



# ibaPDA

## Ausdruckseditor

Handbuch Teil 4  
Ausgabe 8.4

Messsysteme für Industrie und Energie  
[www.iba-ag.com](http://www.iba-ag.com)

---

## Hersteller

iba AG  
Königswarterstraße 44  
90762 Fürth  
Deutschland

## Kontakte

Zentrale	+49 911 97282-0
Telefax	+49 911 97282-33
Support	+49 911 97282-14
Technik	+49 911 97282-13
E-Mail	iba@iba-ag.com
Web	www.iba-ag.com

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts sind nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz.

© iba AG 2023, alle Rechte vorbehalten.

Der Inhalt dieser Druckschrift wurde auf Übereinstimmung mit der beschriebenen Hard- und Software überprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass für die vollständige Übereinstimmung keine Garantie übernommen werden kann. Die Angaben in dieser Druckschrift werden jedoch regelmäßig aktualisiert. Notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten oder können über das Internet heruntergeladen werden.

Die aktuelle Version liegt auf unserer Website [www.iba-ag.com](http://www.iba-ag.com) zum Download bereit.

Version	Datum	Revision	Autor	Version SW
8.4	08/2023	Neue verschiedene und Diagnosefunktionen	rm	8.4.1

Windows® ist eine Marke und eingetragenes Warenzeichen der Microsoft Corporation. Andere in diesem Handbuch erwähnte Produkt- und Firmennamen können Marken oder Handelsnamen der jeweiligen Eigentümer sein.

## Inhalt

<b>1</b>	<b>Zu dieser Dokumentation.....</b>	<b>8</b>
1.1	Zielgruppe und Vorkenntnisse.....	8
1.2	Schreibweisen.....	8
1.3	Verwendete Symbole.....	9
1.4	Aufbau der Dokumentation.....	10
<b>2</b>	<b>Ausdruckseditor (Virtuelle Signale) .....</b>	<b>11</b>
2.1	Verschlüsselung bestimmter Textargumente.....	12
2.2	Logische Funktionen.....	12
2.2.1	Vergleichsfunktionen >, >=, <, <=, <>, = .....	12
2.2.2	Boolesche Funktionen .....	13
2.2.3	Logische Vernüpfungen bitweise.....	15
2.2.4	ExtendPulse .....	17
2.2.5	F_TRIG .....	18
2.2.6	FALSE .....	18
2.2.7	If.....	18
2.2.8	OneShot.....	20
2.2.9	R_TRIG .....	22
2.2.10	SetReset.....	23
2.2.11	Switch .....	25
2.2.12	TOF.....	26
2.2.13	Toggle .....	27
2.2.14	TON.....	28
2.2.15	TP.....	29
2.2.16	TRUE .....	29
2.3	Mathematische Funktionen.....	30
2.3.1	Grundrechenarten +, -, *, /.....	30
2.3.2	Abs.....	31
2.3.3	Add .....	32
2.3.4	Ceiling.....	33
2.3.5	Diff .....	34
2.3.6	Eff.....	35

2.3.7	Exp .....	36
2.3.8	Floor.....	36
2.3.9	Int .....	37
2.3.10	Log .....	40
2.3.11	Log10 .....	40
2.3.12	Mod .....	41
2.3.13	Pow .....	42
2.3.14	Round .....	43
2.3.15	Sqrt .....	44
2.3.16	Truncate .....	45
2.3.17	Trigonometrische Funktionen.....	46
2.4	Statistische Funktionen.....	48
2.4.1	Avg .....	48
2.4.2	Avg2 .....	49
2.4.3	AvgInTime .....	50
2.4.4	KurtosisInTime .....	52
2.4.5	MAvg .....	53
2.4.6	MAvgOnTrigger .....	55
2.4.7	Max .....	57
2.4.8	Max2 .....	58
2.4.9	MaxInTime .....	59
2.4.10	Median2 .....	61
2.4.11	MedianInTime .....	61
2.4.12	Min .....	63
2.4.13	Min2 .....	65
2.4.14	MinInTime .....	66
2.4.15	MKurtosis.....	68
2.4.16	MMax .....	69
2.4.17	MMedian .....	69
2.4.18	MMin .....	71
2.4.19	MSkewness .....	72
2.4.20	MStdDev .....	72

2.4.21	SkewnessInTime .....	73
2.4.22	StdDev .....	75
2.4.23	StdDev2 .....	76
2.4.24	StddevInTime.....	76
2.5	Trigger-Funktionen .....	78
2.5.1	PeriodicTrigger.....	78
2.5.2	TriggerChangeRate.....	79
2.5.3	TriggerConstant .....	80
2.5.4	TriggerEdge .....	82
2.5.5	TriggerLevel.....	83
2.5.6	TriggerHarmonicLevel.....	85
2.6	Text-Funktionen .....	87
2.6.1	CharValue .....	87
2.6.2	CompareText.....	88
2.6.3	ConcatText .....	90
2.6.4	ConvertFromText .....	90
2.6.5	ConvertToText.....	92
2.6.6	CountText.....	93
2.6.7	DeleteText.....	94
2.6.8	FindText .....	94
2.6.9	InsertText .....	96
2.6.10	MidText.....	97
2.6.11	ReplaceText.....	98
2.6.12	TextLength .....	100
2.6.13	TrimText .....	101
2.7	Verschiedene Funktionen .....	102
2.7.1	ClientInfo .....	102
2.7.2	ClientInfoText.....	103
2.7.3	Count .....	103
2.7.4	CountUpDown .....	106
2.7.5	Delay.....	108
2.7.6	DelayLengthL .....	109

2.7.7	DelayLengthV.....	110
2.7.8	DWORD.....	111
2.7.9	ElapsedTime .....	112
2.7.10	ExecuteCommand.....	113
2.7.11	GenerateSignal .....	116
2.7.12	GenerateText .....	118
2.7.13	GetFloatBit.....	119
2.7.14	GetIntBit .....	121
2.7.15	GetSignalMetaData.....	122
2.7.16	GetSystemTime.....	123
2.7.17	GetSystemTimeAsText .....	125
2.7.18	GetWeekOfYear .....	127
2.7.19	LimitAlarm .....	127
2.7.20	ModuleSignalCount .....	129
2.7.21	PulseFreq .....	130
2.7.22	RestartAcquisition .....	130
2.7.23	SampleAndHold .....	131
2.7.24	SampleOnce.....	132
2.7.25	Sign .....	132
2.7.26	T.....	133
2.7.27	VarDelay.....	134
2.7.28	WindowAlarm.....	136
2.8	Diagnosefunktionen .....	138
2.8.1	CameraStatus.....	138
2.8.2	DataStoreInfo.....	139
2.8.3	DataStoreInfoDB , ...Influx, ...Kafka, ...MindSphere, ...MQTT .....	140
2.8.4	DataStoreInfoHD.....	141
2.8.5	DongleInfo .....	142
2.8.6	FobDLinkStatus .....	143
2.8.7	FobFastLinkStatus .....	143
2.8.8	FobFlexDeviceStatus .....	144
2.8.9	FobFLinkStatus.....	145

2.8.10	FobMLinkStatus .....	145
2.8.11	FobPlusControlLinkStