskalierung).

'Precision' ist eine optionale Angabe in m. Wenn nicht angegeben, dann werden die Punkte für die längenbasierte Kurve entsprechend der Anzahl der Messpunkte des Originalsignals berechnet und im Signalstreifen eingetragen. Wenn eine Genauigkeit angegeben wird, z. B. "0.1", dann wird alle 0,1 m ein neuer längenbasierter Wert berechnet und als Punkt der Kurve eingetragen.

Die nachfolgende Abbildung zeigt Zeit-Länge-Funktion *TimeToLength*.



TimeToLengthL

```
TimeToLengthL('Expression','Length','Precision')
```

Argumente

'Expression'	Signal oder Ausdruck, der auf Länge umgerechnet wird
'Length'	Längensignal
'Precision'	Optionaler Parameter, der die Abtastrate des Ergebnissignals in m festlegt

Beschreibung

Diese Funktion wandelt den zeitbezogenen Messwert 'Expression' in einen längenbezogenen um, mit einem Längenmesswert 'Length' als Position in m.

Die Erläuterungen unter *TimeToLength* gelten entsprechend, nur dass anstelle der Geschwindigkeit ein passender Längenmesswert oder Positionsmesswert verwendet wird.

8 X-Achsen-Operationen

8.1 FillGaps

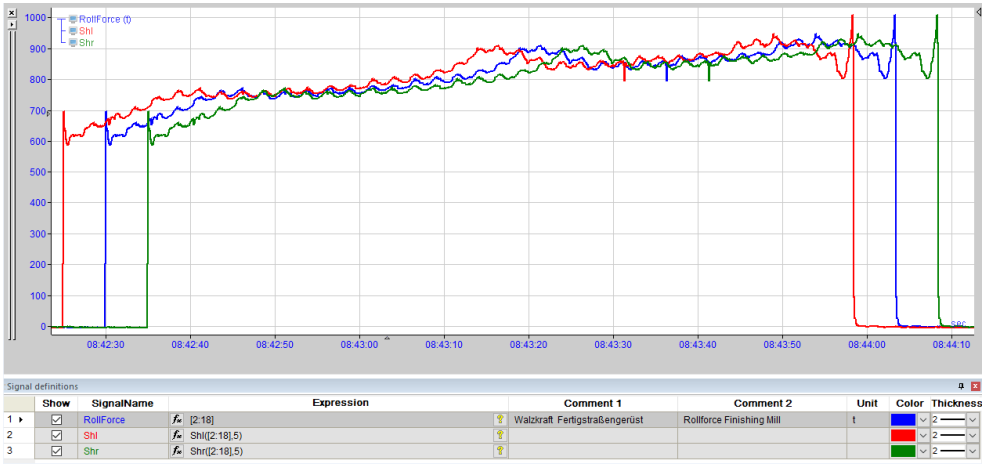
```
FillGaps('Expression')
```

Beschreibung
Mit der Funktion *FillGaps* können Sie Lücken in einem Signal 'Expression' durch linear interpolierte Einträge füllen.

Diese Funktion ist insbesondere nützlich, wenn bei Trendabfragen aus einer Datenbank durch NULL-Einträge Lücken entstehen.

8.2 Verschiebung entlang der X-Achse

Die folgende Abbildung zeigt die Zeit-Länge-Funktionen Shift links (rot) und Shift rechts (grün).



Argumente

'Expression'	Signal oder Ausdruck, der verschoben wird
'Distance'	Verschiebung in Sekunden bzw. in Metern bei längenbezogenen Signalen

Die Funktionen können Sie sowohl für zeitbasierte Signale ('Distance' in Sekunden) als auch für längenbasierte Signale ('Distance' in Metern) verwenden.

Shl

```
Shl('Expression','Distance')
```

Diese Operation liefert als Ergebnis einen Signalverlauf, der um einen Betrag der Länge 'Distance' auf der X-Achse nach links gegenüber dem Originalsignal verschoben ist. Ansonsten bleiben die Messwerte unverändert.

Shr

```
Shr('Expression','Distance')
```

Diese Operation liefert als Ergebnis einen Signalverlauf, der um einen Betrag der Länge 'Distance' auf der X-Achse nach rechts gegenüber dem Originalsignal verschoben ist. Ansonsten bleiben die Messwerte unverändert.

8.3 VarDelay

```
VarDelay('Expression','Delay')
```

Argumente

'Expression'	Signal oder Ausdruck
'Delay'	Verzögerungszeit in Sekunden

Beschreibung
Diese Operation liefert als Ergebnis den Ausdruck 'Expression' um eine Zeitkonstante 'Delay' verzögert zurück. 'Delay' kann auch negativ sein.

8.4 XAlignFft

```
XAlignFft('FixedExpression','AlignExpression','Start','End','MinScale',
'MaxScale','ScaleStep','QualityParam')
```

Argumente

'FixedExpression'	Signal oder Ausdruck, der als Referenz dient
'AlignExpression'	Signal oder Ausdruck, welcher der Referenz angeglichen wird
'Start'	Beginn des X-Achsenabschnitts relativ zum Nullpunkt von 'FixedExpression'
'End'	Ende des X-Achsenabschnitts
'MinScale'	Kleinster x-Skalierungsfaktor, der überprüft wird
'MaxScale'	Größter x-Skalierungsfaktor, der überprüft wird
'ScaleStep'	Schrittweite zur Steuerung von Genauigkeit und Geschwindigkeit des Algorithmus
'QualityParam'	Gütekriterium, um das Abstandmaß zwischen den Zeitreihen zu bilden

Beschreibung
Diese Funktion richtet längenbasierte Signale mit derselben physikalischen Bedeutung, die an verschiedenen Stellen im Prozess gemessen werden, zueinander aus.

Einige Parameter werden nachfolgend genauer beschrieben.

'FixedExpression'
Eine Dickenmessung, die im Laufe des Algorithmus als "fest", also nicht skalierbar oder verschiebbar betrachtet wird. Dies sollte die Dickenmessung sein, die das Profil der anderen Messung enthält. (Im Warmband-Kaltband-Vergleich also das Warmband.)

'AlignExpression'
Auf diese Dickenmessung bezieht sich später das Ergebnis des Alignments. Diese Messung muss also mit den Ergebniswerten skaliert und verschoben werden.

'Start'
Das Intervall von 'Start' bis 'End' gibt den X-Achsenabschnitt an, in dem das Signal 'FixedExpression' verschoben werden darf. Der Nullpunkt ist hierbei der Nullpunkt des Ausdrucks. Auch negative Werte sind erlaubt. Wenn die Messung 'AlignExpression' auf der linken Seite in *ibaAnalyzer* 10 Achseneinheiten im Vergleich zu 'FixedExpression' herausragen darf, so muss Start = -10 sein.

'End'
Dieser Parameter gibt das Ende des beschriebenen Intervalls an. Es empfiehlt sich, dieses Ende in Abhängigkeit von der Länge von 'FixedExpression' zu wählen. Also z. B.
End = XSize([FixedExpression]) oder End = 1.2 * Xsize([FixedExpression]), falls ein Überhang von 20 % erlaubt sein soll.

'ScaleStep'
Dieser Parameter steuert das Verhältnis zwischen Genauigkeit und Geschwindigkeit. Je kleiner der Wert, desto langsamer und verlässlicher arbeitet der Algorithmus. Je höher der Wert, desto mehr wird der Algorithmus durch eine Heuristik beschleunigt. Bei zu hohen Werten für 'ScaleStep' kann dies jedoch zu einem falschen Ergebnis führen. Für ein optimales Ergebnis empfiehlt es sich, die Auflösung der Messdaten zu übergeben. Wenn die Messpunkte z. B. einen Abstand von 10 cm haben, ist 'ScaleStep' = 0.1. Falls die Ergebnisberechnung zu langwierig ist, kann der Wert dann nach oben korrigiert werden.

'QualityParam'
Der Parameter legt das verwendete Gütekriterium der Funktion fest (Abstandsmaß zwischen Zeitreihen). Die Standardeinstellung 0 steht für den Mittleren Quadratischen Fehler (MQF) und 1 bezeichnet den Gleichläufigkeitskoeffizienten (GLK).

8.5 XBase und XOffset

XBase

```
XBase('Expression')
```

Beschreibung

Diese Funktion ermittelt die Aufzeichnungsbasis, also den Abstand zwischen den Samples. Die Basis kann dabei zeitbasiert, längenbasiert oder frequenzbasiert sein.

bzw. längen- oder frequenzbasierte Abstände zwischen den Samples ermitteln. Die Funktion liefert bei einem äquidistant abgetasteten Signal den Abstand zwischen zwei Messpunkten in X-Achseneinheiten.

Bei einem Signal, dessen Samples nicht den gleichen Abstand haben, wird der Abstand in X-Achseneinheiten ausgegeben, der bei einem Resampling auf äquidistante Samples ermittelt werden würde. Standardmäßig ist das der kleinste Abstand zwischen zwei Samples des Signals.

XOffset

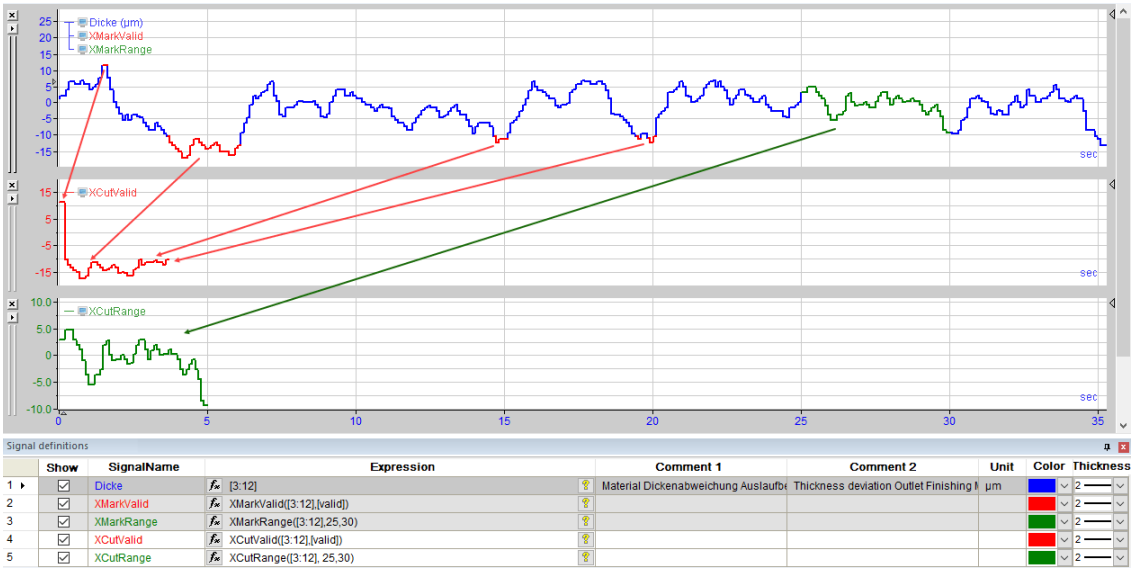
```
XOffset('Expression')
```

Beschreibung

Diese Funktion liefert den zeitlichen Abstand des ersten Messpunktes eines Signals vom Beginn der Messdatei in Sekunden. Das Ergebnis ist negativ, wenn das Signal früher beginnt und positiv, wenn es später beginnt.

Wenn mehrere Messdateien gleichzeitig geöffnet sind und die Option *Messdateien synchronisieren mit Aufnahmezeit* aktiviert ist, dann wird der Offset nicht notwendigerweise mit Bezug auf den Beginn der Messdatei des gewählten Signals ermittelt, sondern auf den Beginn der Messdatei, die den frühesten Startzeitpunkt hat.

8.6 XCutRange und XCutValid



XCutRange

```
XCutRange('Expression','Form','To')
```

Argumente

'Expression'	Signal oder Ausdruck, aus dem ein Teil ausgeschnitten wird
'From'	Anfang des ausgewählten Bereichs in Sekunden bzw. Metern
'To'	Ende des ausgewählten Bereichs in Sekunden bzw. Metern

Beschreibung

Mit dieser Funktion können Sie einen Bereich eines Kurvenzuges ausschneiden. Der herausgeschnittene Teil wird an den Anfang eines eigenen Signalstreifens geschoben. Da die X-Achse (Zeit oder Länge) jedoch unverändert bleibt, ist der korrekte Zeitbezug bzw. Längenbezug der Messwerte nicht mehr gegeben.

Die Funktion können Sie sowohl für zeitbasierte Signale ('From' und 'To' in Sekunden) als auch für längenbasierte Signale ('From' und 'To' in Metern) verwenden.

XCutValid

```
XCutValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, aus dem ein Teil ausgeschnitten wird
'Valid'	Binärsignal, welches den ausgewählten Bereich beschreibt

Beschreibung

Diese Funktion schneidet alle Messpunkte eines Signalverlaufes 'Expression' abhängig von einer Bedingung 'Valid' heraus, wenn diese Bedingung den Wert TRUE liefert. Der Parameter 'Valid' ist ein boolescher Ausdruck. Das kann ein Digitalsignal sein, das Ergebnis einer Vergleichsoperation oder jeder andere binäre Ausdruck. Messpunkte, für die die Bedingung FALSE ist, werden nicht übernommen.

Die herausgeschnittenen Teile werden hintereinander an den Anfang eines neuen Signalstreifens gestellt.

Die Funktion können Sie sowohl für zeitbasierte Signale als auch für längenbasierte Signale verwenden.

8.7 XFirst, XLast und XNow

XFirst

```
XFirst('Expression','Skip=0','SkipInitialEdge=FALSE')
```

Argumente

'Expression'	(boolesches) Eingangssignal oder Ausdruck
'Skip'	Optional zum Überspringen der ersten steigenden Flanken
'SkipInitialEdge'	Parameter, der bestimmt, ob das erste Sample als steigende Flanke gezählt wird, falls es TRUE ist

Beschreibung

Diese Funktion liefert als Ergebnis denjenigen Wert auf der X-Achse (Zeit in s oder Weg in m), für den der Ausdruck 'Expression' zum ersten Mal TRUE ist. 'Expression' ist ein boolescher Ausdruck. Das kann ein Digitalsignal sein, das Ergebnis einer Vergleichsoperation oder jeder andere binäre Ausdruck.

Mit dem Parameter 'Skip' können Sie Flanken überspringen. Um z. B. den Wert der dritten steigenden Flanke zu erhalten, geben Sie 'Skip'=2 an.

Tipp

Um den Wert einer fallenden Flanke zu erhalten, nutzen Sie folgenden Ausdruck:

```
XFirst(NOT('Expression'))
```

XLast

```
XLast('Expression','Skip=0','SkipFinalEdge=FALSE')
```

Argumente

'Expression'	(boolesches) Eingangssignal oder Ausdruck
'Skip'	Optional zum Überspringen der letzten fallenden Flanken
'SkipFinalEdge'	Parameter, der bestimmt, ob das letzte Sample als fallende Flanke gezählt wird, falls es TRUE ist

Beschreibung

Diese Funktion liefert als Ergebnis denjenigen Wert auf der X-Achse (Zeit [s] oder Weg [m]), für den der Ausdruck 'Expression' zum letzten Mal TRUE ist. Dementsprechend muss 'Expression' eine boolesche Größe sein. Das kann ein digitales Eingangssignal sein, das Ergebnis einer Vergleichsoperation oder jeder andere binäre Ausdruck.

Mit dem Parameter 'Skip' können Sie Flanken überspringen. Um z. B. den Wert der drittletzten steigenden Flanke zu erhalten, geben Sie 'Skip'=2 an.

Tipp

Um den Wert einer fallenden Flanke zu erhalten, nutzen Sie folgenden Ausdruck:

```
XLast(NOT('Expression'))
```

XNow

```
XNow()
```

Beschreibung

Diese Funktion gibt die relative Zeit seit dem letzten Start von *ibaAnalyzer* zurück. Wenn keine Messdatei geladen ist, gibt die Funktion die relative Zeit seit der UNIX-Startzeit (01.01.1970; 00:00:00 Uhr) zurück.

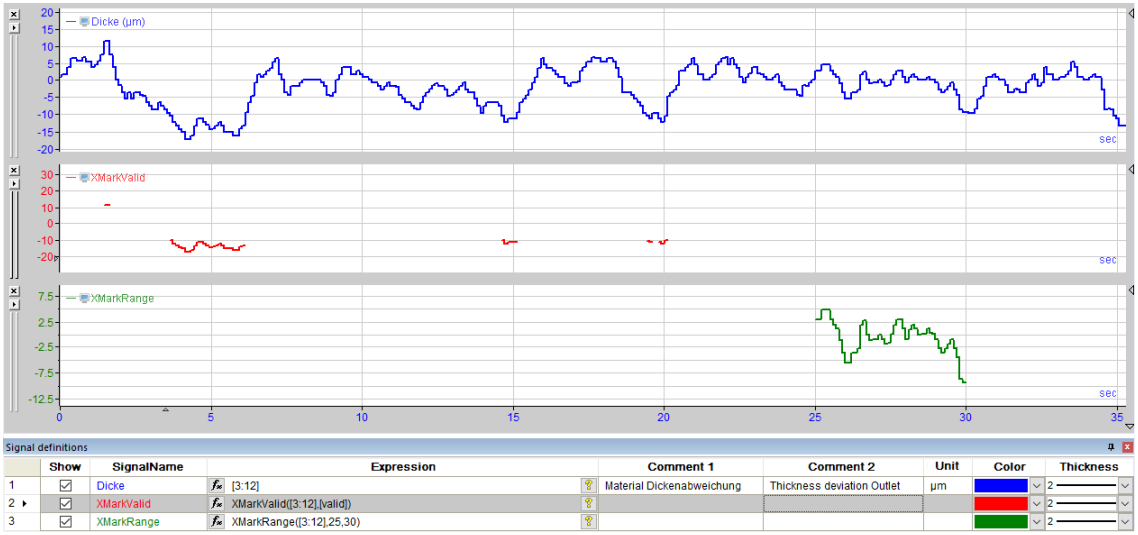
8.8 XMarker1 und XMarker2

```
XMarker1 () und XMarker2 ()
```

Beschreibung

Diese Funktion liefert als Ergebnis die Position des Markers X1 oder X2 auf der X-Achse.

8.9 XMarkRange und XMarkValid



XMarkRange

```
XMarkRange('Expression','From','To')
```

Argumente

'Expression'	Signal oder Ausdruck, aus dem ein Teil ausgewählt wird
'From'	Anfang des ausgewählten Bereichs in Sekunden bzw. Metern
'To'	Ende des ausgewählten Bereichs in Sekunden bzw. Metern

Beschreibung

Mit dieser Funktion können Sie einen Bereich eines Kurvenzuges ähnlich wie mit *XCutRange* ausschneiden. Die Parameter 'From' und 'To' definieren Anfang und Ende des auszuschneidenden Bereiches.

Der herausgeschnittene Teil wird in einem eigenen Signalstreifen dargestellt, bleibt jedoch an der Originalstelle auf der Zeitachse oder Längenachse stehen, und die Messpunkte, die sich nicht in dem angegebenen Bereich befinden, werden verworfen.

Die Funktion können Sie sowohl für zeitbasierte Signale ('From' und 'To' in Sekunden) als auch für längenbasierte Signale ('From' und 'To' in Metern) verwenden.

XMarkValid

```
XMarkValid('Expression','Valid')
```

Argumente

'Expression'	Signal oder Ausdruck, aus dem ein Teil ausgewählt wird
'Valid'	Digitalsignal, das den ausgewählten Bereich beschreibt

Beschreibung

Ähnlich wie mit *XCutValid* schneidet diese Funktion alle Messpunkte eines Signalverlaufes 'Expression' abhängig von einer Bedingung 'Valid' heraus, wenn diese Bedingung den Wert TRUE liefert. Der Parameter 'Valid' ist ein boolescher Ausdruck. Das kann ein Digitalsignal sein, das Ergebnis einer Vergleichsoperation oder jeder andere binäre Ausdruck. Messpunkte, für die die Bedingung FALSE ist, werden nicht übernommen.

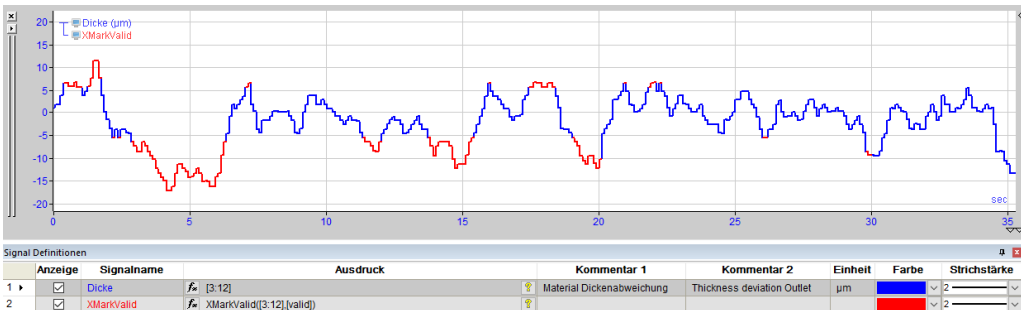
Die herausgeschnittenen Teile werden in einem neuen Signalstreifen dargestellt, wobei sie ihre X-Positionen beibehalten.

Die Funktion können Sie sowohl für zeitbasierte Signale als auch für längenbasierte Signale verwenden.

Tipp

Die Funktion *XMarkValid* ist hilfreich, um z. B. Grenzwertverletzungen farbig in einem Signalverlauf hervorzuheben, indem das Ergebnissignal im selben Streifen und auf derselben Y-Achse wie das Originalsignal dargestellt wird. Durch eine andere Farbgebung lassen sich die Bereiche der Grenzverletzung leicht erkennen.

Beispiel: Werte in Toleranz = blau, Werte außer Toleranz = rot



8.10 XMirror, XStretch und XStretchScale

XMirror

```
XMirror('Expression')
```

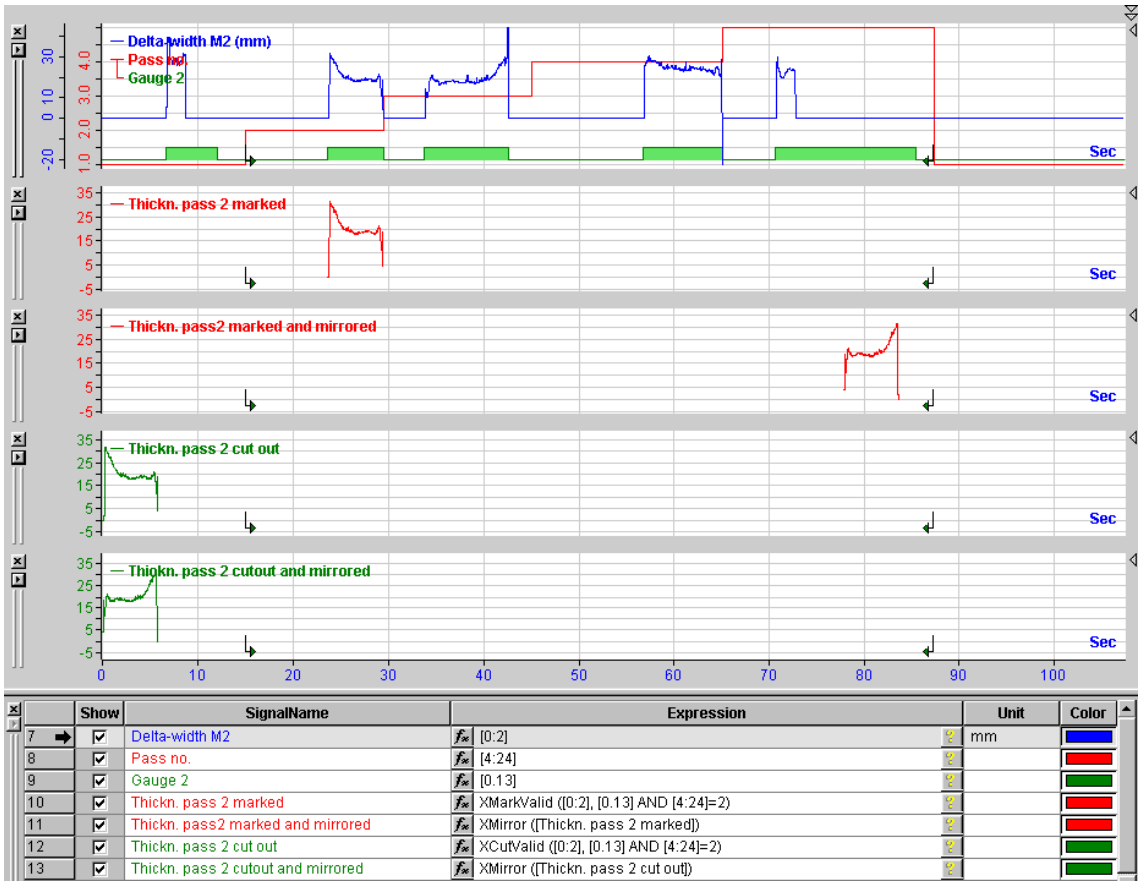
Beschreibung

Mit dieser Funktion können Sie einen kompletten Kurvenverlauf spiegeln (Anfang und Ende vertauschen). Die Spiegelung erfolgt dabei um die vertikale Mittelachse des gesamten Signalverlaufs.

Die Funktion können Sie sowohl für zeitbasierte Signale als auch für längenbasierte Signale verwenden.

Damit lassen sich Messkurven von reversierenden Prozessen (Richtungsumkehr) besser miteinander vergleichen. So können Sie beispielsweise in der Walztechnik Bandkopf und Bandende bei (geraden) Reversierstichen vertauschen, um die Richtungsumkehr grafisch zu neutralisieren. Um mehrere Stiche miteinander vergleichen zu können, müssen Sie die entsprechenden Messwerte jedoch zuvor mit der *XCutValid*-Funktion aus dem Originalsignal herausschneiden, um sie individuell spiegeln und später übereinanderlegen zu können.

Die Abbildung zeigt die unterschiedlichen Ergebnisse der Spiegelung, je nachdem, ob der zu spiegelnde Ausschnitt zuvor mit *XMarkValid* (rot) oder mit *XCutValid* (grün) herausgeschnitten wurde.



	Show	SignalName	Expression	Unit	Color
7	☑	Delta-width M2	[0:2]	mm	
8	☑	Pass no.	[4:24]		
9	☑	Gauge 2	[0.13]		
10	☑	Thickn. pass 2 marked	XMarkValid ([0:2], [0.13] AND [4:24]=2)		
11	☑	Thickn. pass2 marked and mirrored	XMirror ([Thickn. pass 2 marked])		
12	☑	Thickn. pass 2 cut out	XCutValid ([0:2], [0.13] AND [4:24]=2)		
13	☑	Thickn. pass 2 cutout and mirrored	XMirror ([Thickn. pass 2 cut out])		

XStretch

```
XStretch('Expression','Reference expression')
```

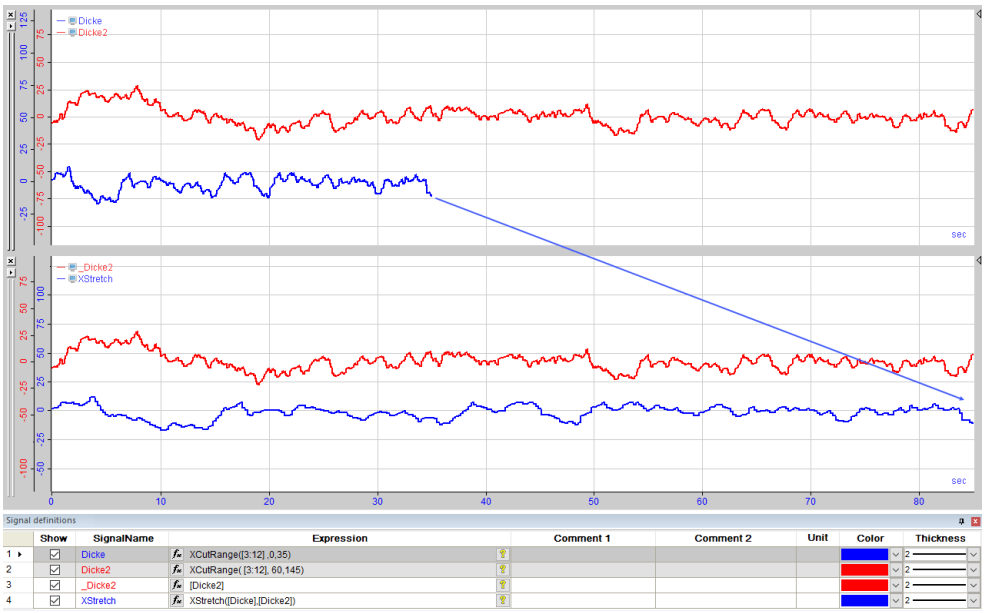
Beschreibung

Mit dieser Funktion können Sie den Signalverlauf auf die gleiche (End-)Länge eines anderen Signals grafisch strecken.

Die Funktion können Sie sowohl für zeitbasierte Signale als auch für längenbasierte Signale verwenden.

So lassen sich z. B. Messwerte eines gewalzten Bandes von der Vorstraße mit denen von der Fertigstraße in Relation setzen oder die einzelnen Stiche eines Reversierwalzwerks miteinander vergleichen.

Die Abbildung zeigt die Walzkraftkurve vom ersten Stich (blaue Kurve) auf die Endlänge gemäß dem neunten Stich gedehnt.

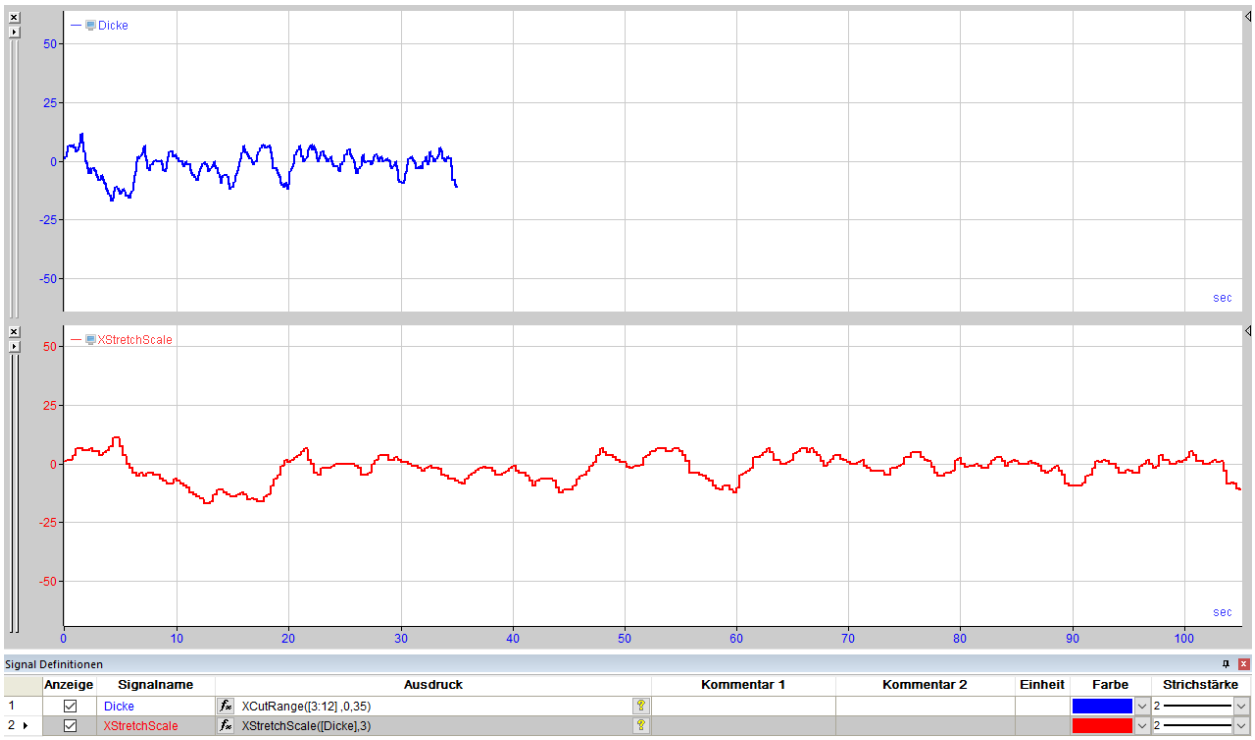


XStretchScale

```
XStretchScale('Expression','Scale')
```

Beschreibung

Mit dieser Funktion können Sie den Signalverlauf um einen festgelegten Faktor strecken. Der Skalierungsfaktor wird auch dann angewendet, wenn der Kurvenzug bereits mit einem Offset versehen ist.



Signal Definitionen

	Anzeige	Signalname	Ausdruck	Kommentar 1	Kommentar 2	Einheit	Farbe	Strichstärke
1	☑	Dicke	XCutRange([3:12] ,0,35)					2
2	☑	XStretchScale	XStretchScale([Dicke],3)					2

8.11 XSize und XSumValid

XSize

```
XSize ('Expression')
```

Beschreibung

Diese Funktion liefert als Ergebnis die Gesamtlänge von 'Expression' in Einheiten der X-Achse (Zeit in s oder Weg in m). Das Ergebnis ist konstant 0, wenn das Eingangssignal ungültig ist.

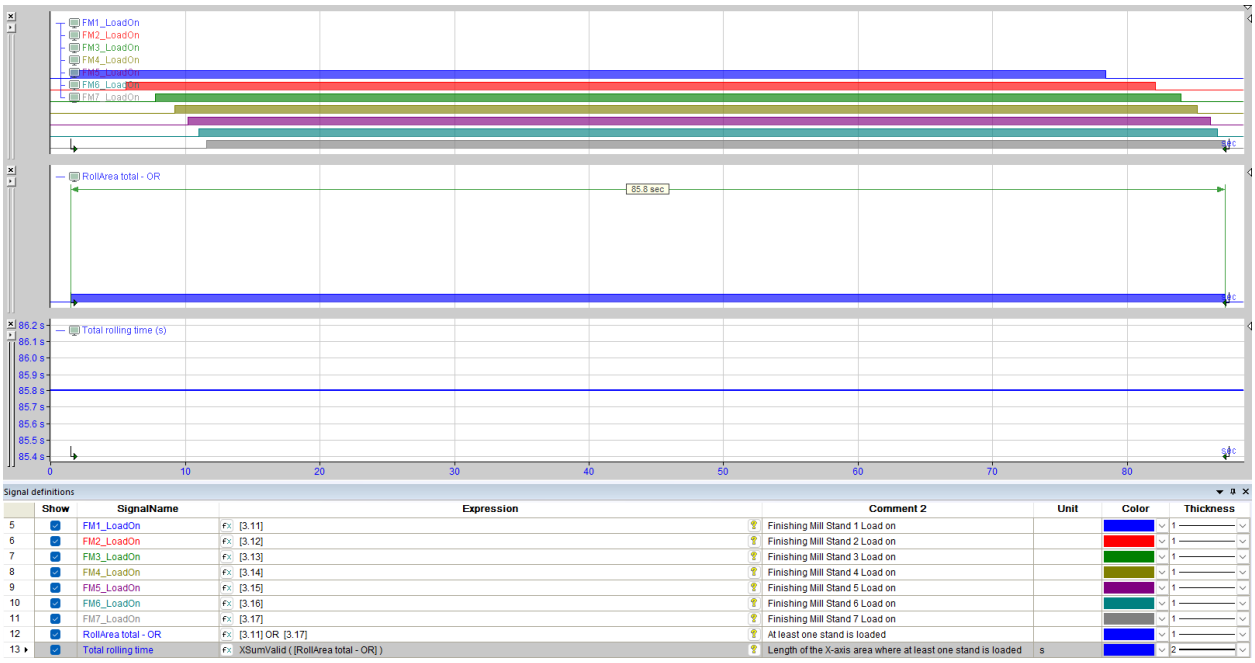
XSumValid

```
XSumValid ('Expression')
```

Beschreibung

Mit dieser Funktion können Sie die Dauer oder die Länge ermitteln, für die die Bedingung 'Expression' TRUE ist. Messpunkte, für die die Bedingung FALSE ist, werden nicht übernommen. 'Expression' ist ein boolescher Ausdruck. Das kann ein Digitalsignal sein, das Ergebnis einer Vergleichsoperation oder jeder andere binäre Ausdruck.

Wenn das Eingangssignal ungültig ist, ist das Ergebnis konstant 0.



	Show	SignalName	Expression	Comment 2	Unit	Color	Thickness
5	☑	FM1_LoadOn	[3.11]	Finishing Mill Stand 1 Load on			1
6	☑	FM2_LoadOn	[3.12]	Finishing Mill Stand 2 Load on			1
7	☑	FM3_LoadOn	[3.13]	Finishing Mill Stand 3 Load on			1
8	☑	FM4_LoadOn	[3.14]	Finishing Mill Stand 4 Load on			1
9	☑	FM5_LoadOn	[3.15]	Finishing Mill Stand 5 Load on			1
10	☑	FM6_LoadOn	[3.16]	Finishing Mill Stand 6 Load on			1
11	☑	FM7_LoadOn	[3.17]	Finishing Mill Stand 7 Load on			1
12	☑	RollArea total - OR	[3.11] OR [3.17]	At least one stand is loaded			1
13	☑	Total rolling time	XSumValid ([RollArea total - OR])	Length of the X-axis area where at least one stand is loaded	s		2

8.12 XValues und YValues

XValues

```
XValues('Expression')
```

Beschreibung

Diese Funktion liefert als Ergebnis die X-Werte aller Messpunkte eines Ausdrucks zurück. Die Funktion können Sie auch für Signale bzw. Ausdrücke verwenden, die nicht zeitbasiert sind, also als Basis Länge (m), Frequenz (Hz) oder inverse Länge (1/m) haben.

Bei einem normalen zeit- oder wegkontinuierlichen Signalverlauf ist das Ergebnis eine steigende Gerade, mit den Basiseinheiten (Zeit- oder Wegwerte) auf der Y-Achse in s oder m. Die Funktion arbeitet auch mit nicht-äquidistanten Messwerten.

YValues

```
YValues('Expression','TimeBase'=1)
```

Argumente

'Expression'	Ausdruck oder Signal
'TimeBase'	Zeitbasis des Ergebnissignals

Beschreibung

Diese Funktion liefert als Ergebnis die Y-Werte aller Messwerte eines Ausdrucks zurück. Unabhängig davon, ob das Eingangssignal äquidistant gesampelt ist oder nicht, ist das Ergebnis ein äquidistantes Signal mit Zeitbasis 'TimeBase'.

Die Angabe der Zeitbasis ist optional, und als Standardwert wird 'TimeBase'=1 verwendet.

8.13 XY

```
XY('Expression1','Expression2','Precision')
```

Argumente

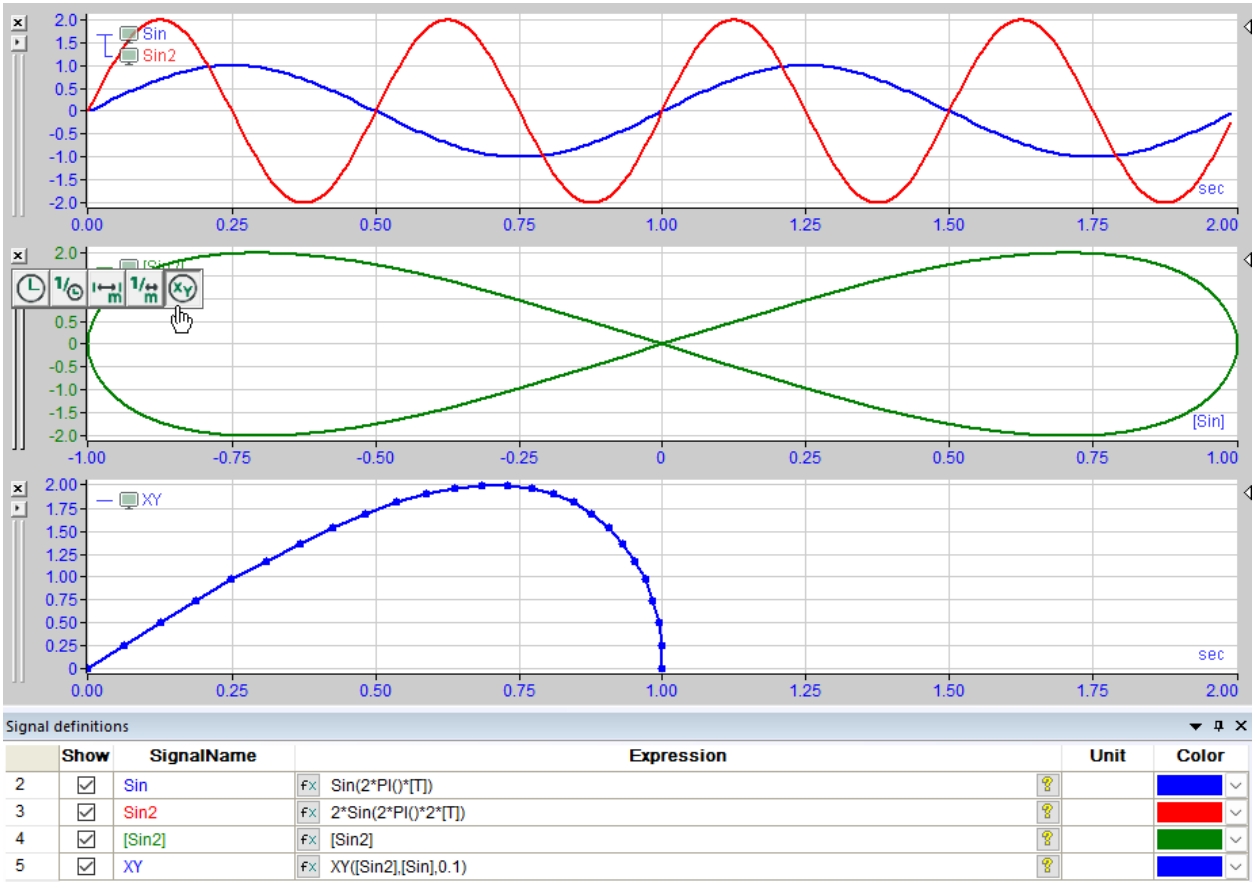
'Expression1'	Signal oder Ausdruck, der die Y-Werte des neuen Signals enthält
'Expression2'	Signal oder Ausdruck, der die X-Werte des neuen Signals enthält
'Precision'	Optionaler Parameter, der die Abtastrate des Ergebnissignals festlegt

Beschreibung

Sie können diese Funktion verwenden, wenn das Resultat aus der X-Y-Darstellung für weitere Operationen genutzt werden soll. Nach erfolgter Auswahl werden der Funktion die Signale der X-Achse und Y-Achse zugewiesen.

Beachten Sie, dass in der resultierenden Funktion die Abstände zwischen den Signalpunkten nicht gleich den Abständen der Ursprungssignale sind. Auch der Abstand zwischen den Signalpunkten selbst ist unterschiedlich. Mit dem Parameter 'Precision' können Sie einen festen Abstand zwischen den Signalpunkten einstellen. Wenn Sie kein Parameter angeben, wird für alle Folgeoperationen der kürzeste Abstand der Signalpunkte als fester Wert verwendet.

Sie können die X-Y-Darstellung auch über den X-Achsenmodus bestimmen. Die Abbildung zeigt die Ergebnisse des X-Achsenmodus und der *XY*-Funktion im Vergleich.



Signal definitions

	Show	SignalName	Expression	Unit	Color
2	☑	Sin	Sin(2*PI()*[T])		
3	☑	Sin2	2*Sin(2*PI()*2*[T])		
4	☑	[Sin2]	[Sin2]		
5	☑	XY	XY([Sin2],[Sin],0.1)		

Weitere Informationen siehe *ibaAnalyzer*-Handbuch Teil 2, Kapitel *X-Achsen-Modus X – Y*.

9 Vektor-Operationen

Vektor-Operationen erweitern die Analysemöglichkeiten für zweidimensionale Signale.

Vektoren oder auch Arrays können Sie auf unterschiedliche Arten bilden:

- Gruppieren Sie mehrere Signale in *ibaPDA* und kennzeichnen Sie die Gruppe als Vektor.
- Stellen Sie mehrere Signale in den logischen Ausdrücken in *ibaAnalyzer* zusammen.
- Verschiedene Berechnungsfunktionen liefern Vektoren als Ergebnis, z. B. FFT-Funktionen.

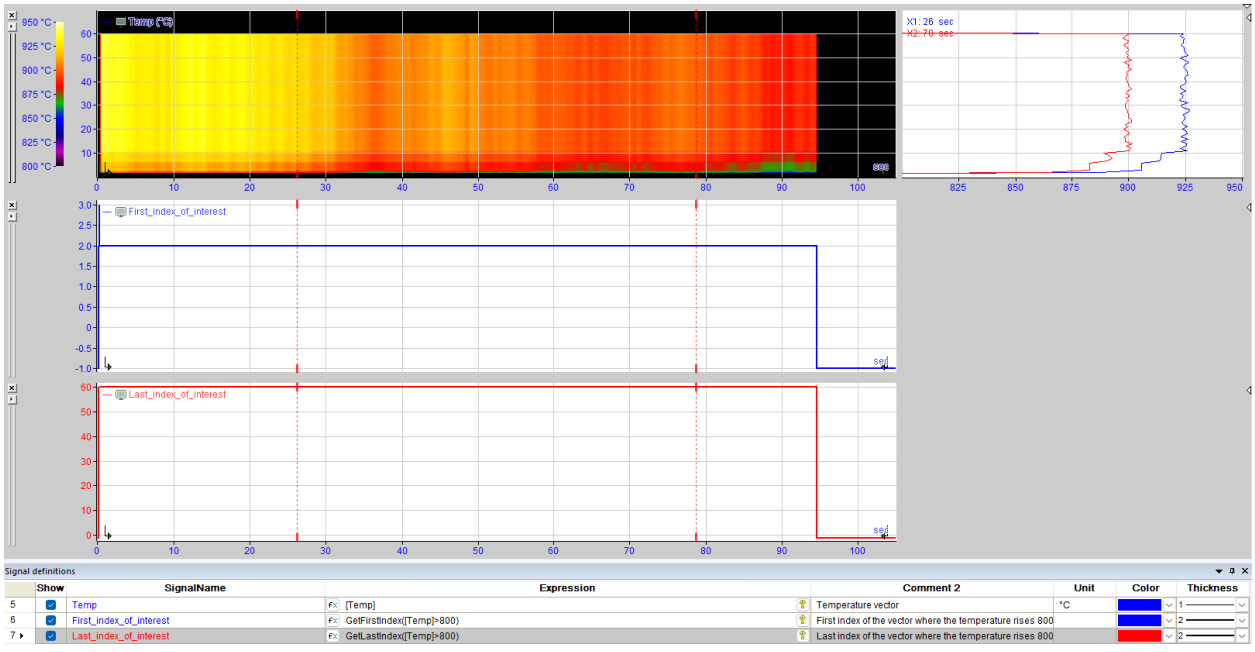
Vektoren können Sie in *ibaAnalyzer* mit der 2D-Draufsicht und 3D-Ansicht darstellen. Mit den Vektor-Operationen im Ausdruckseditor können Sie Vektordaten für weiterführende Berechnungen verwenden.

Ausführliche Hinweise zu diesen Darstellungen und ihren Einstellungen finden Sie im *ibaAnalyzer*-Handbuch Teil 2, *Darstellungsarten*.

9.1 GetFirstIndex und GetLastIndex

`GetFirstIndex('Condition') und GetLastIndex('Condition')`

Beschreibung
Diese Funktionen liefern den Index des ersten bzw. letzten Signals im Vektor zurück, für den die Bedingung 'Condition' TRUE ist. Der Vektor selbst muss dabei ein Operand von 'Condition' sein. Wenn 'Condition' FALSE für alle Signale des Vektors ergibt, dann liefert die Funktion -1 als Ergebnis.



9.2 GetRows

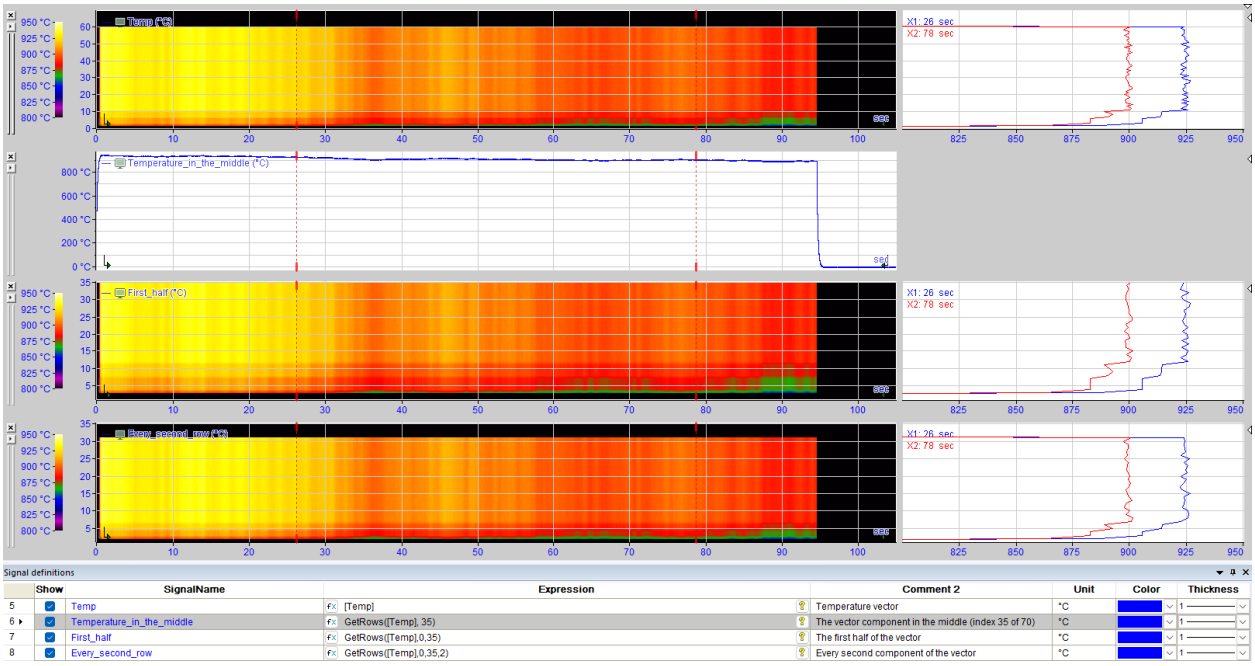
```
GetRows('Vector','StartIndex','Counter','Step')
```

Argumente

'Vector'	Vektor, aus dem einzelne Einträge ausgelesen werden
'StartIndex'	Erster Index, aus dem Einträge ausgelesen werden
'Counter'	Anzahl der Einträge
'Step'	Schrittgröße

Beschreibung

Diese Funktion liest Wertereihen aus einem Vektor (Array) aus. Dabei wird ausgehend von einem 'StartIndex' (der kleinste mögliche Index ist 0) in Schritten der Größe 'Step' eine Gesamtzahl 'Counter' an Einträgen ausgelesen.



	Show	SignalName	Expression	Comment 2	Unit	Color	Thickness
5	☑	Temp	[Temp]	Temperature vector	°C		1
6	☑	Temperature_in_the_middle	GetRows([Temp], 35)	The vector component in the middle (index 35 of 70)	°C		1
7	☑	First_half	GetRows([Temp],0,35)	The first half of the vector	°C		1
8	☑	Every_second_row	GetRows([Temp],0,35,2)	Every second component of the vector	°C		1

9.3 GetZoneCenters

```
GetZoneCenters('Vector')
```

Beschreibung

Diese Funktion ermittelt die Position der Zonenmitte auf der Y-Achse für jede Zone des Vektors. Einziges Argument der Funktion ist der Vektor. Das Ergebnis ist wieder ein Vektor mit einer Anzahl Werte entsprechend der Zonenanzahl.

Beispiel

Die Funktion *GetZoneCenters* ist besonders hilfreich, wenn sie auf das Ergebnis einer *FftInTime*-Funktion angewendet wird. Die *FftInTime*-Funktion liefert als Ergebnis einen Vektor mit *n* "Zonen", die den Frequenzbändern (Bins) entsprechen. Mit der Funktion *GetZoneCenters* können die Mittenfrequenzen der einzelnen Bänder des Spektrums und somit der Frequenzvektor bestimmt werden. Damit ist es dann möglich in der Frequenzdomäne zu differenzieren oder zu integrieren, indem die Ergebnisse der *FftInTime*- und der *GetZoneCenters*-Funktion entsprechend multipliziert oder dividiert werden.

9.4 GetZoneOffset

```
GetZoneOffset('Vector')
```

Beschreibung
Diese Funktion ermittelt den Offset der ersten Zone, d. h. die Position der Zonenmitte der ersten Zone, bezogen auf die Nulllinie der Y-Achse. Einziges Argument der Funktion ist der Vektor. Das Ergebnis ist ein konstanter Wert.

9.5 GetZoneWidths

```
GetZoneWidths('Vector')
```

Beschreibung
Diese Funktion ermittelt die Breite jeder Zone des Vektors in Einheiten der Y-Achse. Einziges Argument der Funktion ist der Vektor. Das Ergebnis ist wieder ein Vektor mit einer Anzahl Werte entsprechend der Zonenanzahl.

9.6 MakeVector

```
MakeVector(r_0,r_1,…,r_n)
```

Beschreibung
Diese Funktion erzeugt einen Vektor mit den Wertereihen *r_0* bis r_n. Die Argumente können konstante Werte, Signale oder Ausdrücke sein. Das ist vergleichbar mit der Erzeugung eines Vektors im Dialog der *Logischen Ausdrücke*.

Beispiel
Die Funktion *MakeVector* dient hauptsächlich dazu, dass Makros mehrdimensionale Signale als Ergebnis ausgeben können. Im Makro-Editor können die Teilergebnisse verschiedener Berechnungen innerhalb des Makros als Zwischenwerte deklariert werden. Als finales Ergebnis des Makros kann dann ein Vektor definiert werden, dessen Argumente die Zwischenwerte sind. Der Vektor wird quasi als Behälter für Makroergebnisse genutzt, um die Makroschnittstelle zu vereinfachen.

9.7 SetZoneWidths

```
SetZoneWidths('Vector','Widths','Offset')
```

Argumente

'Vector'	Vektor mit den (Mess-)Werten des Ergebnisvektors
'Widths'	Vektor, der als Werte die Zonenbreiten enthält
'Offset'	Abstand der Zonenmitte der ersten Zone von der Nulllinie

Beschreibung
Diese Funktion erzeugt einen Vektor mit vorgegebenen Zonenbreiten. Dabei werden die Werte des Ergebnisvektors aus einem Vektor 'Vector' und die Zonenbreiten aus einem Vektor 'Widths' entnommen. Da der Vektor mit den Zonenbreiten seinerseits Ausdrücke als Argumente verwenden kann, lassen sich mit dieser Funktion Vektoren mit verschiedenen Zonenbreiten in Abhängigkeit der geladenen Daten erzeugen. Die Ausdrücke zur Definition der Zonenbreiten sollten dabei konstant über die Zeit sein und sich für einmal geladene Daten nicht mehr ändern. Wenn dies nicht der Fall ist, werden die Breitenwerte über den Gesamtzeitraum gemittelt.

Beispiel
Die Funktion *SetZoneWidths (MakeVector(1,2,3,2,1),MakeVector(2,4,10,4,2), -10)* erzeugt den gleichen Vektor wie den, der mit den logischen Ausdrücken erzeugt wurde.

Weitere Informationen siehe *ibaAnalyzer*-Handbuch Teil 2, Kapitel *Logische Ausdrücke*.

9.8 Traverse und TraverseW

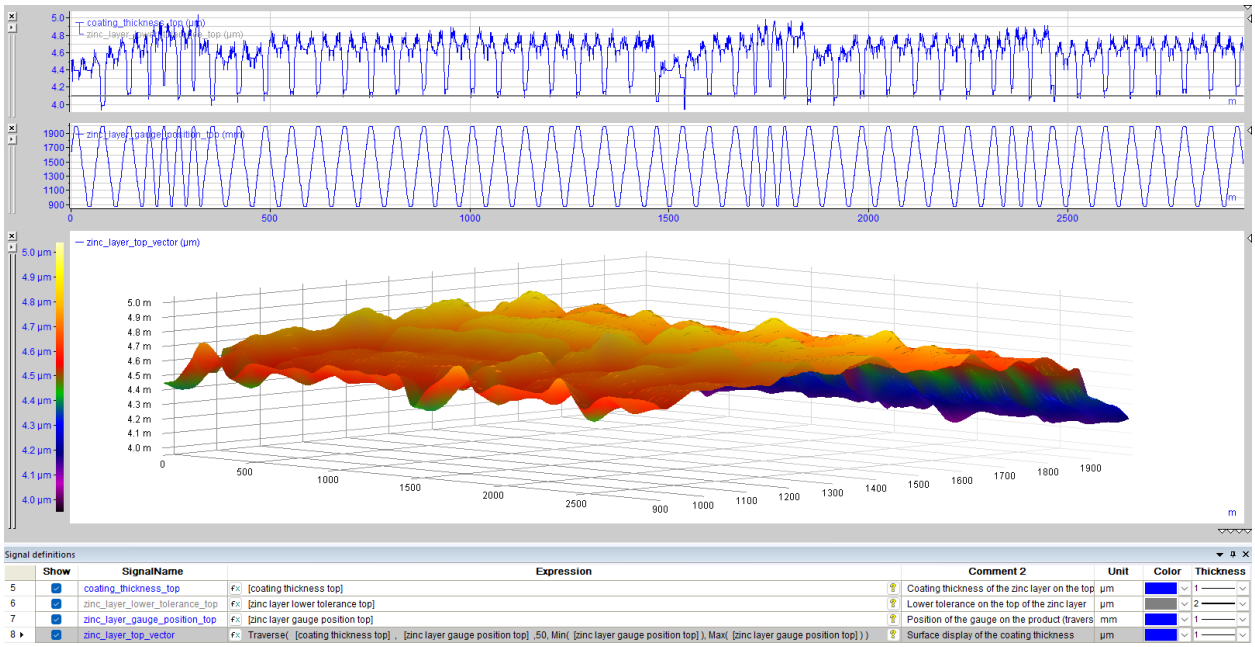
Traverse

```
Traverse('Signal','Position','N'=40,'Min','Max','Avg'=TRUE)
```

Argumente

'Signal'	Von einem traversierenden Messgerät gemessenes Signal
'Position'	Position des traversierenden Messgeräts entlang des Querprofils
'N'	Anzahl der Zonen des resultierenden Vektors
'Min'	Optionaler Threshold für das Minimum des abzubildenden Bereichs
'Max'	Optionaler Threshold für das Maximum des abzubildenden Bereichs
'Avg'	Optionaler, binärer Parameter, um mehrere Samples innerhalb eines Zonendurchgangs zu mitteln; standardmäßig wird der Mittelwert gebildet

Beschreibung
Diese Funktion wandelt Signale, die von einem traversierenden Messgerät stammen, in einen Vektor zur zweidimensionalen Darstellung um.



TraverseW

```
TraverseW('Signal','Position','Widths','Offset'=0,'Avg'=TRUE)
```

Argumente

'Signal'	Von einem traversierenden Messgerät gemessenes Signal
'Position'	Position des traversierenden Messgeräts entlang des Querprofils
'Widths'	Breite der resultierenden Zonen
'Offset'	Optionaler Versatz der ersten Zone
'Avg'	Optionaler, binärer Parameter, um mehrere Samples innerhalb eines Zonen-durchgangs zu mitteln; standardmäßig wird der Mittelwert gebildet

Beschreibung

Diese Funktion funktioniert ähnlich wie *Traverse*, mit dem Unterschied, dass die Dimensionen des resultierenden Vektors direkt über die Zonenbreiten 'Widths' und einen optionalen Offset-Parameter gesetzt werden.

Der Parameter 'Widths' muss ein Vektor sein, der die Breite für jede Zone enthält.

9.9 VectorAvg

```
VectorAvg('Vector')
```

Beschreibung

Diese Funktion berechnet für jedes Sample den Mittelwert des Querprofils, d. h. den Mittelwert aller Vektorspuren pro Zeitpunkt bzw. pro X-Achsenposition. Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf des Querprofil-Mittelwerts über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

9.10 VectorKurtosis

```
VectorKurtosis('Vector')
```

Beschreibung
Diese Funktion berechnet für jedes Sample die Kurtosis (Wölbung) des Querprofils. Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf der Querprofil-Wölbung über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

Hinweis

Es müssen mindestens 4 Signalpunkte vorhanden sein, d. h. der Vektor muss mindestens 4 Eingänge haben.

9.11 VectorMarkRange

```
VectorMarkRange('Vector','PositionFrom','PositionTo')
```

Argumente

'Vector'	Vektor mit Eingangssignalen
'PositionFrom'	Anfang des auszuwählenden Bereichs
'PositionTo'	Ende des auszuwählenden Bereichs

Beschreibung
Diese Funktion liefert als Ergebnis einen Teilvektor von 'Vector' mit einer Zonenbreite von 'PositionFrom' (untere Kante) bis 'PositionTo' (obere Kante).

Die Angabe der Positionen muss in Einheiten der Y-Achse erfolgen. Die Positionen können sowohl feste Werte als auch Signale oder Ausdrücke und somit abhängig von den geladenen Daten sein.

Die Ausdrücke zur Definition der Positionen sollten dabei konstant über die Zeit sein und sich für einmal geladene Daten nicht mehr ändern. Wenn dies nicht der Fall ist, werden die Positionswerte über den Gesamtzeitraum gemittelt.

9.12 VectorMin und VectorMax

VectorMax

```
VectorMax('Vector')
```

Beschreibung

Diese Funktion berechnet für jedes Sample das Maximum des Querprofils, d. h. den Maximalwert aller Vektorspuren pro Zeitpunkt bzw. pro X-Achsenposition. Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf des Querprofil-Maximums über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

VectorMin

```
VectorMin('Vector')
```

Beschreibung

Diese Funktion berechnet für jedes Sample das Minimum des Querprofils, d. h. den Minimalwert aller Vektorspuren pro Zeitpunkt bzw. pro X-Achsenposition. Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf des Querprofil-Minimums über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

9.13 VectorMedian

```
VectorMedian('Expression')
```

Beschreibung

Diese Funktion berechnet für jedes Sample den Median des Querprofils, d. h. den Median aller Vektorspuren pro Zeitpunkt oder pro X-Achsenposition. Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf des Querprofil-Medians über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

9.14 VectorPercentile

```
VectorPercentile('Vector','Percentile'=0.5)
```

Argumente

'Vector'	Vektor mit Eingangssignalen
'Percentile'	Perzentile, zwischen 0 und 1

Beschreibung

Diese Funktion berechnet für jedes Sample die Perzentile des Querprofils.

Das zweite Argument neben dem 'Vector' ist die Angabe der 'Percentile', die berechnet werden soll. Standardwert ist 0.5 (Median). Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf der Querprofil-Perzentilen über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

9.15 VectorPolynomial und VectorLSQPolyCoef

VectorPolynomial

```
VectorPolynomial('Coefs','Vector','PolynomialType'=0)
```

Argumente

'Coefs'	Koeffizienten des Interpolationspolynoms; berechnet mit *VectorLSQPolyCoef*
'Vector'	Stützstellen zur Auswertung des Interpolationspolynoms
'PolynomialType'	Definiert die Basispolynome, die verwendet werden 0 = Lagrange (Standard) 1 = Chebyshev I 2 = Chebyshev II 3 = Legendre

Beschreibung

Diese Funktion können Sie nutzen, um das Interpolationspolynom darzustellen, welches durch die Koeffizienten 'Coefs' als Ergebnis der Funktion *VectorLSQPolyCoef* beschrieben wird.

Die Stützstellen zur Auswertung des Polynoms werden durch die Abtastpunkte des Vektors 'Vector' festgelegt. Falls Zonen-Offset oder Zonenbreite spezifiziert sind, werden diese auch benutzt, ansonsten werden die Indizes als Y-Werte herangezogen.

Den optionalen Parameter 'PolynomialType' können Sie nutzen, um verschiedene Basispolynome zu verwenden. Unterstützt werden die Typen Lagrange (Standard), Chebyshev I, Chebyshev II und Legendre.

Hinweis

Beachten Sie, dass die Einträge von 'Vector' hier nicht relevant sind.

VectorLSQPolyCoef

```
VectorLSQPolyCoef('Vector','Degree','PolynomialType'=0)
```

Argumente

'Vector'	Vektor, für dessen Einträge die Least-Squares-Approximationspolynome berechnet werden
'Degree'	Grad des Interpolationspolynoms
'PolynomialType'	Definiert die Basispolynome, die verwendet werden 0 = Lagrange (Standard) 1 = Chebyshev I 2 = Chebyshev II 3 = Legendre

Beschreibung
Diese Funktion ist die Erweiterung der Funktion *LSQPolyCoef* auf Vektoren. Dabei werden die Koeffizienten eines Interpolationspolynoms vom Grad 'Degree' mit Hilfe der Methode der kleinsten Quadrate für jedes Querprofil berechnet. Als Basis werden hierzu die Indizes des Vektors herangezogen, es sei denn ein Zonen-Offset oder die Zonenbreite wurden bei der Erstellung des Vektors gesetzt. In diesem Fall werden die entsprechenden Werte als Basis verwendet.

Den optionalen Parameter 'PolynomialType' können Sie nutzen, um verschiedene Basispolynome zu verwenden. Unterstützt werden die Typen Lagrange (Standard), Chebyshev I, Chebyshev II und Legendre.

9.16 VectorSkewness

```
VectorSkewness('Vector')
```

Beschreibung
Diese Funktion berechnet für jedes Sample die Skewness (Schiefe) des Querprofils. Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf der Querprofil-Schiefe über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

Hinweis

Es müssen mindestens 4 Signalpunkte vorhanden sein, d. h. der Vektor muss mindestens 4 Eingänge haben.

9.17 VectorStdDev

```
VectorStdDev('Vector')
```

Beschreibung
Diese Funktion berechnet für jedes Sample die Standardabweichung des Querprofils. Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf der Standardabweichung über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

9.18 VectorSum

```
VectorSum('Vector')
```

Beschreibung
Diese Funktion berechnet für jedes Sample die Summe aller Werte des Querprofils. Als Ergebnis liefert die Funktion ein eindimensionales Signal, das den Verlauf der Wertesumme im Querprofil über die Zeit oder Länge des Vektorsignals mit gleicher Anzahl Samples zeigt.

Beispiel
Wenn Sie den *VectorSum*-Ausdruck durch die Anzahl der Vektorspuren dividieren, dann erhalten Sie das gleiche Ergebnis wie mit der Funktion *VectorAvg*.

9.19 VectorToSignal und SignalToVector

VectorToSignal

```
VectorToSignal('Vector','XBase')
```

Argumente

'Vector'	Vektor mit (möglichst konstanten) Eingangssignalen
'XBase'	Abtastrate des Ergebnissignals

Beschreibung

Diese Funktion erzeugt entlang des Querprofils aus den Elementen eines Vektors ein eindimensionales Signal. Jedes Sample des resultierenden Signals entspricht einem Element des Vektors. Das Ergebnis entspricht dem Querprofil. Wenn die Einträge des Vektors nicht konstant sind, wird das Querprofil der Mittelwerte der einzelnen Signale gebildet.

Der Parameter 'XBase' ist optional. Wird 'XBase' nicht angegeben, dann werden die Zonenbreiten und der Offset des Vektors verwendet. Das resultierende Signal kann somit auch nicht-äquidistante Samples erhalten.

Beispiel

In Verbindung mit den Funktionen *YatX* und der Markerposition kann die Funktion *VectorToSignal* dazu genutzt werden, an einer beliebigen Stelle im Vektor das Querprofil auszugeben:

VectorToSignal (YatX([Vektor],XMarker1()))

SignalToVector

```
SignalToVector('Signal')
```

Beschreibung

Im Gegensatz zur *VectorToSignal* Funktion erzeugt die Funktion *SignalToVector* einen Vektor mit konstanten Einträgen aus 'Signal'. Die Zonenbreite und Offset wird dabei durch die Abtastrate des Eingangssignals bestimmt. Beachten Sie, dass diese Funktion im Gegensatz zu *VectorToSignal* kein optionales Argument besitzt, um die Zonenbreiten zu bestimmen. Dazu kann die Funktion *SetZoneWidth* verwendet werden.

Hinweis

Das Eingangssignal sollte weniger als 1000 Samples haben, ansonsten wird das Signal nicht ausgewertet und als zu komplex markiert.

10 Elektrische Funktionen

10.1 RMS und Eff

```
RMS('Expression','Frequency'=50) und Eff('Expression','Frequency'=50)
```

Argumente

'Expression'	Signal oder Ausdruck, für den der Effektivwert berechnet wird
'Frequency'	Grundfrequenz

Beschreibung
Diese Funktion berechnet den so genannten Root-Mean-Square-Wert bzw. den Effektivwert von 'Expression' mit der Grundfrequenz von 'Frequency':

$$E_{eff} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} e^2(n)}$$

e(n): Messpunkt n von Signal e ('Expression')

N: Anzahl der Messwerte pro Periode

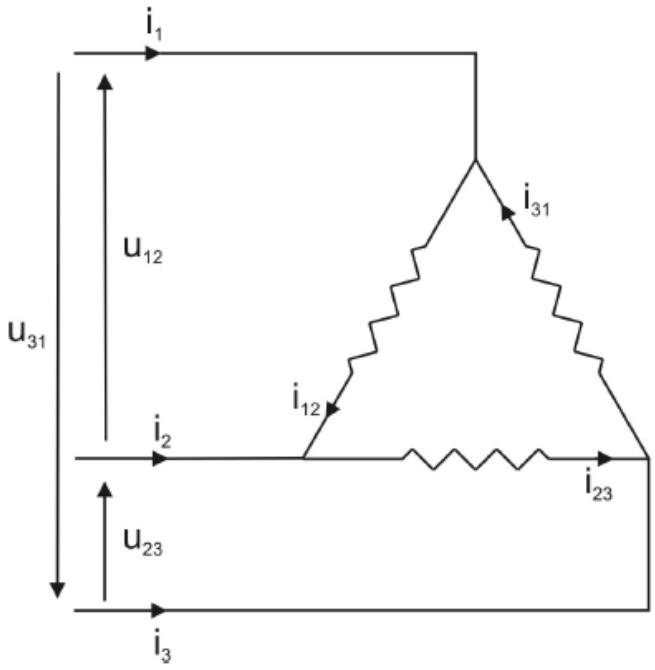
Beispiel
Für einen Wechselspannungsverlauf mit einer Frequenz von 0,1 kHz, der von einer zweiten Wechselspannung mit 0,5 kHz überlagert wird, können Sie den Effektivwert der Spannung für beide Frequenzen ermitteln, indem Sie die Funktion *Eff* mit zweitem Argument 0.1 bzw. 0.5 anwenden.

Hinweis

Zwischen den Funktionen *RMS* und *Eff* gibt es keinen Unterschied. Beide Funktionen werden aus Kompatibilitätsgründen von *ibaAnalyzer* unterstützt.

10.2 Dreieck-Funktionen

Die Dreieck-Funktionen benutzen die Anschlussspannungen und Anschlussströme in einem Dreiecknetz, um die Leistungswerte zu berechnen.



Argumente

'u12', 'u13', 'u23'	Anschlussspannung (gleich den Phasenspannungen)
'i1', 'i2', 'i3'	Anschlussströme
'Frequency'	Grundfrequenz

Hinweis

Die Funktionen werden üblicherweise bei Dreiecknetzen eingesetzt, aber sie können auch für andere Netze verwendet werden, in denen Anschlussspannungen und -ströme gemessen werden können.

DeltaCollectiveUeff

```
DeltaCollectiveUeff('u12','u13','u23','Frequency'=50)
```

Berechnet die gemeinsame Effektivspannung in einem Dreiecknetz:

$$U_{eff} = \sqrt{\frac{1}{3}(U_{12,eff}^2 + U_{23,eff}^2 + U_{31,eff}^2)}$$

$U_{xy,eff}$: der Effektivwert der Anschlussspannung U_{xy}

DeltaCollectiveIeff

```
DeltaCollectiveIeff('i1','i2','i3','Frequency'=50)
```

Berechnet den gemeinsamen Effektivstrom in einem Dreiecknetz:

$$I_{eff} = \sqrt{\sum_{x=1}^{3} I_{x,eff}^2}$$

$I_{x,eff}$: der Effektivwert des Anschlussstroms i_x

DeltaActiveP

```
DeltaActiveP('u13','u23','i1','i2','Frequency'=50)
```

Berechnet die Wirkleistung in einem Dreiecknetz:

$$P = \frac{1}{N} \sum_{n=1}^{N} \left[u_{23}(n) i_2(n) + u_{13}(n) i_1(n) \right]$$

N : die Anzahl der Messwerte pro Periode

u_{xy} : die Spannung zwischen Anschluss x und y (u13 = -u31)

i_x : der Strom durch Anschluss x

DeltaApparentP

```
DeltaApparentP('u12','u13','u23','i1','i2','i3','Frequency'=50)
```

Berechnet die Scheinleistung in einem Dreiecknetz:

$$S = U_{eff} I_{eff}$$

U_{eff} : die gemeinsame Effektivspannung

I_{eff} : der gemeinsame Effektivstrom

DeltaReactiveP

```
DeltaReactiveP('u12','u13','u23','i1','i2','i3','Frequency'=50)
```

Berechnet die Blindleistung in einem Dreiecknetz:

$$Q = \sqrt{S^2 - P^2}$$

S : Scheinleistung

P : Wirkleistung

DeltaReactivePS

```
DeltaReactivePS ('u13','u23','i1','i2','Frequency'=50)
```

Berechnet die vorzeichenbehaftete Blindleistung Q_S im Dreiecknetz.

DeltaActivePFactor

```
DeltaActivePFactor('u12','u13','u23','i1','i2','i3','Frequency'=50)
```

Berechnet den Wirkleistungsfaktor in einem Dreiecknetz:

$$\cos\varphi = \frac{P}{S}$$

S : Scheinleistung

P : Wirkleistung

DeltaReactivePFactor

```
DeltaReactivePFactor('u12','u13','u23','i1','i2','i3','Frequency'=50)
```

Berechnet den Blindleistungsfaktor in einem Dreiecknetz:

$$\tan\varphi = \frac{Q}{P}$$

Q : Blindleistung

P : Wirkleistung

DeltaReactivePFactorS

```
DeltaReactivePFactorS('u13','u23','i1','i2','Frequency'=50)
```

Berechnet den vorzeichenbehafteten Blindleistungsfaktor in einem Dreiecknetz:

$$\tan\varphi = \frac{Q_S}{P}$$

Q_S : vorzeichenbehaftete Blindleistung

P : Wirkleistung

10.3 Harmonische Funktionen

HarmEff

```
HarmEff('u','Nharm','Frequency'=50)
```

Berechnet den Effektivwert der 'Nharm'-ten Harmonischen des Signals 'u':

$$u_{Real,k} = \frac{2}{N}\sum_{n=0}^{N-1} u(n)\cos\frac{2\pi kn}{N}$$

$$u_{Imag,k} = \frac{2}{N}\sum_{n=0}^{N-1} u(n)\sin\frac{2\pi kn}{N}$$

$$U_k = \frac{\sqrt{u_{Real,k}^2 + u_{Imag,k}^2}}{\sqrt{2}}$$

u(n) : Messpunkt n von Signal u

$u_{Real,k}$: der Realteil der k-ten harmonischen Komponente von u

$u_{Imag,k}$: der Imaginärteil der k-ten harmonischen Komponente von u

U_k : der Effektivwert der k-ten harmonischen Komponente von u

HarmPhase

```
HarmPhase('u','Nharm','Frequency'=50)
```

Berechnet die Phasenverschiebung der 'Nharm'-ten harmonischen Komponente des Signals 'u':

$$\varphi_k = -\mathrm{a}\tan\left(\frac{u_{Imag,k}}{u_{Real,k}}\right)$$

$u_{Real,k}$: der Realteil der k-ten harmonischen Komponente von u

$u_{Imag,k}$: der Imaginärteil der k-ten harmonischen Komponente von u

ϕ_k : die Phasenverschiebung der k-ten harmonischen Komponente von u

StarHarmUGeff

```
StarHarmUGeff('u1','u2','u3','Frequency'=50)
```

Berechnet die effektive Gegensystemspannung UGeff.:

$$U_G = \frac{1}{3}\left[u_{1,1} + u_{2,1}\left(-\frac{2}{3}\pi\right) + u_{3,1}\left(-\frac{4}{3}\pi\right)\right]$$

$$U_{Geff} = \frac{\sqrt{U_{G,real}^2 + U_{G,imag}^2}}{\sqrt{2}}$$

$u_{x,1}$: Grundwellenzeiger (komplex) der Phasenspannung u_x

StarHarmUMeff

```
StarHarmUMeff('u1','u2','u3','Frequency'=50)
```

Berechnet die Mitsystemspannnung U_{Meff}:

$$U_M = \frac{1}{3}\left[u_{1,1} + u_{2,1}\left(\frac{2}{3}\pi\right) + u_{3,1}\left(\frac{4}{3}\pi\right)\right]$$

$$U_{Meff} = \frac{\sqrt{U_{M,real}^2 + U_{M,imag}^2}}{\sqrt{2}}$$

$u_{x,1}$: Grundwellenzeiger (komplex) der Phasenspannung u_x

StarHarmUnSym

```
StarHarmUnSym('u1','u2','u3','Frequency'=50)
```

Berechnet die Spannungsunsymmetrie in einem Sternnetz:

$$SYM = \frac{U_{Geff}}{U_{Meff}} \times 100$$

Das Ergebnis wird in % angegeben.

WeightedDistortionFactor

```
WeightedDistortionFactor('u','Nharm'=50,'Frequency'=50)
```

Berechnet den gewichteten Klirrfaktor von 'u' (aller Phasen) mit 'Nharm' Harmonischen:

$$D_W = \frac{\sqrt{\sum_{n=2}^{Nharm} n^2 U_n^2}}{U_1}$$

U_n : Effektivwert der n-ten Harmonischen von u

UnweightedDistortionFactor

```
UnweightedDistortionFactor('u','Nharm'=50,'Frequency'=50)
```

Berechnet den ungewichteten Klirrfaktor von 'u' (aller Phasen) mit 'Nharm' Harmonischen:

$$D_{UW} = \frac{\sqrt{\sum_{n=2}^{Nharm} U_n^2}}{\sqrt{\sum_{n=1}^{Nharm} U_n^2}}$$

U_n : Effektivwert der n-ten Harmonischen von u

10.3.1 TIF

```
TIF('u','Nharm'=50,'Frequency'=50)
```

Berechnet den Telefon-Interferenzfaktor von 'u' unter Berücksichtigung der 'Nharm' Harmonischen:

$$TIF = \frac{1}{U_1}\sqrt{\sum_{n=2}^{Nharm}(K_n \times P_n \times U_n)^2}$$

K_n = 5*n*Frequency

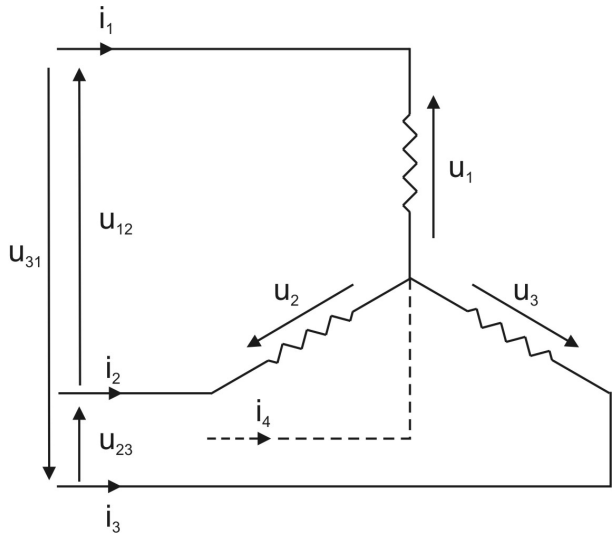
P_n = Gewichtungsfaktor nach BTS (British Telephone System)

U_n : Effektivwert der Spannung der n-ten Harmonischen von u

U_1 : Effektivwert der Spannung der Grundwelle von u

10.4 Stern-Funktionen

Die Stern-Funktionen verwenden die Phasenspannungen und -ströme, um die verschiedenen Leistungswerte zu berechnen.



Argumente

'u1', 'u2', 'u3'	Phasenspannungen (Anschlussspannung = Sqrt(3) * Phasenspannung)
'i1', 'i2', 'i3'	Phasenströme
'i4'	Sternpunktanschluss (iN, Neutralleiter)
'Frequency'	Grundfrequenz

Hinweis

Die Funktionen werden üblicherweise bei Sternnetzen eingesetzt, aber sie können auch für andere Netze verwendet werden, in denen Phasenspannungen und -ströme gemessen werden können.

StarCollectiveUeff

```
StarCollectiveUeff('u1','u2','u3','Frequency'=50)
```

Berechnet die gemeinsame Effektivspannung in einem Sternnetz:

$$U_{eff} = \sqrt{\sum_{x=1}^{4} U_{x\_eff}^2}$$

U_{x_eff} : der Effektivwert der Phasenspannung u_x

$u_4 = u_1 + u_2 + u_3$

StarCollectiveIeff

```
StarCollectiveIeff('i1','i2','i3','i4','Frequency'=50)
```

Berechnet den gemeinsamen Effektivstrom in einem Sternnetz:

$$I_{eff} = \sqrt{\sum_{x=1}^{4} I_{x,eff}^2}$$

$I_{x,eff}$: der Effektivwert des Anschlussstroms i_x

StarActiveP

```
StarActiveP('u1','u2','u3','i1','i2','i3','Frequency'=50)
```

Berechnet die Wirkleistung in einem Sternnetz:

$$P = \sum_{x=1}^{3} \left(\frac{1}{N} \sum_{n=1}^{N} u_x(n) i_x(n) \right)$$

N : Anzahl der Messwerte pro Periode

u_x : Spannung der Phase x

i_x : Strom der Phase x

StarApparentP

```
StarApparentP('u1','u2','u3','i1','i2','i3','i4','Frequency'=50)
```

Berechnet die Scheinleistung in einem Sternnetz:

$$S = U_{eff} I_{eff}$$

U_{eff} : die gemeinsame Effektivspannung

I_{eff} : der gemeinsame Effektivstrom

StarReactiveP

```
StarReactiveP('u1','u2','u3','i1','i2','i3','i4','Frequency'=50)
```

Berechnet die Blindleistung in einem Sternnetz:

$$Q = \sqrt{S^2 - P^2}$$

S : Scheinleistung

P : Wirkleistung

StarReactivePS

```
StarReactivePS('u1','u2','u3','i1','i2','i3','Frequency'=50)
```

Berechnet die vorzeichenbehaftete Blindleistung Q_S im Sternnetz.

StarActivePFactor

```
StarActivePFactor('u1','u2','u3','i1','i2','i3','i4','Frequency'=50)
```

Berechnet den Wirkleistungsfaktor in einem Sternnetz:

$$\cos\varphi = \frac{P}{S}$$

S : Scheinleistung

P : Wirkleistung

StarReactivePFactor

```
StarReactivePFactor('u1','u2','u3','i1','i2','i3','i4','Frequency'=50)
```

Berechnet den Blindleistungsfaktor in einem Sternnetz:

$$\tan\varphi = \frac{Q}{P}$$

Q : Blindleistung

P : Wirkleistung

StarReactivePFactorS

```
StarReactivePFactorS('u1','u2','u3','i1','i2','i3','Frequency'=50)
```

Berechnet den vorzeichenbehafteten Blindleistungsfaktor in einem Sternnetz:

$$\tan\varphi = \frac{Q_S}{P}$$

Q_S : vorzeichenbehaftete Blindleistung

P : Wirkleistung

11 Verschiedene Funktionen

11.1 Count

```
Count('Expression','Level'=0.5,'Hysteresis'=0,'EdgeType'=1,'Reset'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck	
'Level'	Angabe des Niveauwertes	
'Hysteresis'	Angabe eines Hysteresebands	
'EdgeType'	Flankenart, die gezählt wird	
	'EdgeType' <0	nur fallende Flanken (verlassen des Hysteresebands in negativer Richtung)
	'EdgeType' >0	nur steigende Flanken (verlassen des Hysteresebands in positiver Richtung)
	'EdgeType' = 0	fallende und steigende Flanken
'Reset'	Optionaler binärer Parameter zum Zurücksetzen des Zählers. 'Reset' kann selbst auch ein Ausdruck sein.	
	'Reset'=TRUE	Zähler wird zurückgesetzt.
	'Reset'=FALSE	Zählerwert bleibt erhalten/zählt weiter. (Voreinstellung)

Hinweis

Die 'Reset'-Bedingung darf nicht auf die *Count*-Funktion selbst bezogen sein.

Beschreibung

Die Funktion zählt die Durchgänge von 'Expression' durch das Niveau 'Level'.

Mit dem Parameter 'Hysteresis' können Sie ein Toleranzband angeben, was zu gleichen Teilen oberhalb und unterhalb von 'Level' liegt. Nur komplette Durchgänge durch das Toleranzband werden gezählt.

Der Parameter 'EdgeType' bestimmt, welche Flanken gezählt werden. Der Parameter 'Reset' dient zum Rücksetzen des Zählerwertes auf 0. 'Reset' können Sie auch als Ausdruck formulieren.

Tipp

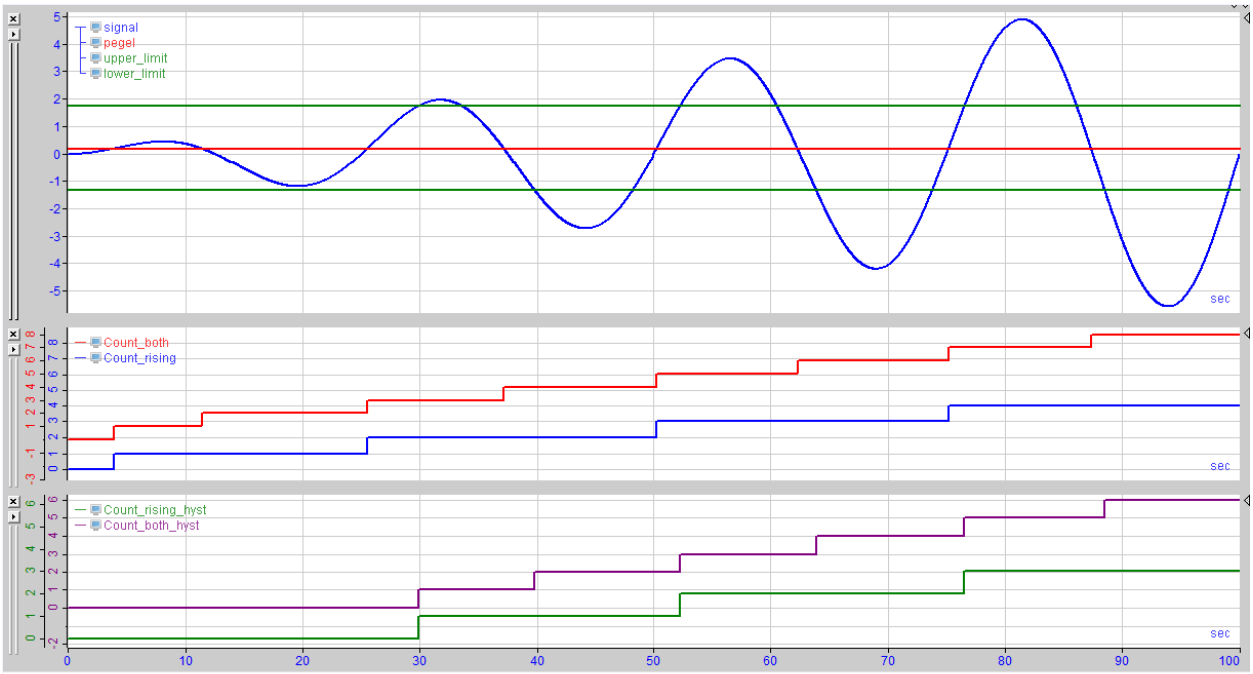
Die *Count*-Funktion können Sie auch für Digitalsignale verwenden. Geben Sie dafür als Pegel 0.5 und als Hysterese z. B. 0.1 ein. Damit werden alle Wechsel von FALSE nach TRUE und umgekehrt erfasst und gezählt.

Beispiel
Angaben:

- 'Level': 2.5
- 'Hysteresis': 2.0

Ergebnis:

Bei Verwendung der Hysteresis werden Pegeldurchgänge in steigender Richtung erst bei 'Expression' > 3.5 und in fallender Richtung erst bei 'Expression' < 1.5 gezählt.



Signal definitions

	Show	SignalName	Expression	Comment 1	Comment 2	Unit	Color	Thickness
5	☑	upper_limit	[pegel] + 0.5 * [hysterese]					2
6	☑	lower_limit	[pegel] - 0.5 * [hysterese]					2
7	☑	Count_both	Count ([signal], [pegel], 0, 0)					2
8	☑	Count_rising	Count ([signal], [pegel])					2
9	☑	Count_rising_hyst	Count ([signal], [pegel], [hysterese])					2
10	☑	Count_both_hyst	Count ([signal], [pegel], [hysterese], 0)					2

11.2 Debounce

```
Debounce('Expression','Debounce interval')
```

Argumente

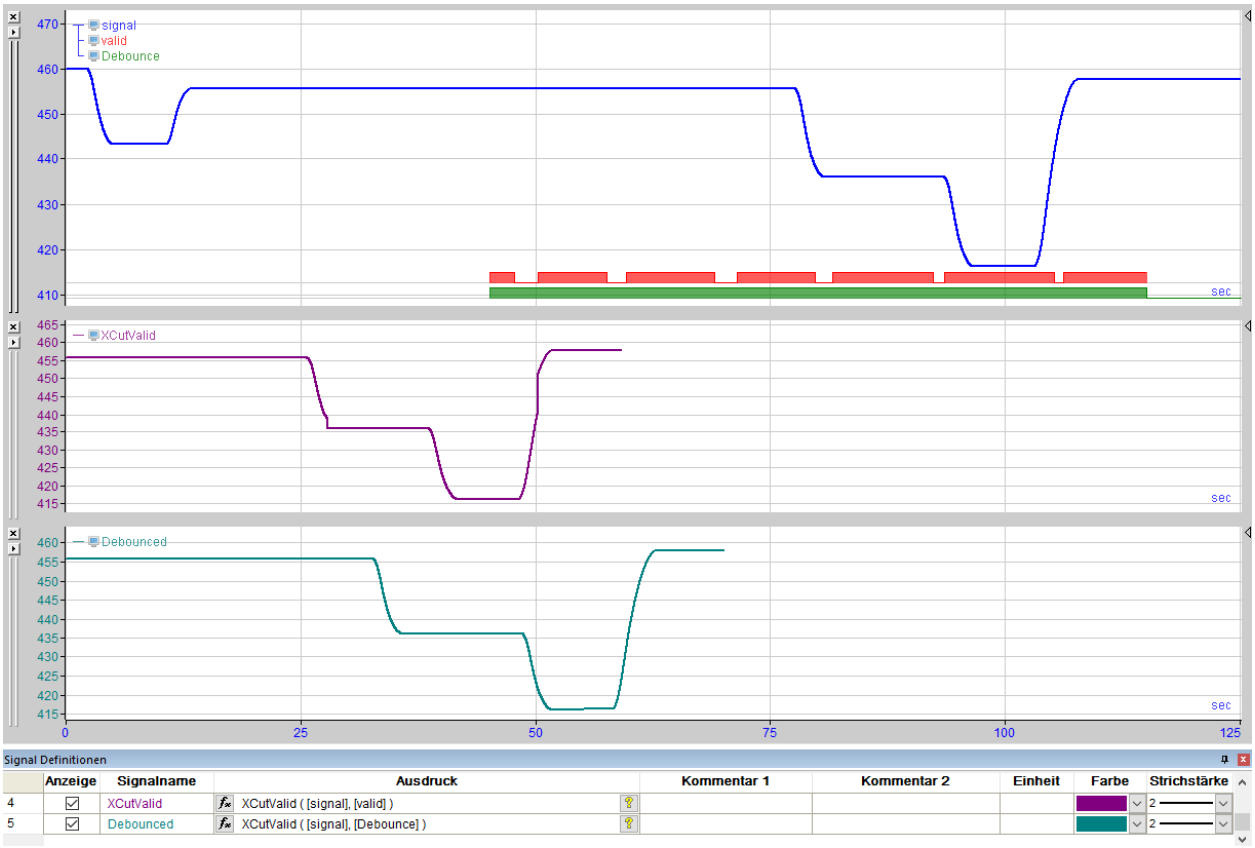
'Expression'	Signal, das entprellt wird
'Debounce interval'	Totzeit

Beschreibung

Diese Funktion liefert als Ergebnis einen entprellten Signalverlauf von 'Expression' mit 'Debounce interval' als Totzeit in [s]. Bei längenbezogenen Signalen wird 'Debounce interval' als Weg in [m] interpretiert.

Die Funktion arbeitet ähnlich einem abfallverzögerten Zeitrelais, mit dem Unterschied, dass der Signalwechsel von TRUE nach FALSE (fallende Flanke) zeitrichtig, also unverzögert dargestellt wird, wenn innerhalb der eingestellten Zeit nicht wieder ein Wechsel von FALSE nach TRUE (steigende Flanke) folgt.

So können flatternde Signale, z. B. von Fotozellen oder Endschaltern beruhigt werden. Besonders wichtig ist dies, wenn diese Signale als Bedingungen in Operationen wie *XMarkValid* oder *XCutValid* verwendet werden, da bei jedem Aussetzer die Berechnung der Operation unterbrochen und damit Ergebniswerte verloren gehen würden. Der Unterschied ist im nachfolgenden Bild deutlich zu sehen.



11.3 DynSignal

```
DynSignal('TextExpression')
```

Beschreibung
Diese Funktion liefert den Wert des Signals bzw. des Ausdrucks 'TextExpression'.

'TextExpression' muss einen Text ergeben, den *ibaAnalyzer* als einen neuen, gültigen Ausdruck interpretieren kann. In der Folge liefert *ibaAnalyzer* dann das Signal, das durch Verarbeitung dieses Ausdrucks ermittelt wurde.

Ein typischer, möglicher Anwendungsfall ist die Erzeugung einer gültigen Signal-ID in der geladenen Messdatei, z. B. eine Signal-ID, bestehend aus [Modulnummer:Signalnummer] für Analogsignale bzw. [Modulnummer.Signalnummer] für Digitalsignale.

Die Funktion wird insbesondere in Zusammenhang mit dem Messpunktkonzept aus dem Bereich der iba-Energiemesstechnik angewendet. Dabei werden verschiedene Metadaten ausgelesen, die zu jedem Messpunkt in strukturierter Form im Info-Bereich der Messdatei gespeichert sind. Der Inhalt dieser Metadaten ist jeweils die Modulnummer, unter der der entsprechende Messwert in der Messdatei gespeichert ist.

Damit ist es möglich, die Auswertung von Messpunktdaten zu standardisieren, selbst wenn die I/O-Konfiguration von Anlage zu Anlage unterschiedlich ist.

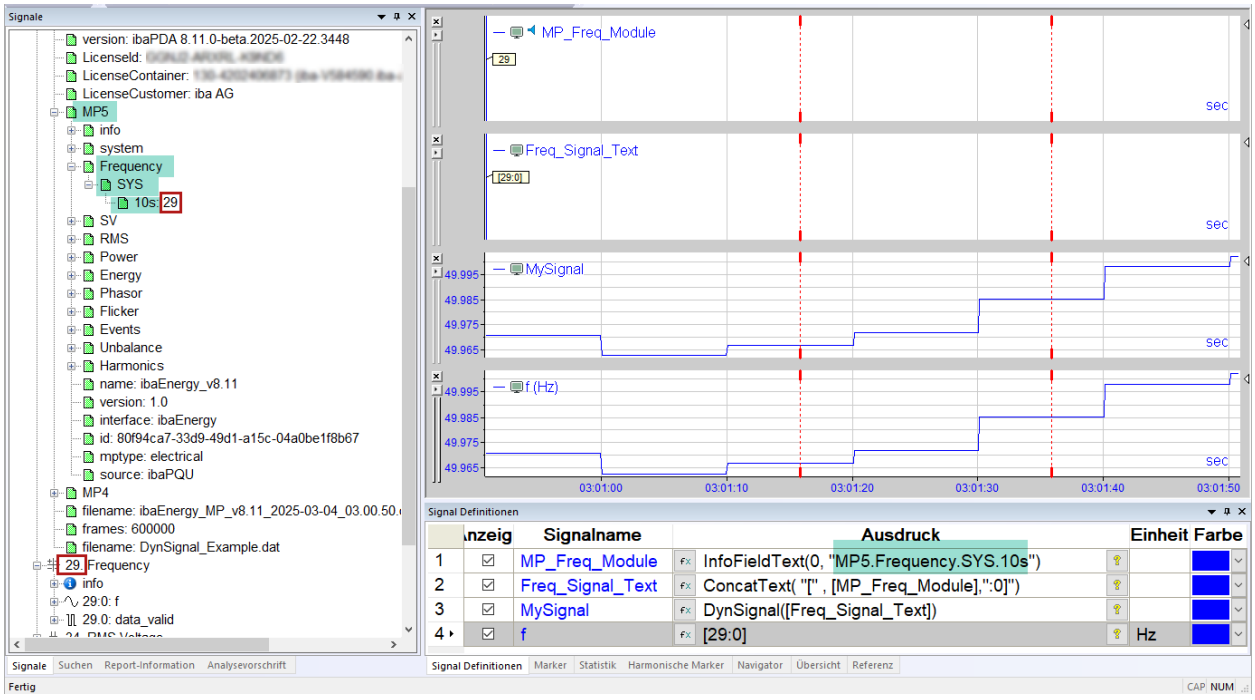
Beispiel
In der Analyse soll der berechnete 10-s-Wert (Zehn-Sekunden-Wert) der Frequenz an einem Messpunkt ermittelt werden.

Die Modulnummer, unter der dieser Wert als Signal gespeichert ist, steht in dem Metadatum bzw. Infofeld:

```
Messpunkt.Frequency.SYS.10s
```

Im folgenden Beispiel wird der 10-s-Frequenzwert an Messpunkt 5 (MP5) ermittelt.

Der Messwert ist in der Messdatei im Modul 29, Signal 0 gespeichert, hat also die Kennung [29:0].



Die Ermittlung erfolgt in drei Schritten, entsprechend der drei Zeilen in der Signaltabelle:

1. Ermittlung des Inhalts des Infofeldes *MP5.Frequency.SYS.10s* mit der Funktion *InfoFieldText*. Das Ergebnis "29" wird als Textsignal *MP_Freq_Module* zwischengespeichert (siehe oberster Signalstreifen).
2. Um die korrekte Form der Signalangabe zu erhalten, werden mit der *ConCat*-Funktion das Zeichen "[", das Textsignal *MP_Freq_Module* und die Zeichen ":0]" zu einem String zusammengesetzt. Der zusammengesetzte Text wird als Textausdruck *Freq_Signal_Text* zwischengespeichert (siehe zweiten Signalstreifen von oben).
   Bei der Angabe der Signalnummer (hier: ":0") können Sie auch Wildcards verwenden oder Bereiche eingeben.
   - ":*" – alle Signale des Moduls
   - ":n-m" – Signale von-bis (z. B. ":20-29")
3. Schließlich wird über die Funktion *DynSignal* der Textausdruck *Freq_Signal_Text* als Signal-ID bestehend aus [Modulnummer:Signalnummer] interpretiert und dessen Wert als Signal *MySignal* ausgelesen (siehe dritter Signalstreifen von oben).
4. Diese Zeile zeigt das Signal [29:0] an. Der Wert ist identisch mit dem von *MySignal*.

11.4 Envelope

```
Envelope('Expression','Interval')
```

Argumente

'Expression'	Signal oder Ausdruck, um den die Hüllkurve gebildet wird
'Interval'	X-Achsen-Intervall

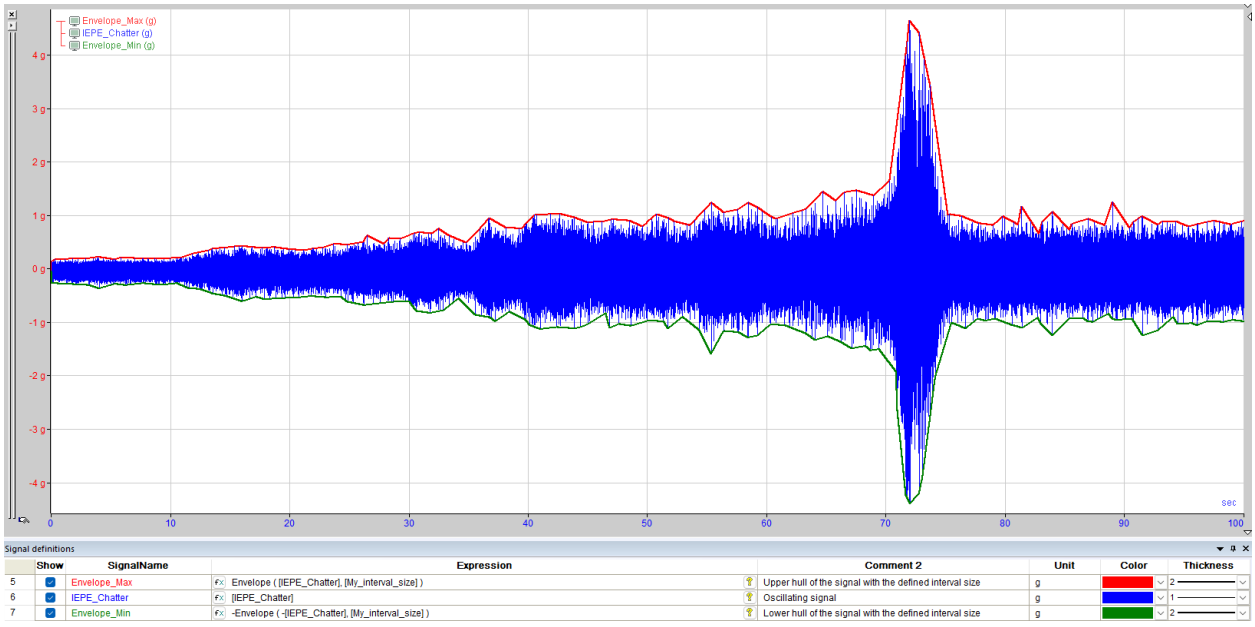
Beschreibung

Diese Funktion berechnet die obere Hüllkurve um ein Signal. Die Hüllkurve entsteht durch Verbindung der Maxima. Die Qualität der Hüllkurve kann mit dem Parameter 'Interval' beeinflusst werden. Ohne Angabe dieses Parameters werden nur die größten Maxima über die Aufzeichnungsdauer berücksichtigt. Mit dem Parameter 'Interval' geben Sie die Intervalllänge in Einheiten der X-Achse (s, m, Hz, 1/m) vor. Es werden dann auch die Maxima innerhalb dieser Intervalle berücksichtigt und die Hüllkurve schmiegt sich stärker an die Signalkurve an.

Tipp

Um auch eine Hüllkurve auf der Unterseite der Signalkurve zu erhalten, können Sie die gleiche Funktion in folgender Form verwenden. Damit werden dann die Minima miteinander verbunden.

```
-Envelope (-'Expression','Interval')
```



11.5 False und True

```
False() und True()
```

Beschreibung

Diese Operanden liefern den konstanten Wert 0 oder 1.

In booleschen Operationen (AND, OR usw.) wird der Wert als logisch 0 (false) oder logisch 1 (true) interpretiert. In arithmetischen Operationen und in Zusammenhang mit Analogwerten wird der Wert als 0,0 oder1,0 interpretiert ("feste Null" bzw. "feste Eins").

11.6 GetBit und GetBitMask

GetBit

```
GetBit('Expression','Bitnumber')
```

Beschreibung

Diese Funktion liefert als Ergebnis den booleschen Wert des Bits 'Bitnumber' von 'Expression' nach Rundung auf den nächsten Integerwert. Die Rundungsgrenze liegt jeweils bei 0,5-Schritten. (2,48 --> 2; 2,50 -->3). Gültige Bitnummernreihenfolge: 0 (LSB) bis 15 (MSB).

Hinweis

Integerwerte mit 64 Bit können nicht mit dieser Funktion ausgewertet werden, da sie von *ibaPDA* nicht unterstützt werden und somit nicht in einer Messdatei enthalten sein können.

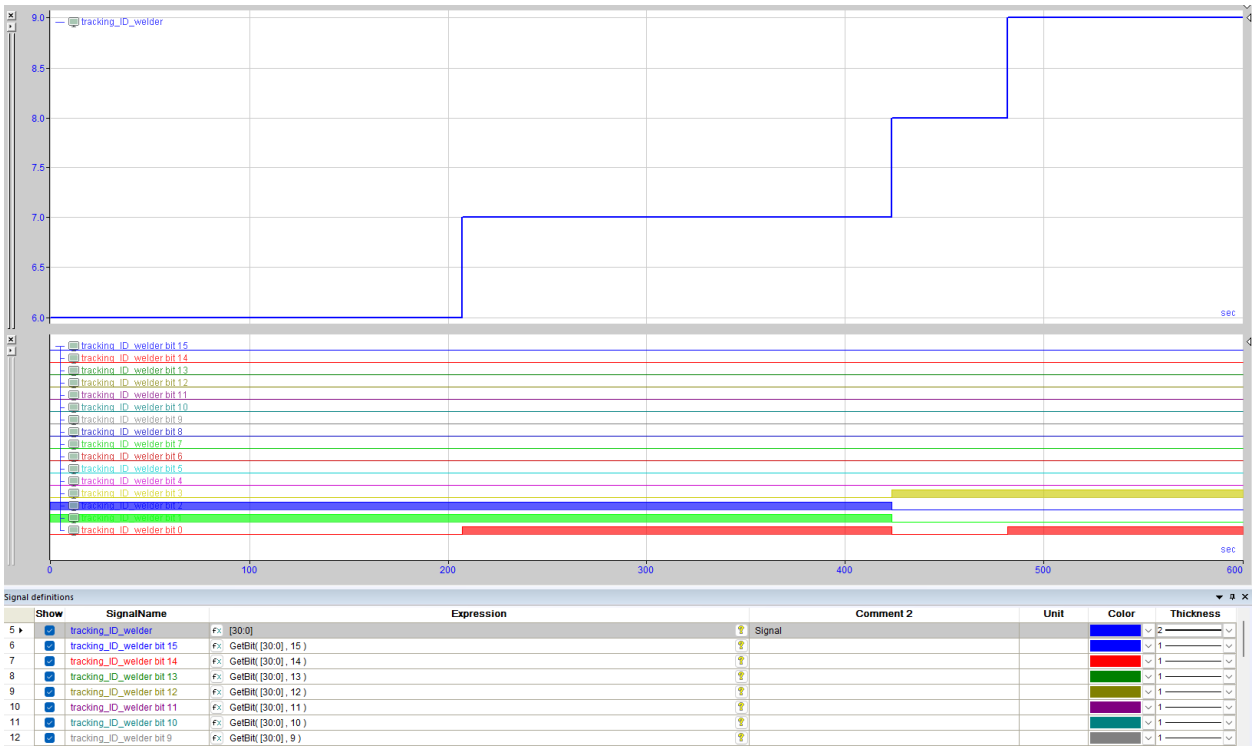
Beispiel

In der folgenden Tabelle ist als Beispiel das niederwertigste Byte eines Integerwerts dargestellt mit den Bits 0…7. Um die Werte 0…8 abzubilden, werden die einzelnen Bits wie mit "X" markiert gesetzt (X = TRUE). Die linke Spalte zeigt die Ausgangswerte, die durch die summierten Potenzen von 2 (Kopfzeile) dargestellt werden, z. B. 1 = 2 ^ 0; 2 = 2 ^ 1; 3 = 2 ^ 0 + 2 ^1.

Bitnr.	7	6	5	4	3	2	1	0
0								
1								X
2							X	
3							X	X
4						X		
5						X		X
6						X	X	
7						X	X	X
8					X			

Tipp

Wenn Sie mehrere 8-, 16- oder 32-Bit Integerwerte in Einzelbits zerlegen wollen, können Sie im Signalbaum auf dem gewünschte Signal das Kontextmenü öffnen und den Befehl *Bits anzeigen* wählen. Alle Bits werden dann als einzelne Digitalsignale angezeigt, ohne die *GetBit*-Funktion zu programmieren. Intern wird der gleiche Mechanismus wie bei *GetBit* angewendet.



GetBitMask

```
GetBitMask('Expression','Bitnumber')
```

Beschreibung

Diese Funktion interpretiert 'Expression' als Bitmaske eines Floatwerts und liefert als Ergebnis den Wert des Bits 'Bitnumber'. Gültiger Bereich: 0 (LSB) bis 31 (MSB)

Diese Funktion wurde speziell für die Arbeit mit Daten von SimadynD in einem speziellen Anwendungsfall entwickelt, wo bis zu 32 digitale Werte gepackt als Float-Variable aufgezeichnet werden. Die Funktion *GetBitMask* wertet lediglich die Valenz des spezifizierten Bits 'Bitnumber' aus, ungeachtet dessen, ob es Teil der Mantisse oder des Exponenten ist. Im Gegensatz zur Funktion *GetBit* wird keine Rundung zum Integer vorgenommen.

Tipp

Wenn Sie ein oder mehrere 32-Bit-Floating-Werte in Einzelbits zerlegen wollen, können Sie im Signalbaum das Kontextmenü auf dem gewünschte Signal öffnen und den Befehl *Bits anzeigen* wählen. Alle Bits werden dann als einzelne Digitalsignale angezeigt. Intern wird der gleiche Mechanismus wie bei *GetBitMask* angewendet.

11.7 HighPrecision

```
HighPrecision('Expression')
```

Beschreibung
Mit dieser Funktion wird 'Expression' als Größe mit doppelter Genauigkeit gekennzeichnet. Berechnungen, die dann mit 'Expression' durchgeführt werden, erfolgen mit doppelter Genauigkeit, auch wenn der ursprüngliche Ausdruck nur einfache Genauigkeit hat.

Doppelte Genauigkeit hat zwar den Vorteil, dass Berechnungen genauer ausgeführt werden können. Allerdings wird auch doppelt so viel Speicherplatz belegt. *ibaAnalyzer* entscheidet daher automatisch auf Basis der Eingangsargumente für eine Berechnung, welche Genauigkeit verwendet wird.

11.8 InfoField, ChannelInfoField und ModuleInfoField

Diese Funktionen lesen Informationen aus einem Info-Feld einer Messdatei, eines Signals oder eines Moduls aus.

Hinweis

Die Funktionen *InfoField, ChannelInfoField* und *ModuleInfoField* erwarten einen numerischen Wert. Falls Text ausgelesen werden soll, verwenden Sie die Funktionen *InfoFieldText*, *ChannelInfoFieldText* und *ModuleInfoFieldText*. Siehe ↗ *InfoFieldText, ChannelInfoFieldText und ModuleInfoFieldText*, Seite 126.

Argumente

'Index'	Index der Messdatei, des Signals oder Moduls
'InfoField'	Infofeld, das ausgelesen wird; in Anführungszeichen setzen.
'Begin'	Optional: Erstes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der komplette Inhalt ausgelesen.
'End'	Optional: Letztes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der Inhalt von 'Begin' bis zum letzten Zeichen ausgelesen.

InfoField

```
InfoField('FileIndex',"'InfoField'",'Begin'=0,'End'=Text end)
```

Tipp

Wenn Sie im Signalbaum einen Doppelklick auf das gewünschte Infofeld machen, dann fügt *ibaAnalyzer* die entsprechende Funktion automatisch als neues Signal in die Signaltabelle ein. Anschließend brauchen Sie bei Bedarf nur noch den Signalnamen und Anfang oder Ende anpassen. Diese Methode funktioniert auch im Eingabefeld des Ausdruckseditors. Die Funktion wird dann an der Cursorposition eingefügt.

Wenn Sie den Inhalt eines Infofeldes als Textkanal auslesen wollen, verwenden Sie die Funktion *ChannelInfoFieldText*.

ChannelInfoField

```
ChannelInfoField('ChannelIndex',"'InfoField'",'Begin'=0,'End'=Text end)
```

ModuleInfoField

```
ModuleInfoField('FileIndex','ModuleIndex',"'InfoField'",'Begin'=0,'End'=Text end)
```

Hinweis

Bei dieser Funktion müssen Sie zwei Indizes angebe: Den Index der Messdatei als erstes Argument und den Index des Moduls als zweites.

11.9 LimitAlarm und WindowAlarm

LimitAlarm

```
LimitAlarm('Expression','Limit','DeadBand','Time')
```

Argumente

'Expression'	Signal oder Ausdruck
'Limit'	Grenzwert, ab dem die Funktion TRUE zurückgibt
'DeadBand'	Angabe einer Totzone unterhalb des Grenzwertes, innerhalb der die Funktion nicht auf FALSE zurückgesetzt wird
'Time'	Angabe der Zeit, die 'Expression' oberhalb des Grenzwertes liegen muss, bis die Funktion auf TRUE gesetzt wird

Beschreibung

Diese Funktion überwacht das Signal ('Expression') und setzt das Ergebnis auf TRUE, wenn das Signal länger als die angegebene Zeit ('Time') oberhalb des Grenzwertes ('Limit') liegt. Das Ergebnis der Funktion wird wieder FALSE, wenn das Signal den Grenzwert um den unter Totzone ('DeadBand') angegebenen Wert unterschreitet.

Tipp

Sie können die Funktion *LimitAlarm* auch für einen unteren Grenzwert verwenden. Spiegeln Sie dafür das Signal und der Grenzwert, d. h. mit (-1) multiplizieren.

Zum Beispiel: `LimitAlarm([0:1] *(-1), 9 *(-1), 0.5, 0.4)`

WindowAlarm

```
WindowAlarm('Expression','Limit1','DeadBand1','Limit2','DeadBand2','Time')
```

Argumente

'Expression'	Signal oder Ausdruck
'Limit1'	Oberer Grenzwert, ab dem die Funktion TRUE zurückgibt
'DeadBand1'	Angabe der Totzone unterhalb des oberen Grenzwertes ('Limit1'), innerhalb der die Funktion nicht auf FALSE zurückgesetzt wird
'Limit2'	Unterer Grenzwert, ab dem die Funktion TRUE zurückgibt
'DeadBand2'	Angabe der Totzone oberhalb des unteren Grenzwertes ('Limit2'), innerhalb der die Funktion nicht auf FALSE zurückgesetzt wird
'Time'	Angabe der Zeit, die 'Expression' oberhalb des Grenzwerts oder unterhalb des Grenzwerts liegen muss, bis die Funktion auf TRUE gesetzt wird

Beschreibung

Diese Funktion überwacht das Signal ('Expression') und setzt das Ergebnis auf TRUE, wenn das Signal länger als die angegebene Zeit ('Time') außerhalb des Bereichs zwischen oberem Grenzwert ('Limit1') und unterem Grenzwert ('Limit2') liegt. Das Ergebnis der Funktion wird wieder FALSE, wenn das Signal den oberen Grenzwert um den unter Totzone1 ('DeadBand1') angegebenen Wert unter-, bzw. den unteren Grenzwert um den unter Totzone2 ('DeadBand2') angegebenen Wert überschreitet.

11.10 ManY

```
ManY('Xbase','y0','y1',…)
```

Argumente

'Xbase'	Abtastrate des Ergebnissignals
'y0', 'y1',…	Y-Werte des Ergebnissignals

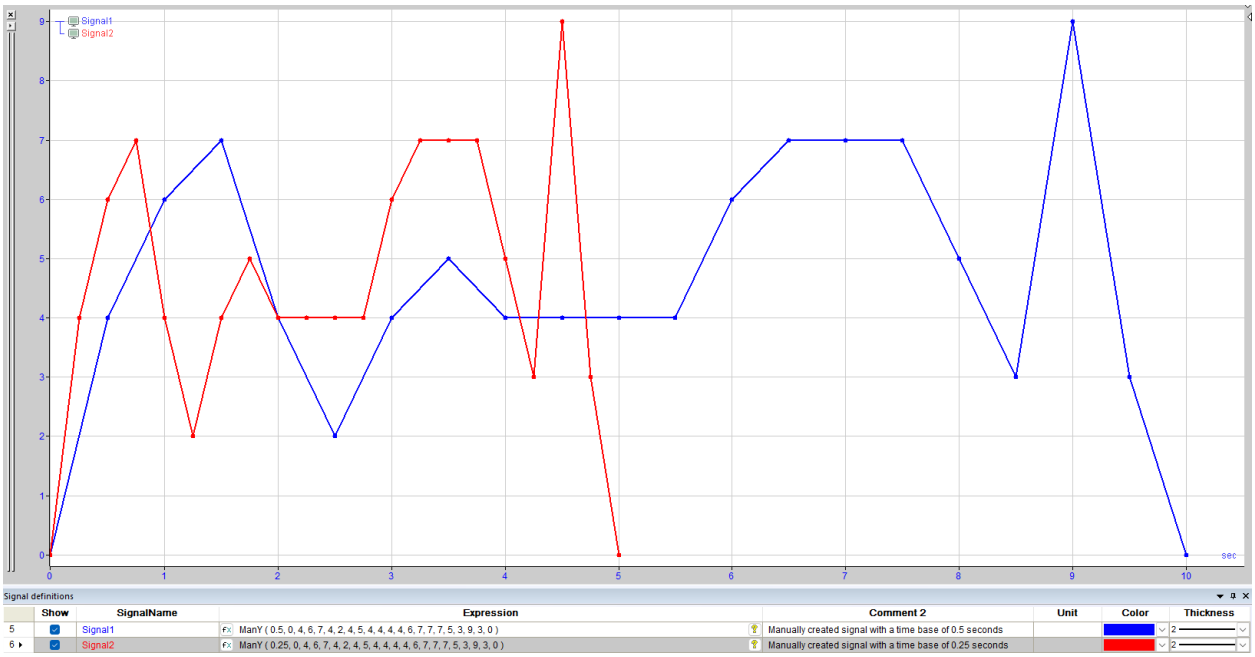
Beschreibung

Mit dieser Funktion können Sie manuell ein Signalverlauf mit den "Messwerten" 'y0'…'y99' erzeugen, die jeweils einen Zeit- oder Wegabstand 'Xbase' voneinander haben. Die Angabe von 'Xbase' ist für zeitbezogene Werte in Sekunden und für längenbezogene Werte in Meter. Die Menge der Punkte ist auf 100 begrenzt.

So können Sie z. B. Referenzkurven eingeben, mit denen Sie dann die real gemessenen Signale vergleichen. Oder Sie fügen einer Analyse Daten hinzu, die nicht als Messwert vorliegen. Außerdem können Sie mit dieser Funktion auch Textsignale manuell erzeugen und mit unterschiedlichen Werten belegen.

Tipp

Wenn Sie mindestens einen der Parameter y0 bis maximal y99 in Anführungszeichen setzen, werden eingegebene Zeichen nicht als Zahlenwerte sondern als ASCII-Zeichen übernommen.



11.11 PulseFreq

```
PulseFreq('Expression','Omega'=0,'EdgeType'=2,'MinFreq'=0.05)
```

Argumente

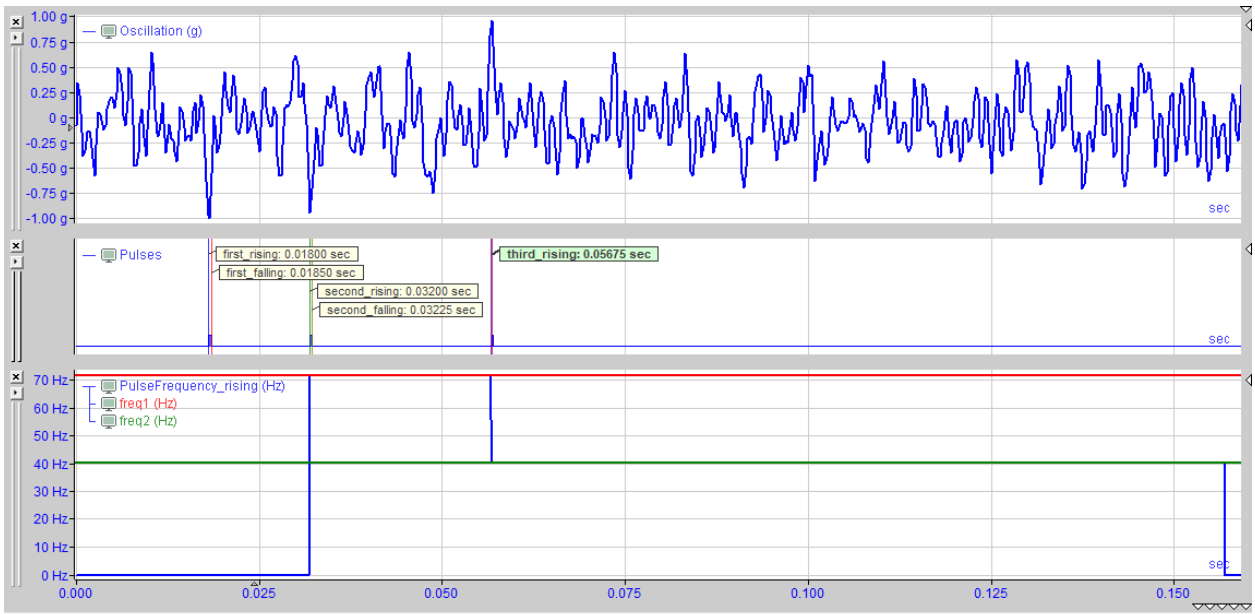
'Expression'	Pulszählersignal	
'Omega'	Filterfrequenz	
'EdgeType'	Flankenart, die gezählt wird	
	'EdgeType' = -1	nur fallende Flanken
	'EdgeType' = 0	steigende und fallende Flanken
	'EdgeType' = 1	nur steigende Flanken
	'EdgeType' = 2	'Expression' ist ein Impulszähler
'MinFreq'	Kleinste Frequenz, die dargestellt wird	

Beschreibung

Diese Funktion berechnet die Frequenz von Impulsen oder Impulszählern 'Expression'. Ergebniseinheit ist Pulse/Sek. bzw. Hz.

Ein Tiefpassfilter mit einer Grenzwinkelgeschwindigkeit 'Omega' wird auf das Ergebnis angewendet. Wenn 'Omega' 0 ist, dann ist der Tiefpassfilter deaktiviert. 'EdgeType' bestimmt, welche Flanken der Pulse gezählt werden sollen. Als berechnete Frequenz wird Null zurückgegeben, wenn während 1000 Abtastungen kein Puls auftritt.

Die Frequenz wird erst ausgegeben, sobald die zweite Flanke auftritt. Der Wert basiert auf dem Abstand zwischen der ersten und der zweiten Flanke, wobei die Berechnung erst zum Zeitpunkt der zweiten Flanke erfolgt.



Signal definitions

	Show	SignalName	Expression	Comment 2	Unit	Color	Thickness
5	☑	Oscillation	[Oscillation]	Oscillating signal	g		2
6	☑	Pulses	[Oscillation] > 0.9 OR [Oscillation] < - 0.9	Pulses where the signal leaves a dead ban			1
7	☑	PulseFrequency_risi	PulseFreq ([Pulses], 0, 1,10)	Frequency of the rising pulse edges	Hz		2
8	☑	freq1	1 / ([second_rising]-[first_rising])	Frequency from the first to the second rising	Hz		2
9	☑	freq2	1 / ([third_rising] - [second_rising])	Frequency from the second to the third risin	Hz		2

Diese Funktion ist speziell für die Auswertung des WAGO-Inkrementalgebers 750-631 erstellt worden. Die Funktion kann zur Geschwindigkeitsberechnung aus dem Pulszählerstand benutzt werden. Der Pulszählerstand wird unter Beachtung eines möglichen Überlaufs differenziert. Da das Ergebnis dieser Differentiation mit Störfrequenzen bzw. einem Rauschen versehen sein kann, wird anschließend ein Tiefpassfilter darauf angewendet. Die einzustellende Filterfrequenz sollte etwas oberhalb der maximalen Pulsfrequenz liegen.

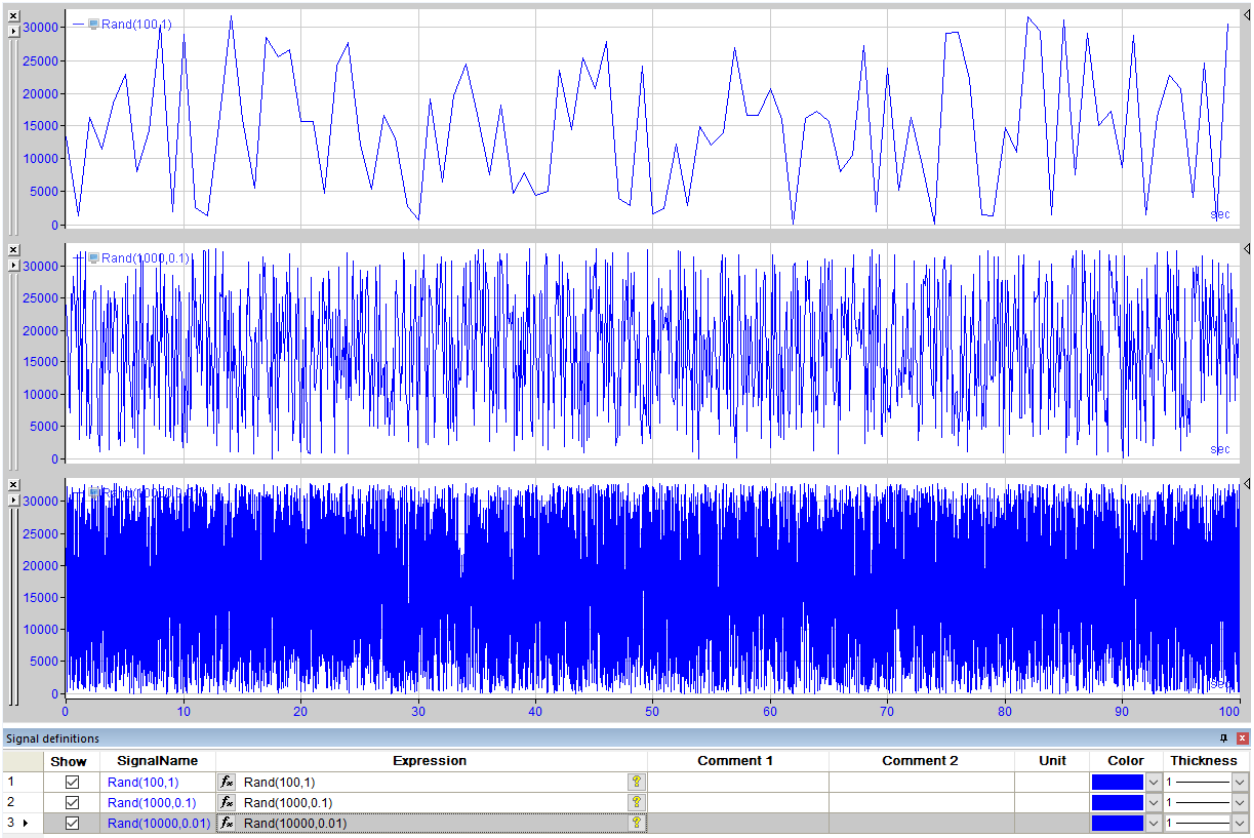
11.12 Rand

```
Rand('Count','XBase')
```

Beschreibung

Diese Funktion erzeugt ein Signal bestehend aus Zufallszahlen im Bereich von 0 bis 32767. Der Parameter 'Count' legt dabei die Länge des Signalstreifens (in Sekunden oder Meter) fest. Der Parameter 'XBase' definiert den Abstand der Samples.

Die nachfolgende Abbildung zeigt drei Signale, die 100 s lang dauern, jedoch aus einer unterschiedlichen Anzahl Punkten bestehen. Die Zeitbasis 'XBase' beträgt 1 s, 100 ms und 10 ms.



Tipp

Wenn Sie für die Zufallszahlen durch einen Bereich "a bis b" eingrenzen wollen, können Sie folgende Formel nutzen:

Rand('Count','XBase') * (b-a) - a

11.13 SampleAndHold

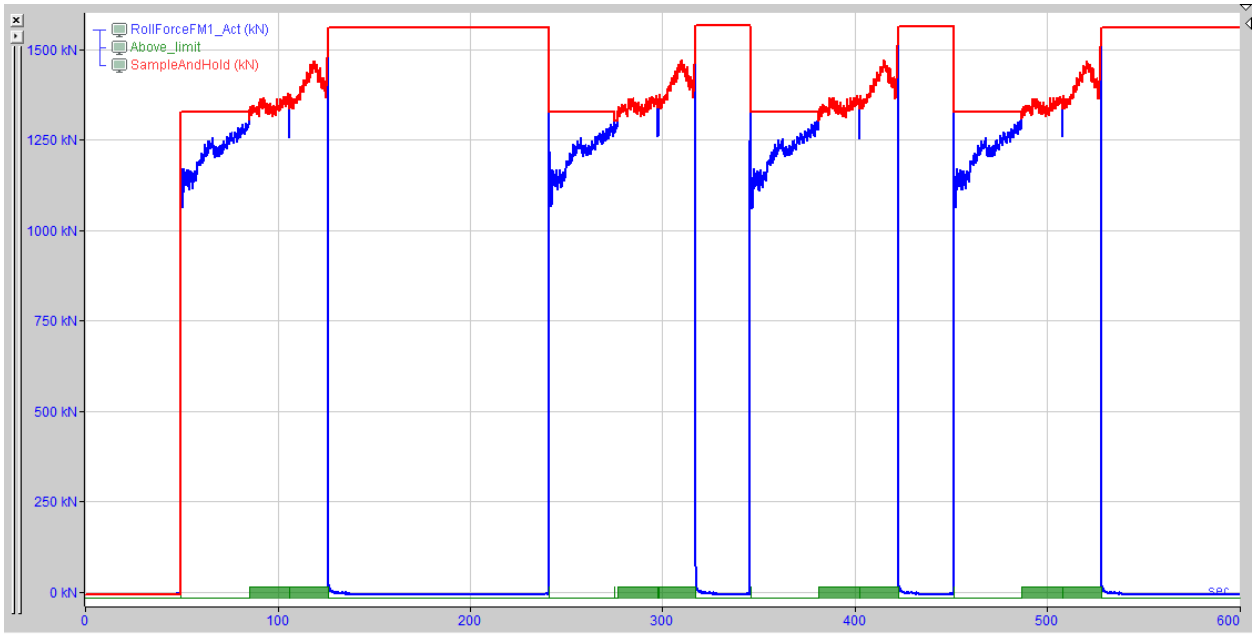
```
SampleAndHold('Expression','Sample')
```

Argumente

'Expression'	Signal oder Ausdruck
'Sample'	Parameter, der bestimmt, ob die Funktion dem Signal folgt (1) oder den letzten Messwert hält (0). 'Sample' kann selbst eine Bedingung sein oder durch eine andere Funktion bestimmt werden.

Beschreibung

Diese Funktion ist eine Abtast-Halte-Funktion. Der Ausgang folgt 'Expression', wenn 'Sample' = TRUE. Er bleibt unverändert, wenn 'Sample' = FALSE.



Signal definitions

	Show	SignalName	Expression	Comment 2	Unit	Color	Thickness
5	☑	RollForceFM1_Act	[8:0]	Rollforce Finishing Mill Stand 1 total (Actual value)	kN		2
6	☑	Above_limit	[8:0]>1300	Areas where the roll force exceeds the limit			1
7	☑	SampleAndHold	SampleAndHold ([8:0], [Above_limit])	Returns the rollforce when the signal exceeds the limit, else the last value is hold	kN		2

11.14 SampleOnce

```
SampleOnce('Expression','Sample')
```

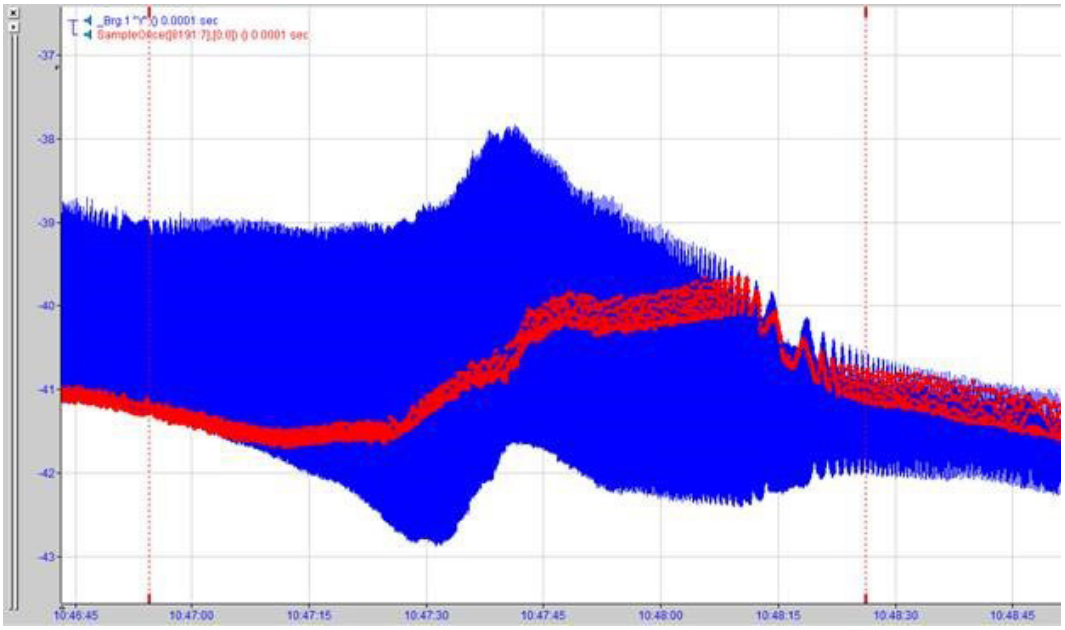
Argumente

'Expression'	Signal oder Ausdruck
'Sample'	Digitalsignal, dessen steigende Flanken die Abtastpunkte festlegen.

Beschreibung
Diese Funktion tastet ein Eingangssignal 'Expression' an einzelnen Punkten neu ab, die durch die steigenden Flanken des Digitalsignals 'Sample' bestimmt werden. Das Ergebnis enthält einen Messpunkt pro steigender Flanke und ist ungültig in den Bereichen dazwischen.

Beispiel
Diese Funktion kann genutzt werden um ein Phasengeber-Signal (Keyphasor) im Zeitbereich darzustellen. Immer wenn der Phasengeber auf TRUE springt, wird das ursprüngliche Signal abgetastet. Durch Überlagerung beider Darstellungen werden die Zeitpunkte des Phasengebers passend dargestellt. Im Beispiel unten kann eine 180° Phasenverschiebung beim Durchlaufen einer Resonanz erkannt werden.

Die nachfolgende Abbildung zeigt die Funktion *SampleOnce* angewendet auf ein Schwingungssignal.



11.15 Sign

```
Sign('Expression')
```

Beschreibung
Diese Funktion liefert als Ergebnis das Vorzeichen von 'Expression'.

- 'Expression' > 0 --> +1
- 'Expression' = 0 --> 0
- 'Expression' < 0 --> -1

11.16 Technostring

```
Technostring('Index','Begin','End')
```

Argumente

'Index'	Index der Messdatei
'Begin'	Anfang des auszulesenden Bereichs
'End'	Ende des auszulesenden Bereichs

Beschreibung
Diese Funktion extrahiert den String aus dem Messdatei-Index 'Index' zwischen 'Begin' und 'End'. Standardstartindex ist 0. Somit können Informationen aus dem Technostring als Signale interpretiert werden (nur numerische Zeichen).

Ausgewertet wird die Technostring-Information, die im Info-Zweig im Signalbaumfenster zu sehen ist. Voraussetzung ist natürlich, dass *ibaPDA* die Technostring-Informationen in der Messdatei abgespeichert hat.

'Begin' und 'End' entsprechen der Position der Zeichen im Technostring, die den gewünschten Bereich begrenzen, der als Signal ausgewertet werden soll. Es können nur numerische Zeichen ausgewertet werden. Führende Nullen werden verworfen.

Die Angabe des 'Index' ist nur dann erforderlich, wenn mehrere Messdateien gleichzeitig geöffnet sind. Die Datei an oberster Stelle im Signalbaumfenster hat den Index 0. Alle weiteren Dateien dann von oben nach unten 1, 2, usw. Wenn nur eine Datei geöffnet ist, dann muss der Index stets = 0 sein.

11.17 YatX und SetYatX

YatX

```
YatX('Expression','X','Continuous'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck
'X'	Position, an welcher der Wert ausgelesen wird
'Continuous'	Optionaler Parameter, um variable Werte von 'X' zuzulassen

Beschreibung
Diese Funktion liefert als Ergebnis den Y-Wert von 'Expression' bei Position 'X' auf der X-Achse zurück. Die Funktion können Sie auf zeitbasierte und längenbasierte Signale anwenden.

Im Standardmodus wird der Parameter 'Continuous' nicht angegeben oder ist FALSE bzw. 0. Die Funktion erwartet dann einen konstanten X-Wert und liefert einen konstanten Y-Wert als Ergebnis.

Der Parameter 'X' kann aber auch variabel sein, d. h. er kann selbst eine Funktion sein. In diesem Fall müssen Sie den kontinuierlichen Modus aktivieren, indem Sie den Parameter 'Continuous' auf TRUE bzw. 1 setzen. Die Funktion ermittelt dann zu jedem Wert von 'X' den passenden Y-Wert.

SetYatX

```
SetYatX('Expression','SampleValue','X')
```

Argumente

'Expression'	Signal oder Ausdruck, der verändert wird
'SampleValue'	Wert, der an der Stelle 'X' eingefügt wird
'X'	X-Position, an der der Wert 'SampleValue' eingefügt wird

Beschreibung

Mit dieser Funktion können Sie eine Kopie eines Signals erstellen, in der ein Wert geändert wurde. Die Funktion liefert als Ergebnis eine Kopie des Signals 'Expression', bei dem an der Stelle 'X' der Wert 'SampleValue' eingefügt wurde.

Sie können diese Funktion ebenfalls nutzen, um Texte einzufügen.

Je nachdem, ob ein äquidistant gesampeltes Signal vorliegt oder nicht, verhält sich die Funktion anders. Bei äquidistanten Signalen werden folgende Fälle unterschieden:

- Falls 'X' kleiner als der Offset des Signals ist, wird das Signal unverändert zurückgegeben.
- Falls 'X' der Größe des Signals (vgl. *XSize*) plus der Abtastgröße entspricht, wird das Signal um ein Sample mit dem Wert 'SampleValue' verlängert.
- In allen anderen Fällen wird der neue Wert an der Stelle 'X' oder an der nächstkleineren Sample-Position eingefügt.

Für nicht äquidistant abgetastete Signale ersetzt die Funktion den Wert an der Stelle 'X', falls vorhanden, oder fügt ein neues Sample ein.

12 Filter-Funktionen

Digitale Filter, die Sie mit dem Filtereditor erstellt haben, können Sie im System abspeichern. Diese Filter stehen dann auch im Ausdruckseditor als Filterfunktionen zur Verfügung.

12.1 LP

```
Lp('Expression','Omega')
```

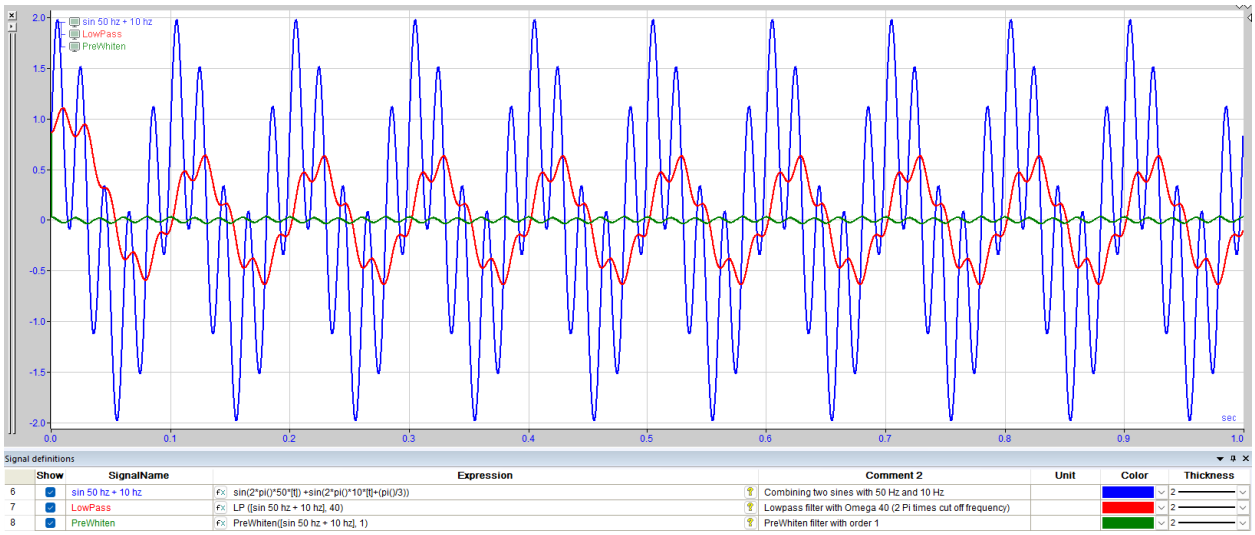
Argumente

'Expression'	Signal oder Ausdruck
'Omega'	2 Pi mal die Eckfrequenz für den Tiefpassfilter

Beschreibung

Diese Funktion wendet einen sehr einfachen digitalen Tiefpassfilter mit einer Eckfrequenz 'Omega'/(2*Pi) an. Bei Anwendung auf ein Signal 'Expression' liefert die Funktion ein Signal, bei dem die Frequenzen oberhalb der Eckfrequenz gedämpft sind.

Beachten Sie, dass die Abschwächung aufgrund des einfachen Algorithmus eher gering ist, und es wird empfohlen, den Filterdesigner zu verwenden, wenn spezifische Filter erforderlich sind.



12.2 PreWhiten

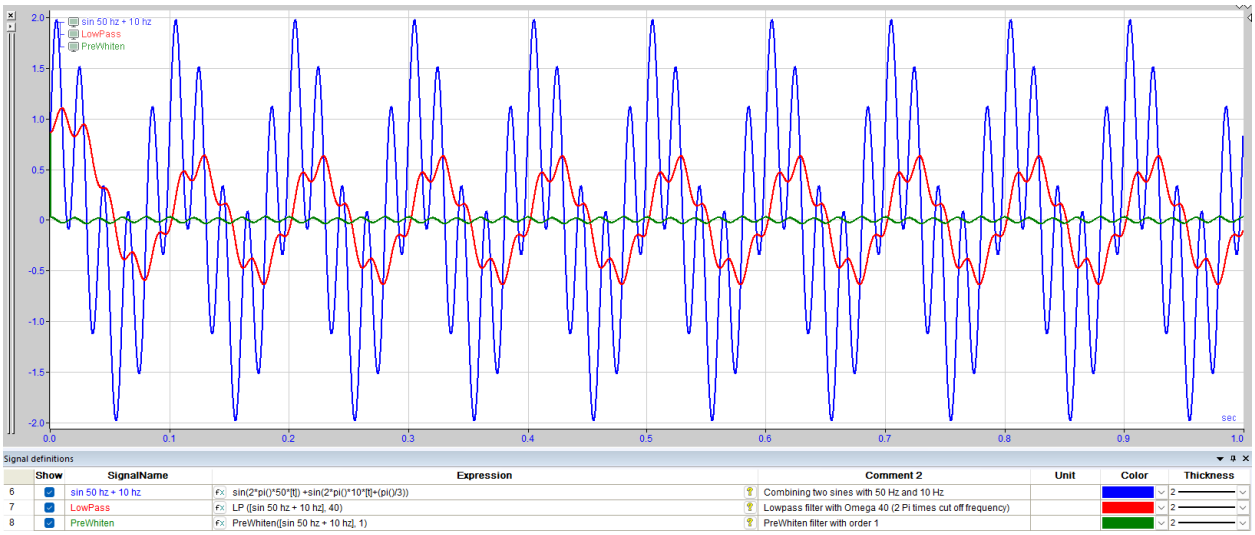
```
PreWhiten('Expression','Order')
```

Argumente

'Expression'	Signal oder Ausdruck, der gefiltert wird
'Order'	Ordnung des FIR-Filters

Beschreibung

Diese Funktion wendet einen FIR-Filter mit Koeffizienten an, die mit Hilfe der Yule-Walker-Gleichung bestimmt werden. Es handelt sich um einen Hochpass-Filter, der nur weißes Rauschen und die Impulskomponenten des Signals übriglässt.



12.3 Vold-Kalman-Filter

```
VoldKalmanFilter('Expression','Position','Speed','Bandwidth','Filterpoles','Or-
der','OTS'=FALSE)
```

Argumente

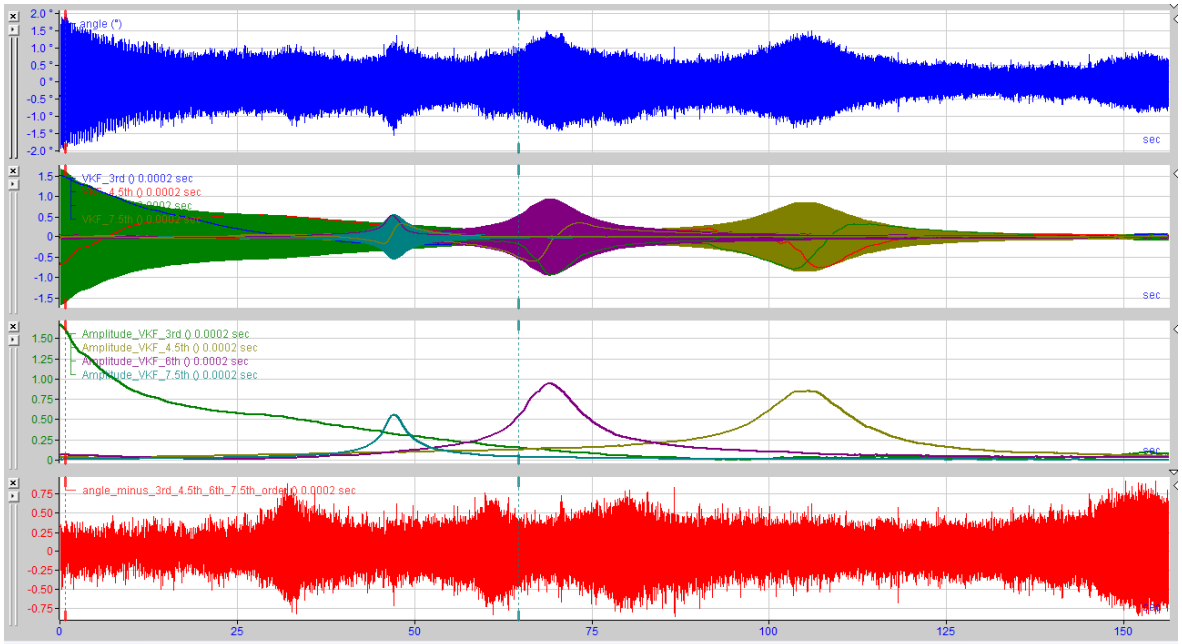
'Expression'	Signal oder Ausdruck
'Position'	Positionssignal (kann 0 sein, wenn 'Speed' angegeben)
'Speed'	Geschwindigkeitssignal (kann 0 sein, wenn 'Position' angegeben)
'Bandwidth'	Bandweite des Filters
'Filterpoles'	1,2 oder 3 (3 hat höchste Genauigkeit, aber längste Berechnungsdauer)
'Order'	Ordnung des gefilterten Signals
'OTS'	Zusätzliche Ausgabe des Order-Time-Signals

Beschreibung

Diese Funktion wendet den Vold-Kalman-Filter auf ein Signal 'Expression' an, um bestimmte Sinusschwingungen aus dem Signal zu extrahieren. Die Funktion gibt einen Vektor zurück, der aus Real- und Imaginärteil des komplexen Ergebnisses besteht. Zusätzlich kann als dritte Komponente auch die herausgefilterte Schwingung zurückgegeben werden, wenn Sie 'OTS' auf TRUE setzen.

Hinweis

Den resultierenden Vektor können Sie direkt in den Funktionen *Amplitude* und *Phase* verwenden, um die Phase und die Amplitude der Schwingung zu erhalten.



Signal definitions

	Show	SignalName	Expression	Comment 2	Unit	Color	Thickness
5	✓	angle	[angle]	Signal	°		1
6	✓	VKF_3rd	VoldKalmanFilter ([angle], 0, [1:1] / 60, 0.1, 2, 3, True())	Signal filtered with the order 3 (real and imaginary part and the filtered out signal)			1
7	✓	VKF_4.5th	VoldKalmanFilter ([angle], 0, [1:1] / 60, 0.1, 2, 4.5, True())	Signal filtered with the order 4.5 (real and imaginary part and the filtered out signal)			1
8	✓	VKF_6th	VoldKalmanFilter ([angle], 0, [1:1] / 60, 0.1, 2, 6, True())	Signal filtered with the order 6 (real and imaginary part and the filtered out signal)			1
9	✓	VKF_7.5th	VoldKalmanFilter ([angle], 0, [1:1] / 60, 0.1, 2, 7.5, True())	Signal filtered with the order 7.5 (real and imaginary part and the filtered out signal)			1
10	✓	Amplitude_VKF_3rd	Amplitude([VKF_3rd])	Amplitude of the signal filtered with order 3			2
11	✓	Amplitude_VKF_4.5th	Amplitude([VKF_4.5th])	Amplitude of the signal filtered with order 4.5			2
12	✓	Amplitude_VKF_6th	Amplitude([VKF_6th])	Amplitude of the signal filtered with order 6			2
13	✓	Amplitude_VKF_7.5th	Amplitude([VKF_7.5th])	Amplitude of the signal filtered with order 7.5			2
18	✓	angle_minus_3rd_4.5th_6	[angle] - GetRows ([VKF_3rd], 2) - GetRows ([VKF_4.5th], 2) -	Signal without the filtered out parts			1

13 Technologische Funktionen

13.1 Amplitude und Phase

Amplitude

```
Amplitude('Vector')
```

Beschreibung

Diese Funktion berechnet den Absolutwert eines Vektors 'Vector', welcher den Real- und Imaginärteil einer komplexen Zahl bzw. eines komplexen Signals beschreibt. Die Komponenten des Vektors können dabei wiederum Signale sein.

Hinweis

Die Funktion interpretiert den ersten Eintrag von 'Vector' immer als den Realteil und den zweiten als Imganinärteil. Wenn der Vektor mehr als zwei Einträge hat, werden nur die ersten beiden betrachtet.

Phase

```
Phase('Vector')
```

Beschreibung

Diese Funktion berechnet die Phase eines Vektors 'Vector', welcher den Real- und Imaginärteil einer komplexen Zahl bzw. eines komplexen Signals beschreibt. Die Komponenten des Vektors können dabei wiederum Signale sein.

13.2 ChebyCoef

```
ChebyCoef('Vector','Beginsegment','Endsegment','Order','CoverFactor'=1)
```

Argumente

'Vector'	Messwerte, die approximiert werden sollen
'Beginsegment'	Erstes zu verwendendes Vektor-Segment
'Endsegment'	Letztes zu verwendendes Vektor-Segment
'Order'	Ordnung des Chebyshev-Polynoms
'CoverFactor'	Optionaler Parameter, um den Deckungsfaktor festzulegen

Beschreibung
Die Funktion *ChebyCoef* berechnet den Koeffizienten des Chebyshev-Polynoms der Ordnung 'Order' über das Querprofil eines Vektors 'Vector'. Dabei werden nur die Einträge des Vektors zwischen den Segmenten 'Beginsegment' und 'Endsegment' berücksichtigt. Ein optionaler Deckungsfaktor 'CoverFactor' bestimmt das Verhalten an den Rändern.

Beispiel
Das Chebyshev-Polynom, benannt nach dem gleichnamigen russischen Mathematiker, ist geeignet, um das Profil eines Walzspalts mathematisch zu beschreiben. Relevant für die Walzspaltapproximation sind die Ordnungen 0 bis 6 des Polynoms, für die die Funktion die entsprechenden Koeffizienten liefert.

In der Praxis können die Koeffizienten aus den Messwerten einer Planheitsmessrolle abgeleitet werden. Dabei werden die Planheitsmesswerte der einzelnen Zonen in einem mehrdimensionalen Signal 'Vector' zusammengefasst. Jedes Array-Feld entspricht einem Segment im Sinne eines Querprofils.

13.3 CubicSpline

```
CubicSpline('Expression','X','Y')
```

Argumente

'Expression'	Hilfssignal, um die Abtastrate des Ergebnisses festzulegen
'X'	X-Koordinaten (Stützstellen), die den Spline definieren
'Y'	Y-Koordinaten, die den Spline definieren

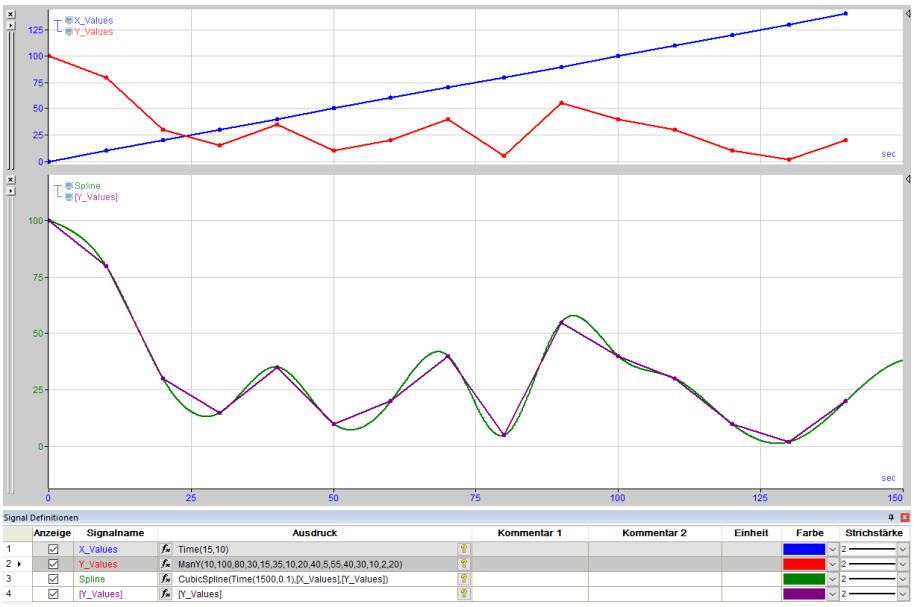
Beschreibung
Diese Funktion liefert als Ergebnis einen kubischen Spline, der an den Stützstellen 'X' mit zugehörigen Werten 'Y' ausgerichtet ist. Die Abtastrate und die Auswertepunkte des Ergebnisses werden durch 'Expression' festgelegt.

Die X-Koordinaten müssen nicht eindeutig und sortiert sein. Wenn mehrere Wertepaare mit der gleichen X-Koordinate vorhanden sind, wird für die Berechnung des Splines nur das letzte Wertepaar verwendet. Die verbleibenden Wertepaare werden automatisch nach X-Koordinaten sortiert.

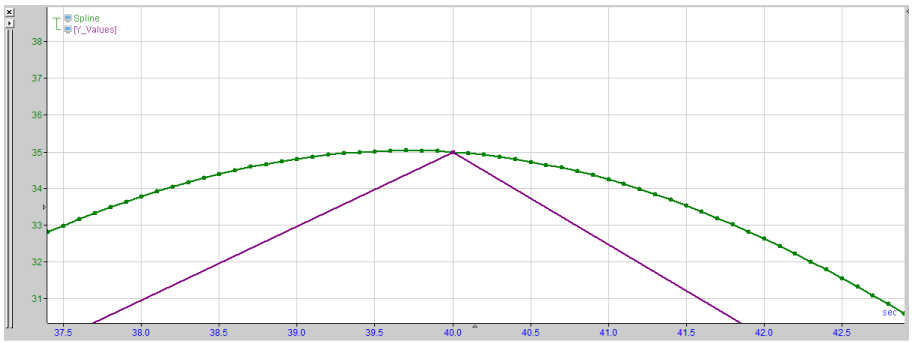
Beispiel

Für eine Reihe von Punkten liefert die Funktion als Ergebnis ein geglättetes Signal entlang des berechneten Splines. Die Funktion kann dazu verwendet werden, für ein Signal mit wenigen Samples eine Ausgleichskurve zu interpolieren:

Eine Kurv weist nur 17 Samples über einen Zeitraum von 5000 s auf (Y-Werte, grüne Kurve). Die dazu passenden X-Koordinaten – ebenfalls nur 17 Werte – sind als blaue Kurve eingezeichnet. Die Ausgleichskurve soll als geglättetes Signal eine deutlich höhere Auflösung erhalten (mehr Samples). Daher wird der Funktion *CubicSpline* als Parameter 'Expression' eine lineare Funktion mit 5000 Samples im Abstand von 1 s übergeben.



In der stark gezoomten Ansicht sind die berechneten Samples der Ausgleichskurve zu erkennen (grün). Die ursprünglichen X-Y-Koordinaten bilden die Knoten des Splines (violett).



13.4 Exponential

```
Exponential('Coefs','X')
```

Argumente

'Coefs'	Vektor mit Koeffizienten, z. B. das Ergebnis von *LSQExponentialCoef*
'X'	X-Koordinaten (Abtastpunkte), an denen die Exponentialfunktion ausgewertet wird

Beschreibung
Diese Funktion berechnet den Wert einer Exponentialfunktion a*exp(b*X) für jedes Sample von 'X' mit den Koeffizienten a und b, die als Vektor 'Coefs' angegeben sind. Die Funktion kann die Ergebnisse von *LSQExponentialCoef* visualisieren.

13.5 LSQExponentialCoef

```
LSQExponentialCoef('X','Y')
```

Argumente

'X'	X-Koordinaten (Abtastpunkte), die die interpolierende Exponentialfunktion definieren
'Y'	Y-Werte, die die interpolierende Exponentialfunktion definieren

Beschreibung
Diese Funktion berechnet die Koeffizienten eines interpolierenden Polynoms a*exp(b*x) für die durch die X-Koordinaten 'X' gegebene Funktion und die entsprechenden Funktionswerte 'Y' nach der Methode der kleinsten Quadrate. Das Ergebnis der Funktion ist ein Vektor, der die Koeffizienten enthält.

13.6 LSQPolyCoef

```
LSQPolyCoef('X','Y','Degree','PolynomialType'=0)
```

Argumente

'X'	X-Koordinaten (Stützstellen), die das Interpolationspolynom definieren
'Y'	Y-Koordinaten, die das Interpolationspolynom definieren
'Degree'	Polynomgrad (0 = Mittelwert, 1 = linear, 2 = quadratisch, 3 = kubisch, etc.)
'PolynomialType'	Definiert die Basispolynome, die verwendet werden 0 = Lagrange (Standard), 1 = Chebyshev I, 2 = Chebyshev II, 3 = Legendre

Beschreibung

Diese Funktion berechnet die Koeffizienten eines Interpolationspolynoms vom Grad 'Degree' für Wertepaare 'X' und 'Y' nach der Methode der kleinsten Quadrate.

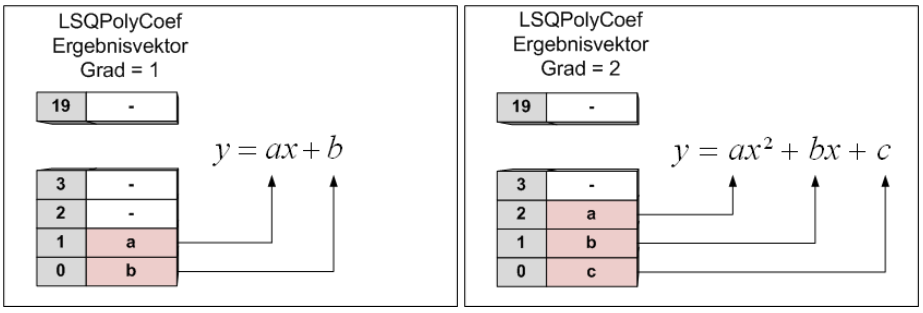
Das Ergebnis der Funktion ist ein Vektor (mehrdimensionales Signal, Array), in dem die Koeffizienten enthalten sind. Das Array-Feld mit dem Index 0 enthält den konstanten Anteil oder Offset des Polynoms. Die Koeffizienten werden mit aufsteigendem Grad entsprechend in Array-Felder mit aufsteigendem Index geschrieben.

Die Funktion *Polynomial* kann das Polynom auswerten.

Mit dem optionalen Parameter 'PolynomialType' können Sie verschiedene Basispolynome verwenden. Unterstützt werden die Typen Lagrange (Standard), Chebyshev I, Chebyshev II und Legendre.

Beispiel

Eine quadratische Approximation liefert ein Polynom der Form $y = ax^2 + bx + c$, d. h. das Ergebnis ist ein Vektor mit insgesamt 3 Koeffizienten.



Die Funktion basiert prinzipiell auf einer X-Y-Beziehung, d. h. die Operanden X und Y können auch zwei unterschiedliche Messsignale sein. Wenn Sie nur eine Regressionskurve für ein Signal über die Zeit berechnen wollen, dann müssen die Zeitwerte auch in Form eines Signals vorgegeben werden, z. B. mithilfe der Funktion *XValues* ([Signal]). Dieses Zeitsignal, dessen Y-Werte identisch mit der Zeit entlang der X-Achse sind, können Sie dann als Operand 'X' in der Funktion *LSQPolyCoef* verwenden.

13.7 Polynomial

```
Polynomial('Coefs','X','PolynomialType'=0)
```

Argumente

'Coefs'	Vektor mit Koeffizienten, z. B. als Ergebnis von *LSQPolyCoef*
'X'	X-Koordinaten (Stützstellen), an denen das Polynom ausgewertet wird
'PolynomialType'	Definiert die Basispolynome, die verwendet werden; 0 = Lagrange (Standard), 1 = Chebyshev I, 2 = Chebyshev II, 3 = Legendre

Beschreibung

Diese Funktion berechnet für jedes Sample von 'X' den Polynomwert auf Basis eines Koeffizientenvektors 'Coefs'. Die Funktion stellt Regressionsgeraden oder Ausgleichskurven dar, deren Koeffizienten zuvor mit der Funktion *LSQPolyCoef* berechnet wurden.

Mit dem optionalen Parameter 'PolynomialType' können Sie verschiedene Basispolynome verwenden. Unterstützt werden die Typen Lagrange (Standard), Chebyshev I, Chebyshev II und Legendre.

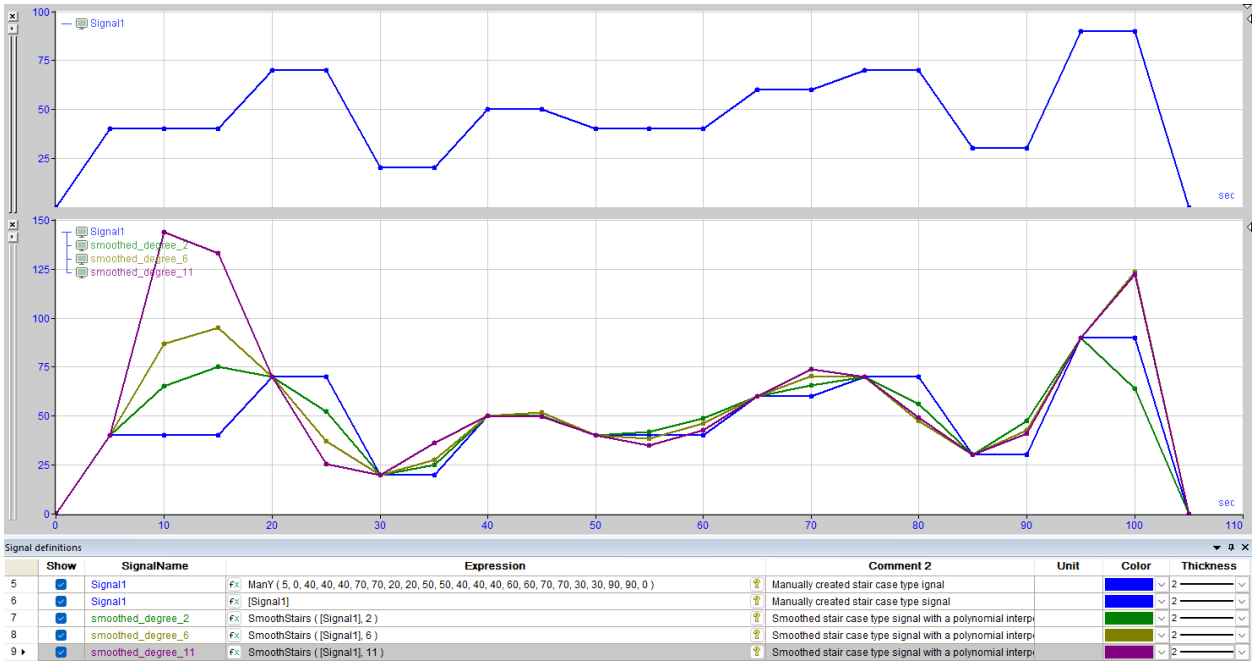
13.8 SmoothStairs

```
SmoothStairs('Expression','N')
```

'Expression'	Eingangssignal in Treppenstufenform
'N'	Grad des Interpolationspolynoms

Beschreibung

Diese Funktion dient zum Glätten eines Treppenstufensignals. Hierzu wird ein Interpolationspolynom vom Grad 'N' gebildet, dessen Stützstellen jeweils die ersten Werte einer Treppenstufe sind.



14 Spektralanalyse (FFT-Operationen)

ibaAnalyzer verfügt über die Möglichkeit der Spektralanalyse in Form der Fast Fourier Transformation (FFT). Mit den zur Verfügung stehenden FFT-Operationen kann ein zeit- oder längenbasiertes Signal nicht nur im FFT-Modus angezeigt werden, sondern als berechneter Ausdruck zur Verfügung gestellt und für weiterführende Analysen verwendet werden.

Für die meisten der hier beschriebenen Funktionen können Sie entweder einen Amplituden- oder einen Leistungstrend anzeigen. Dies wird durch die Endungen "Ampl" bzw. "Power" in den Funktionsnamen gekennzeichnet.

14.1 AWeighting und DbScale

AWeighting

```
AWeighting('Spectrum','type'=0)
```

Argumente

'Spectrum'	Spektrum
'type'	Angabe des Typs

Beschreibung
Diese Funktion gewichtet ein Spektrum nach der sogenannten A-Bewertung. Dies ist ein Bewertungsfilter, welcher dem menschlichen Gehör entspricht.

Beispiel
Nach dem Anwenden dieser Funktion kann ein Spektrum bezüglich der wahrnehmbaren Geräuschemission beurteilt werden. Die Bewertung für Schalldruckpegel hat folgende Typen:

- A: Pegel von ca. 20 phon bis 40 phon
- B: Pegel von ca. 50 phon bis 70 phon
- C: Pegel von ca. 80 phon bis 90 phon
- D: für hohe Schalldrücke

DbScale

```
DbScale('Spectrum','ref'=1)
```

Argumente

'Spectrum'	Spektrum, welches logarithmisch skaliert wird
'ref'	Optional

Beschreibung
Diese Funktion stellt eine logarithmische Skalierung in dB für ein Signalspektrum zur Verfügung. Für ein brauchbares Ergebnis muss als Eingang die Amplitude eines Spektrums gegeben sein.

Tipp

Die folgFnden Funktionen berechnen eine solche Amplitude: *FftAmpl, FftInTimeAmpl, FftOrderAnalysisAmpl*

14.2 FftAmpl und FftPower

```
z. B. FftAmpl('Expression','Samples','Window'=0,'SuppressDC'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Fourier Transformierte berechnet wird
'Samples'	Anzahl der zu berücksichtigenden Messwerte und dadurch implizite Festlegung des verwendeten Zeit- bzw. Längenintervalls, abhängig von der Abtastrate
'Window'	Fenstertyp: 0 = Rechteck 1 = Bartlett 2 = Blackman 3 = Hamming 4 = Hanning 5 = Blackman-Harris 6 = Flat top
'SuppressDC'	Gleichanteilunterdrückung

Beschreibung

Diese Funktionen berechnen die Amplitude bzw. Leistung der Fourier-Transformierten des Signals. Der verwendete Zeitabschnitt wird durch Runden der verwendeten Messpunkte 'Samples' auf eine Zweierpotenz bestimmt.

Hinweis

Der Parameter 'Samples' wird aufgerundet. Mindestens 128 Messpunkte müssen verwendet werden.

Über den Parameter 'Window' können Sie den Fenstertyp steuern, der für die Berechnung verwendet wird. Optional können Sie die Gleichanteilunterdrückung mit dem Parameter 'SuppressDC' aktivieren.

14.3 FftComplex

```
FftComplex('Expression','Inverse'=FALSE,'Normalize'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Fourier-Transformation berechnet wird
'Inverse'	Optionaler Parameter, um eine inverse Fourier-Transformation zu ermöglichen
'Normalize'	Optionaler Parameter, um eine Normierung zu wählen

Beschreibung

Diese Funktion führt eine Fourier-Transformation für ein komplexes Signal über den gesamten Ausdruck aus und liefert als Ergebnis einen Vektor mit Realteil und Imaginärteil der Fourier-Transformierten. Das Eingangssignal kann sowohl ein einzelnes Signal oder ein Vektor bestehend aus Real- und Imaginärteil sein. Dabei wird ein Rechteck-Fenster für die Berechnung verwendet.

Wenn Sie den Parameter 'Inverse' auf True() bzw. 1 setzen, wird eine inverse Fourier-Transformation berechnet. In diesem Fall erwartet die Funktion ein Eingangssignal, das entweder frequenzbasiert oder 1/Länge-basiert ist. Das Ergebnis der Operation ist dann entsprechend ein zeit- oder längenbasiertes Signal.

Für den Parameter 'Normalize' sind folgende Werte zulässig:

- **0**: Es wird keine Normierung durchgeführt.
- **1**: Das Ergebnis wird durch die Zahl der Samples dividiert. Bei einer inversen Transformation wird das Ergebnis nicht verändert.
- **2**: Das Ergebnis wird durch die Quadratwurzel der Anzahl Samples dividiert. Dies gilt sowohl für eine normale als auch eine inverse Transformation
- **Andere Werte**: Funktion wie mit Wert 1.

Die Anzahl der Frequenz-Samples wird durch die Anzahl der Samples des Eingangssignals bestimmt. Wenn N gerade ist, werden N/2+1 Frequenzpunkte berechnet, von denen der erste (Gleichanteil) und der letzte Punkt rein reell sind. Wenn N ungerade ist, dann werden (N+1)/2 Frequenzpunkte berechnet, für die der Gleichanteil rein reell ist.

14.4 FftInTimeAmpl und FftInTimePower

```
z. B. FftInTimeAmpl('Expression','Samples','#Freq','Min frequency'=0,'Max
frequency'=Sampling/2,'Window'=0,'Overlap'=0,'SuppressDC'=FALSE,'ZeroPad'=TRUE)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die FFT berechnet wird
'Time'	Festlegung der verwendeten Zeit- bzw. Längenintervalle. Hier wird auf ein Intervall gerundet, welches 2^N Samples enthält.
'#Freq'	Anzahl der angezeigten Frequenzen
'Min frequency'	Minimale Frequenz
'Max frequency'	Maximale Frequenz
'Window'	Fenstertyp: 0 = Rechteck 1 = Bartlett 2 = Blackman 3 = Hamming 4 = Hanning 5 = Blackman-Harris 6 = Flat top
'Overlap'	Überlappungsfaktor
'SuppressDC'	Gleichanteilunterdrückung
'ZeroPad'	Ergänzen von Nullen

Beschreibung
Diese Funktionen berechnen Amplitude bzw. Leistung der Fourier-Transformierten von 'Expression' für Abschnitte mit jeweils 2^N Messwerten. Dabei wird N bestimmt, indem das Produkt 'Time' × Abtastfrequenz auf eine Zweierpotenz gerundet wird.

Das Ergebnis ist ein Vektor, der '#Freq' gleichmäßig aufgeteilte Frequenzen zwischen 'Min frequency' und 'Max frequency' pro Abschnitt enthält. Über den Parameter 'Window' können Sie den Fenstertyp steuern, der für die Berechnung verwendet wird.

Der Überlappungsfaktor bestimmt die Überlappung der Zeitsegmente und kann zwischen 0 (keine Überlappung) und 1 (vollständige Überlappung) liegen. Optional können Sie die Gleichanteilunterdrückung mit dem Parameter 'SuppressDC' aktivieren.

Wenn der Parameter 'ZeroPad' auf 1 oder TRUE() gesetzt ist, dann wird vor der Berechnung der FFT das letzte Fenster mit Nullen aufgefüllt. Wenn 'ZeroPad'=FALSE, wird das letzte Fenster verworfen.

Beispiel
Mithilfe der *FftInTime*-Funktion lassen sich Frequenzschwankungen über die Zeit darstellen. Dazu kann der resultierende Vektor in einer 2D-Ansicht angezeigt werden.

14.5 FftOrderAnalysisAmpl und FftOrderAnalysisPower

```
z. B. FftOrderAnalysisAmpl('Expression','Time','Freq','MinOrder'=0,'MaxOrder',
'Order subdivision'=1,'Window'=0,'Overlap'=0,'SuppressDC'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Ordnungsanalyse durchgeführt wird
'Time'	Festlegung der verwendeten Zeit- bzw. Längenintervalle
'Freq'	Grundfrequenz für die Ordnungsanalyse (Drehfrequenz)
'MinOrder'	Minimal angezeigte Ordnung
'MaxOrder'	Maximal angezeigte Ordnung
'Order subdivision'	Rasterbreite zwischen den ganzzahligen Ordnungen
'Window'	Fenstertyp: 0 = Rechteck 1 = Bartlett 2 = Blackman 3 = Hamming 4 = Hanning 5 = Blackman-Harris 6 = Flat top
'Overlap'	Überlappungsfaktor
'SuppressDC'	Gleichanteilunterdrückung

Beschreibung

Diese Funktion berechnet die Ordnungen (d. h. Vielfache einer Grundfrequenz 'Freq') für ein Signal und liefert als Ergebnis einen Vektor mit den Ordnungen zwischen 'MinOrder' und 'MaxOrder'. Die Anzahl der Datenpunkte pro Ordnung wird durch den Parameter 'Order subdivision' festgelegt.

Über den Parameter 'Window' können Sie den Fenstertyp steuern, der für die Berechnung verwendet wird. Der Überlappungsfaktor bestimmt die Überlappung der Zeitsegmente und kann zwischen 0 (keine Überlappung) und 1 (vollständige Überlappung) liegen. Optional können Sie die Gleichanteilunterdrückung mit dem Parameter 'SuppressDC' aktivieren.

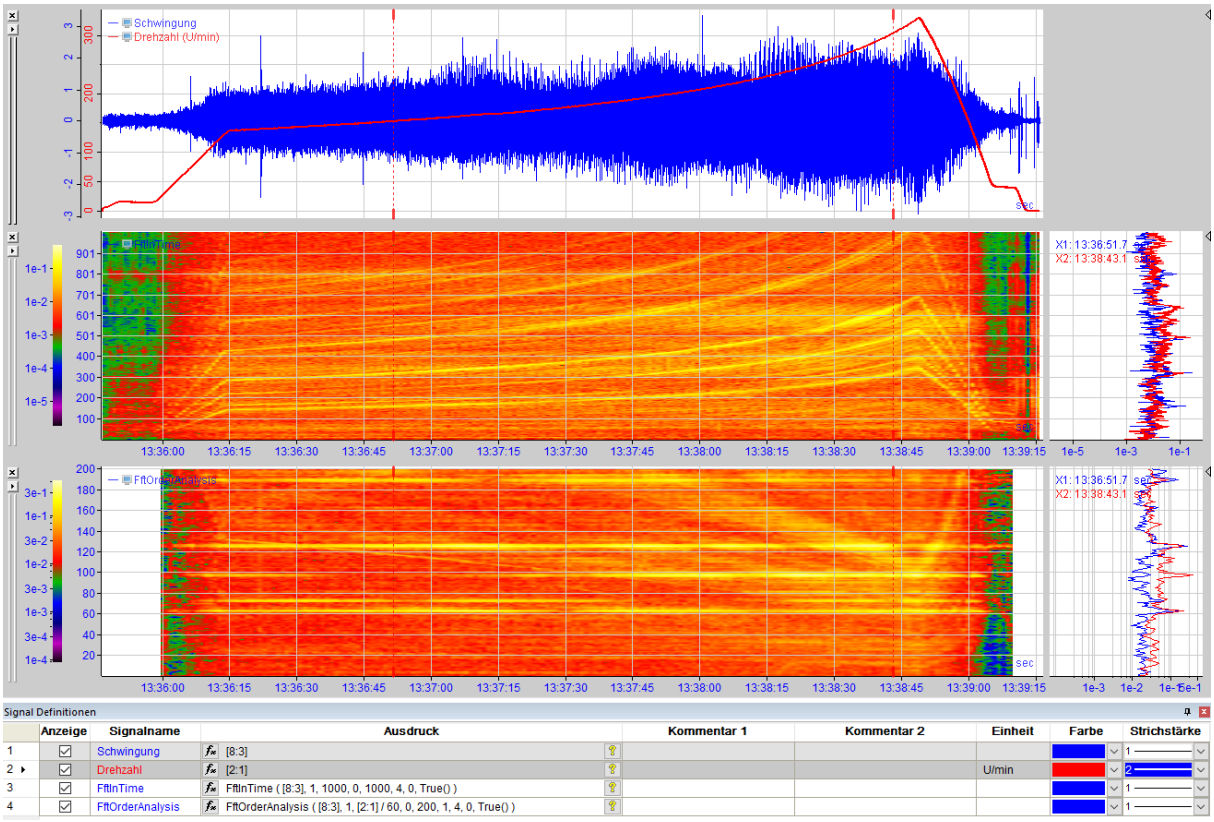
Im Gegensatz zur *FftInTime*-Funktion werden auf der Y-Achse nicht mehr die Zeit/Frequenz dargestellt, sondern die Drehzahlfrequenz und deren Vielfache, d. h. die Ordnungen. Die Frequenzachse wird dazu entsprechend der momentanen Drehzahl verzerrt, sodass die Ordnungen jetzt nicht mehr als Kurve, sondern als gerade Linien dargestellt werden. Je nach Funktion wird entweder ein Amplitudentrend (*FftOrderAnalysisAmpl*) oder ein Leistungstrend berechnet (*FftOrderAnalysisPower*).

Hinweis

Die Funktion liefert keine Resultate, wenn die Anzahl der Signalpunkte pro Umdrehung mehr als doppelt so hoch wie der gewählte Parameter 'Time' ist.

Beispiel

Mit der Funktion *FftOrderAnalysis* können Sie die Berechnung der Ordnungsanalyse durchführen. Frequenzen, die der Motordrehzahl oder deren Vielfachen entsprechen, werden als Ordnungen bezeichnet. Die erste Ordnung entspricht der Frequenz der Motordrehzahl, die zweite Ordnung entspricht der Frequenz der ersten Ordnung multipliziert mit dem Faktor 2, usw. Bei der Ordnungsanalyse wird der Pegel oder der Pegelverlauf dieser Ordnungen berechnet.



Zwischen den einzelnen Signalpunkten erfolgt eine Interpolation zur Berechnung der *FftOrderAnalysis*.

14.6 FftPeaksInTimeAmpl und FftPeaksInTimePower

```
z. B. FftPeaksInTimeAmpl('Expression','Time','#Peaks','Min frequency'=0,'Max
frequency'=Sampling/2,'Window'=0,'Overlap'=0,'SuppressDC'=FALSE,'ZeroPad'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Frequenzspitzen ausgewertet werden
'Time'	Festlegung der verwendeten Zeit- bzw. Längenintervalle. Hier wird auf ein Intervall gerundet welches 2^N Samples enthält.
'#Peaks'	Anzahl der angezeigten Spitzen
'Min frequency'	Optionaler Parameter für die minimal berücksichtigte Frequenz, d. h. Spitzen bei niedrigerer Frequenz werden nicht angezeigt
'Max frequency'	Optionaler Parameter für die maximal berücksichtigte Frequenz, d. h. Spitzen bei höherer Frequenz werden nicht angezeigt

'Window'	Fenstertyp: 0 = Rechteck 1 = Bartlett 2 = Blackman 3 = Hamming 4 = Hanning 5 = Blackman-Harris 6 = Flat top
'Overlap'	Überlappungsfaktor
'SuppressDC'	Gleichanteilunterdrückung
'ZeroPad'	Ergänzen von Nullen

Beschreibung
Diese Funktion dient der Berechnung von Frequenzspitzen über gleitende Zeitintervalle, die durch den Parameter 'Time' festgelegt werden. Dabei werden die '#Peaks' berechnet. Das sind die höchsten Spitzen zwischen den Frequenzen 'Min frequency' und 'Max frequency'. Der Zeitverlauf der Frequenz und Spitzenwertepaare wird als Vektor zurückgegeben.

Dabei sind die Einträge nach folgendem Muster sortiert:

- Index 0: Frequenz mit der höchsten Spitze
- Index 1: Amplitude/Leistungen der höchsten Spitze
- Index 2: Frequenz mit der zweithöchsten Spitze
- Index 3: Amplitude/Leistungen der zweithöchsten Spitze
- Etc.

Tipp

Um die gewünschten Werte aus dem Ergebnisarray auszulesen, können Sie die Funktion *GetRows* benutzen.

Über den Parameter 'Window' können Sie den Fenstertyp steuern, der für die Berechnung verwendet wird. Der Überlappungsfaktor bestimmt die Überlappung der Zeitsegmente und kann zwischen 0 (keine Überlappung) und 1 (vollständige Überlappung) liegen. Optional können Sie die Gleichanteilunterdrückung mit dem Parameter 'SuppressDC' aktivieren.

Wenn der Parameter 'ZeroPad' auf 1 oder TRUE() gesetzt ist, dann wird vor der Berechnung der FFT das letzte Fenster mit Nullen aufgefüllt. Wenn 'ZeroPad'=FALSE, wird das letzte Fenster verworfen.

14.7 FftReal und FftRealInverse

```
z. B. FftReal('Expression','Normalize'=FALSE)
```

Argumente

'Expression'	Signal oder Ausdruck, für den die Fourier-Transformierte berechnet wird
'Normalize'	Optionaler Parameter, um die Normierung zu aktivieren

Beschreibung FftReal
Diese Funktion führt eine Fourier-Transformation für ein reelles Signal über den gesamten Ausdruck aus und liefert als Ergebnis einen Vektor mit Realteil und Imaginärteil der Fourier-Transformierten. Dabei wird ein Rechteck-Fenster verwendet.

Wenn Sie den Parameter 'Normalize' auf True() bzw. 1 setzen, dann wird eine Normierung durchgeführt. Wenn die Anzahl der Samples (N) des Signals ungerade ist, werden alle Frequenzwerte außer dem Gleichanteil durch N/2 dividiert. Wenn N gerade ist, werden alle Frequenzwerte außer dem Gleichanteil und letztem Wert durch N/2 dividiert.

Die Anzahl der Frequenz-Samples wird durch die Anzahl der Samples des Eingangssignals bestimmt. Wenn N gerade ist, werden N/2+1 Frequenzpunkte berechnet, von denen der erste (Gleichanteil) und der letzte Punkt rein reell sind. Wenn N ungerade ist, dann werden (N+1)/2 Frequenzpunkte berechnet, für die der Gleichanteil rein reell ist.

Beschreibung FftRealInverse
Diese Funktion berechnet die inverse Fourier-Transformation, wie mit *FftReal* erstellt. Dabei ist das Ergebnis reell und das Eingangssignal muss dementsprechend ein Vektor sein, der aus Realteil und Imaginärteil besteht. Ansonsten funktioniert die Funktion genauso wie *FftReal*.

14.8 IntSpectrum

```
IntSpectrum('Spectrum')
```

Beschreibung
Diese Funktion integriert ein gegebenes Spektrum. So kann beispielsweise aus der Frequenz eines Beschleunigungssensors die Vibrationsgeschwindigkeit berechnet werden.

15 Text-Funktionen

15.1 InfoFieldText, ChannelInfoFieldText und ModuleInfoFieldText

Diese Funktionen erlauben es, Informationen aus einem Info-Feld einer Messdatei, eines Signals oder eines Moduls, als Textsignal zur Verfügung zu stellen.

Argumente

'Index'	Index der Messdatei, des Signals oder Moduls
'InfoField'	Infofeld, das ausgelesen wird; in Anführungszeichen setzen.
'Begin'	Optional: Erstes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der komplette Inhalt ausgelesen.
'End'	Optional: Letztes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der Inhalt von 'Begin' bis zum letzten Zeichen ausgelesen.

InfoFieldText

```
InfoFieldText('FileIndex',"'InfoField'",'Begin'=0,'End'=Text end)
```

Beschreibung

Diese Funktion gibt den Inhalt eines Infofeldes als Textsignal aus.

Tipp

Wenn Sie im Signalbaum einen Doppelklick auf das gewünschte Infofeld machen, dann fügt *ibaAnalyzer* die entsprechende Funktion automatisch als neues Signal in die Signaltabelle ein. Anschließend brauchen Sie bei Bedarf nur noch den Signalnamen und Anfang oder Ende anpassen.

Wenn Sie den Inhalt eines Infofeldes als Zahlenwert auslesen wollen, verwenden Sie die Funktion *InfoField*.

ChannelInfoFieldText

```
ChannelInfoFieldText('ChannelIndex',"'InfoField'",'Begin'=0,'End'=Text end)
```

Beschreibung

Diese Funktion gibt den Inhalt des Infofeldes eines Signals als Text aus.

ModuleInfoFieldText

```
ModuleInfoFieldText('FileIndex','ModuleIndex',"'InfoField'",'Begin'=0,
'End'=Text end)
```

Hinweis

Bei dieser Funktion müssen Sie 2 Indizes angeben: Den Index der Messdatei als erstes Argument und den Index des Moduls als zweites. Alle anderen Argumente bleiben gleich.

Argumente

'FileIndex'	Index der Messdatei, zu der das Modul gehört
'ModuleIndex'	Index des Moduls
'InfoField'	Infofeld, das aus dem Modul ausgelesen wird; in Anführungszeichen setzen.
'Begin'	Optional: Erstes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der komplette Inhalt ausgelesen.
'End'	Optional: Letztes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der Inhalt von 'Begin' bis zum letzten Zeichen ausgelesen.

Beschreibung

Diese Funktion arbeitet wie die Funktionen *InfoFieldText* und *ChannelInfoFieldText*, jedoch bezieht sie sich auf die Infofelder eines Moduls und nicht der Messdatei oder Signals. Die Funktion liefert als Ergebnis einen Textsignal mit dem Inhalt des spezifizierten Infofelds.

15.2 CharValue

```
CharValue('Text','CharNumber'=0)
```

Argumente

'Text'	Textsignal oder Eingabetext (Eingabe in Anführungszeichen)
'CharNumber'	Nullbasierte Position des zu bewertenden Zeichens.

Beschreibung

Diese Funktion liefert als Ergebnis den ASCII-Wert des Zeichens an der Position 'CharNumber' in 'Text'. Standardmäßig wird das erste Zeichen verwendet, das die 'CharNumber' Null hat.

15.3 CompareText

```
CompareText('Text1','Text2','CaseSensitive'=TRUE,'NumericalSort'=FALSE)
```

Argumente

'Text1/2'	Textsignale oder Eingabetexte, die verglichen werden (Eingabe in Anführungszeichen)
'CaseSensitive'	Optionaler Parameter, um beim Vergleich die Groß- und Kleinschreibung zu berücksichtigen
'NumericalSort'	Optionaler Parameter, um festzulegen, ob Ziffern jeweils einzeln verglichen werden oder Ziffernkombinationen als Zahlen verglichen werden

Beschreibung
Mit dieser Funktion können Sie Textinformationen lexikographisch miteinander vergleichen. Die Funktion arbeitet sowohl mit Inhalten von Textsignalen als auch mit Klartext, den Sie mit Anführungszeichen direkt in die Signaldefinition eingeben können.

Vergleich und Ergebnisse:

- Ergebnis -1: Der erste Text steht lexikographisch vor dem zweiten Text.
- Ergebnis 0: Beide Texte haben identische Informationen.
- Ergebnis +1: Der erste Text steht lexikographisch nach dem zweiten Text.

Beispiele
Die folgende Tabelle zeigt die allgemeine Funktionsweise und den Einfluss des Parameters 'CaseSensitive':

Text1	Text2	Ergebnis		Anmerkung
		CompareText ("Text1","Text2",0)	CompareText ("Text1","Text2",1)	
1234 abcd	1234 abcd	0	0	Text1 = Text2
1234 abcd	1234 bcde	-1	-1	Text1 < Text2 "a" steht vor "b"
1234 Abcd	1234 abcd	0	-1	Text1 = Text2 (Groß-/Kleinschreibung nicht berücksichtigt) Text1 < Text2 (Groß-/Kleinschreibung berücksichtigt) "A" steht vor "a"
12340 abcd	1234 abcd	1	1	Text1 > Text2 "0" steht nach " " (Leerzeichen)
1234 0abcd	1234 abcd	-1	-1	Text1 < Text2 "0" steht vor "a"
12034 abcd	1234 abcd	-1	-1	Text1 < Text2 "0" steht vor "3"
1234 abcd	1y34 abcd	-1	-1	Text1 < Text2 "2" steht vor "y"
1z34 abcd	1Y34 abcd	1	1	Text1 > Text2 "z" steht nach "Y"

Den Einfluss des Parameters 'NumericalSort' zeigt die folgende Tabelle:

Text1	Text2	Ergebnis	
		CompareText ("Text1","Text2",0,0)	CompareText ("Text1","Text2",0,1)
12034 abcd	1234 abcd	-1 Text1 < Text2 "0" steht vor "3"	1 Text1 > Text2 "12034" steht nach "1234"

15.4 ConcatText

```
ConcatText('t1','t2',…)
```

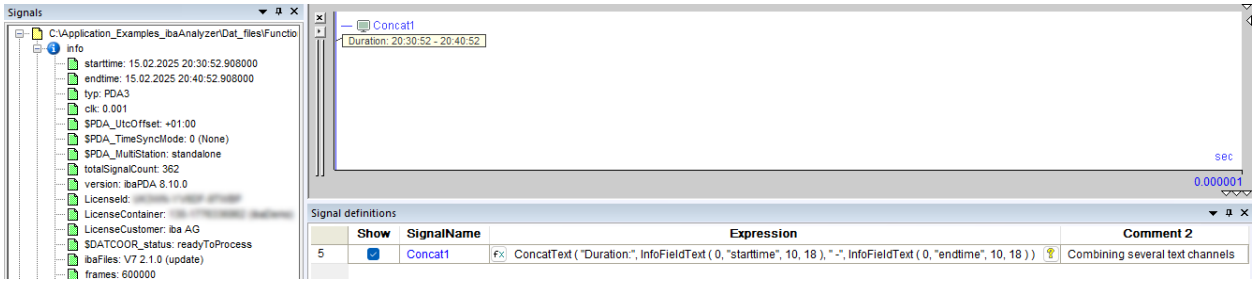
Beschreibung

Mit dieser Funktion können mehrere Textsignale zusammengefügt werden. Dabei sind auch numerische Signale zulässig. Diese werden automatisch in Text umgewandelt.

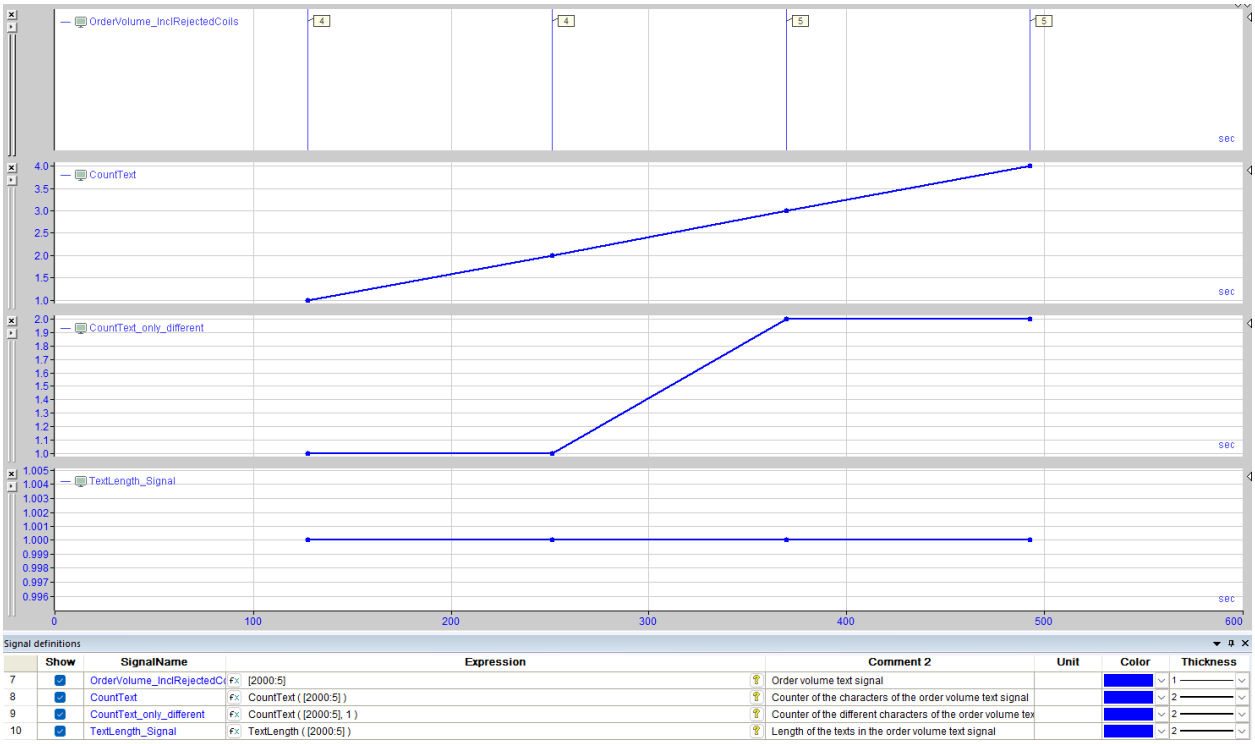
Falls die X-Positionen der einzelnen Signale nicht zueinander passen, wird für jede vorhandene X-Position ein neuer Eintrag erstellt und der fehlende Eintrag durch den linken Nachbarn ersetzt, falls vorhanden.

Beispiel

Für aneinander gereihte Signale ist die File ID als Textsignal vorhanden. Für die Endprodukte gibt es einen Produkt-Counter. Das Ergebnis bringt beide Informationen zusammen.



15.5 CountText und TextLength



CountText

```
CountText('Text','CountOnlyDifferent'=0,'Reset'=0)
```

Argumente

'Text'	Textsignal oder Eingabetext (Eingabe in Anführungszeichen)
'CountOnlyDifferent'	Optionaler Parameter zum Zählen von Texten, die sich vom vorherigen Textausdruck unterscheiden
'Reset'	Optionaler Parameter zum Zurücksetzen des Zählers

Beschreibung

Diese Funktion liefert als Ergebnis die Anzahl der Texte innerhalb eines Textsignals 'Text' zurück. Wenn der Parameter 'CountOnlyDifferent' auf TRUE gesetzt ist, werden nur Texte gezählt, die vom vorherigen Textausdruck abweichen. Mit dem Digitalsignal 'Reset' kann der Zähler auf Null zurückgesetzt werden.

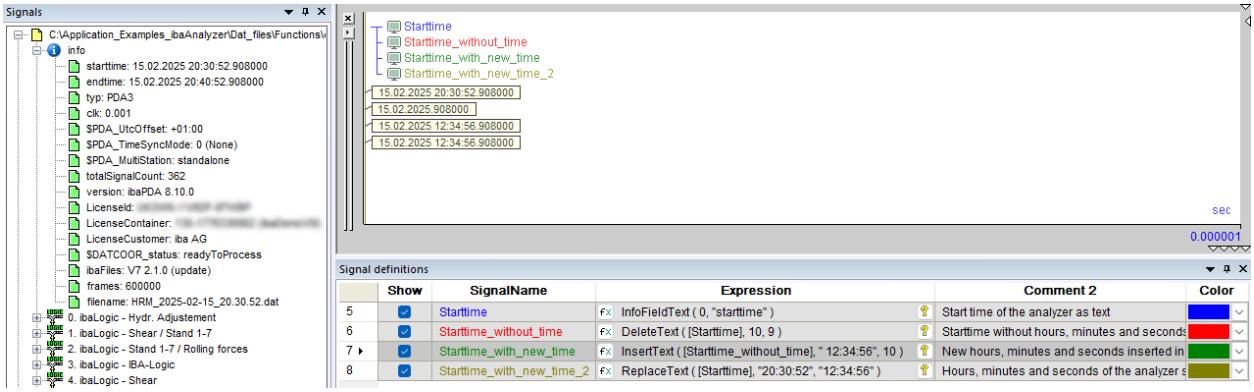
TextLength

```
TextLength('Text')
```

Beschreibung

Diese Funktion liefert als Ergebnis die Anzahl der Zeichen in 'Text'.

15.6 DeleteText, InsertText und ReplaceText



DeleteText

```
DeleteText('Text','StartPos','Length')
```

Argumente

'Text'	Textsignal oder Eingabetext (Eingabe in Anführungszeichen)
'StartPos'	Nullbasierter Index des ersten gelöschten Zeichens
'Length'	Anzahl der zu löschenden Zeichen

Beschreibung

Diese Funktion löscht eine Anzahl von 'Length' Zeichen aus dem 'Text'. Das erste Zeichen, das gelöscht wird, befindet sich an der Position 'StartPos' im Text.

InsertText

```
InsertText('Text1','Text2','Pos')
```

Argumente

'Text1'	Originaltext
'Text2'	Text, der in den Originaltext eingefügt wird
'Pos'	Nullbasierte Zeichenposition, an der 'Text2' eingefügt wird

Beschreibung

Diese Funktion fügt 'Text2' in 'Text1' an der nullbasierten Zeichenposition 'Pos' ein. Wenn 'Pos' kleiner oder gleich Null ist, wird 'Text2' vor 'Text1' gestellt. Wenn 'Pos' größer oder gleich der Länge von 'Text1' ist, wird 'Text2' an 'Text1' angehängt.

ReplaceText

```
ReplaceText('Text','SearchText','ReplaceText')
```

Argumente

'Text'	Textsignal oder Eingabetext (Eingabe in Anführungszeichen)
'SearchText'	Text, der innerhalb von 'Text' ersetzt werden soll
'ReplaceText'	Wenn 'SearchText' gefunden wird, wird dieser durch 'ReplaceText' ersetzt

Beschreibung

Diese Funktion ersetzt alle Vorkommen von 'SearchText' innerhalb von 'Text' durch 'ReplaceText'.

15.7 Merge

```
Merge('t1','t2',…)
```

Beschreibung
Diese Funktion kombiniert zwei oder mehr Textsignale zu einem, indem die Einträge aller Signale in einem gemeinsamen Signalstreifen dargestellt werden. Wenn zwei Einträge verschiedener Signale denselben X-Wert haben, wird nur der Text des zuerst aufgeführten Signals angezeigt.

15.8 MidText und FindText

Signal definitions

	Show	SignalName	Expression	Comment 2	Color	Thickness
5	☑	Starttime	InfoFieldText (0, "starttime")	Starttime of the analyzer		2
6	☑	FirstBlank	FindText ([Starttime], " ", 0, 1)	Position of the first blank		2
7	☑	FirstColon	FindText ([Starttime], ":", 0, 1)	Position of the first colon		2
8	☑	SecondColon	FindText ([Starttime], ":", 0, 2)	Position of the second colon		2
9	☑	hours	MidText ([Starttime], [FirstBlank]+1, 2)	Two charecters from the first blank (hours)		1
10	☑	minutes	MidText ([Starttime], [FirstColon]+1, 2)	Two characters from the first colon (minutes)		1
11	☑	seconds	MidText ([Starttime], [SecondColon]+1, 2)	Two characters from the second colon (secor		1

MidText

```
MidText('Text','StartPos','Length')
```

Argumente

'Text'	Textsignal oder Eingabetext (Eingabe in Anführungszeichen)
'StartPos'	Nullbasierter Index des ersten zu extrahierenden Zeichens aus dem Text
'Length'	Anzahl der Zeichen, die aus dem Text extrahiert werden sollen

Beschreibung
Diese Funktion liefert als Ergebnis eine Anzahl von Zeichen ('Length') aus dem 'Text'. Die resultierende Zeichenfolge beginnt mit dem Zeichen an der nullbasierten Position 'StartPos'.

FindText

```
FindText('Text1','Text2','CaseSensitive'=FALSE,'N'=1)
```

Argumente

'Text1'	Textsignal oder Eingabetext (Eingabe in Anführungszeichen)
'Text2'	Text, der innerhalb von 'Text1' identifiziert werden soll
'CaseSensitive'	Optionaler Parameter, um bei der Suche die Groß- und Kleinschreibung zu berücksichtigen
'N'	Parameter, um nicht das erste, sondern das n-te Auftreten von 'Text2' zu finden

Beschreibung

Diese Funktion liefert als Ergebnis einen nullbasierten Index der Position des 'N'ten Auftretens von 'Text2' im 'Text1'. Wenn der Parameter 'CaseSensitive' auf TRUE gesetzt ist, wird bei der Suche die Groß- und Kleinschreibung beachtet. Wenn 'Text2' wird nicht gefunden wird, ist das Ergebnis -1.

15.9 SortText und XSortText

SortText

```
SortText('Expression','Descending'=FALSE,'CaseSensitive'=TRUE,
'NumericalSort'=FALSE)
```

Argumente

'Expression'	Textsignal, für das die Samples sortiert werden
'Descending'	Optionaler Parameter zum Umkehren der Sortierreihenfolge
'CaseSensitive'	Optionaler Parameter, um beim Vergleich die Groß- und Kleinschreibung zu berücksichtigen
'NumericalSort'	Optionaler Parameter, der angibt, ob Zahlen beim Vergleich als Text oder numerischer Wert betrachtet werden

Beschreibung

Diese Funktion sortiert Einträge eines Textsignals ('Expression') in aufsteigender Reihenfolge.

Standardeinstellungen:

- Sortierung in aufsteigender Reihenfolge ('Descending'=FALSE).
- Groß- und Kleinschreibung wird berücksichtigt ('CaseSensitive'=TRUE): Großbuchstaben werden vor Kleinschreibung einsortiert.
- Zahlen werden als Text verglichen ('NumericalSort'=FALSE):
  Wenn FALSE, steht 12034 vor 1234, da 0 vor 3 kommt.
  Wenn TRUE, steht 12034 nach 1234, da 12034 > 1234.

Um die Texteinträge absteigend zu sortieren, setzen Sie 'Descending' auf TRUE. Um bei der Sortierung die Großschreibung zu ignorieren ("A"="a"), setzen Sie 'CaseSensitive' auf FALSE. Um Zahlen als numerische Werte zu vergleichen, setzen Sie 'NumericalSort' auf TRUE.

XSortText

```
XSortText('Expression','Descending'=FALSE,'CaseSensitive'=TRUE,
'NumericalSort'=FALSE)
```

Argumente

'Expression'	Textsignal, für das die Samples sortiert werden
'Descending'	Optionaler Parameter zum Umkehren der Sortierreihenfolge
'CaseSensitive'	Optionaler Parameter, um beim Vergleich die Groß- und Kleinschreibung zu berücksichtigen
'NumericalSort'	Optionaler Parameter, um festzulegen, ob Ziffern jeweils einzeln verglichen werden oder Ziffernkombinationen als Zahlen verglichen werden

Beschreibung

Diese Funktion sortiert alle (X,Text)-Paare eines Textsignals ('Expression') entsprechend ihrer Texteinträge in aufsteigender Reihenfolge und gibt die X-Werte zurück. Das heißt, der letzte Y-Wert des sortierten Signals entspricht dem X-Wert, an dem 'Expression' sein Maximum hat. Die X-Achse und Y-Achse des sortierten Signals sind gleich lang.

Standardeinstellungen wie bei *SortText*.

Um die Texteinträge absteigend zu sortieren, setzen Sie 'Descending' auf TRUE. Um bei der Sortierung die Großschreibung zu ignorieren ("A"="a"), setzen Sie 'CaseSensitive' auf FALSE. Um Zahlen als numerische Werte zu vergleichen, setzen Sie 'NumericalSort' auf TRUE.

15.10 ToText und FromText

ToText

```
ToText('Expression','Format'="%g",'datatype'=0)
```

Argumente

'Expression'	Signal oder Ausdruck, dessen Inhalt in einen Textkanal konvertiert wird
'Format'	Optionaler Parameter für einen Formatstring
'datatype'	Optionaler Parameter, der das Gleitkommaformat festlegt ■ 'datatype'=0: Standardwert ■ 'datatype'=1: Ausgabe wird als signed 16 bit integer formatiert. ■ 'datatype'=2: Ausgabe wird als unsigned 16 bit integer formatiert. ■ 'datatype'=3: Ausgabe wird als signed 32 bit integer formatiert. ■ 'datatype'=4: Ausgabe wird als unsigned 16 bit integer formatiert.

Beschreibung

Diese Funktion wandelt einen numerischen Signalwert in ein Textsignal um. Dabei wird bei äquidistanten Samples des Eingangssignals 'Expression' und gleichbleibendem Y-Wert nur der Wert des ersten Signalpunkts als Sample im Textsignal eingetragen und angezeigt. Ändert sich der Y-Wert, wird für jeden neuen Y-Wert ein Sample im Textsignal eingetragen und angezeigt.

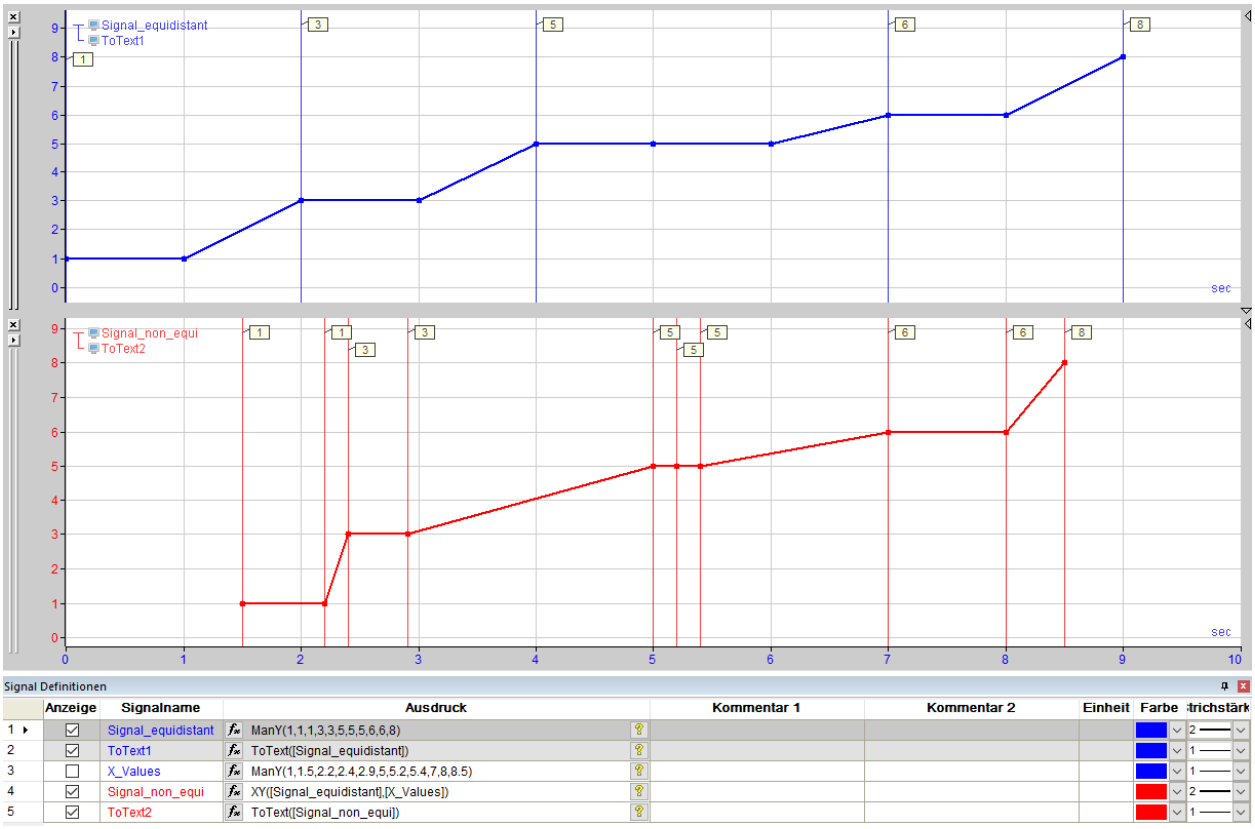
Enthält das Eingangssignal 'Expression' keine äquidistanten Samples, dann wird für jedes Eingangssample auch ein Sample im Textsignal eingetragen und angezeigt.

Geben Sie den optionalen Parameter 'Format' gemäß der C printf-Syntax ein. Sie können nur einen Parameter (%) angeben, der einem IEEE 32-Bit Gleitkommawert entsprechen muss. Standardwert ist %g. Dieser Wert wird auch verwendet, wenn Sie den optionalen Parameter nicht angeben.

Beispiele:
%g = Wandel des Gleitkommawertes in einen Text
%.4f = Text/Zahl mit 4 Stellen nach dem Komma, etc.

Beispiel

Die *ToText*-Funktion kann z. B. hilfreich sein, wenn Trends visualisiert werden, die große Datenmengen beinhalten. Ohne ständig zwischen der Marker- und Signalansicht zu wechseln, können auf einfache Weise die numerischen Werte angezeigt werden. Die nachfolgende Abbildung zeigt eine Anwendung der Funktion *ToText*.



Signal Definitionen

	Anzeige	Signalname	Ausdruck	Kommentar 1	Kommentar 2	Einheit	Farbe	trichstärk
1	☑	Signal_equidistant	ManY(1,1,1,3,3,5,5,5,6,6,8)					2
2	☑	ToText1	ToText([Signal_equidistant])					1
3	☐	X_Values	ManY(1,1.5,2.2,2.4,2.9,5,5.2,5.4,7,8,8.5)					1
4	☑	Signal_non_equi	XY([Signal_equidistant],[X_Values])					2
5	☑	ToText2	ToText([Signal_non_equi])					1

FromText

```
FromText('Text','Begin','End')
```

Argumente

'Text'	Textsignal, das konvertiert wird
'Begin'	Optional: Erstes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der komplette Inhalt ausgelesen.
'End'	Optional: Letztes Zeichen des Feldinhalts, das ausgelesen werden soll. Wenn Sie keinen Wert angeben, wird der Inhalt von 'Begin' bis zum letzten Zeichen ausgelesen.

Beschreibung

Diese Funktion konvertiert den Inhalt des Textsignals 'Text' in einen numerischen Wert. Die Parameter 'Begin' und 'End' können Sie optional als Indizes nutzen, um nicht den kompletten String umzuwandeln. Standardmäßig wird 'Begin'=0 und 'End'=Länge des Strings verwendet.

15.11 TrimText

```
TrimText('Text','RemoveOption'=0)
```

Argumente

'Text'	Textsignal oder Ausdruck, aus dem Leerzeichen entfernt werden
'RemoveOption'	Parameter zur Einstellung der Betriebsart: 0: Leerzeichen vor und nach dem Text entfernen (Standard) 1: Nur Leerzeichen vor dem Text entfernen 2: Nur Leerzeichen nach dem Text entfernen 3: Alle Leerzeichen entfernen, auch im Text

Beschreibung

Mithilfe dieser Funktion können Sie aus Texten die Leerzeichen entfernen. Die Funktion können Sie sowohl auf Textsignale, die bereits in der Messdatei enthalten sind, als auch auf Ergebnisse der Funktionen *InfoFieldText* und *ToText* anwenden.

Signal definitions

	Show	SignalName	Expression	Comment 2	Color
5	☑	Text	" abcd efgh 1234 5678 "	Original text	
6	☑	Trimmed_leading_and_trailing	TrimText ([Text], 0)	Blanks at the beginning and the end are remo	
7	☑	Trimmed_leading	TrimText ([Text], 1)	Blanks at the beginning are removed	
8	☑	Trimmed_trailing	TrimText ([Text], 2)	Blanks at the end are removed	
9	☑	Trimmed_all	TrimText ([Text], 3)	All blanks are removed	

16 Support und Kontakt

Support

Tel.: +49 911 97282-14

E-Mail: support@iba-ag.com

Hinweis

Wenn Sie Support benötigen, dann geben Sie bitte bei Softwareprodukten die Nummer des Lizenzcontainers an. Bei Hardwareprodukten halten Sie bitte ggf. die Seriennummer des Geräts bereit.

Kontakt

Hausanschrift

iba AG
Königswarterstraße 44
90762 Fürth
Deutschland

Tel.: +49 911 97282-0

E-Mail: iba@iba-ag.com

Postanschrift

iba AG
Postfach 1828
90708 Fürth

Warenanlieferung, Retouren

iba AG
Gebhardtstraße 10
90762 Fürth

Regional und weltweit

Weitere Kontaktadressen unserer regionalen Niederlassungen oder Vertretungen finden Sie auf unserer Webseite:

www.iba-ag.com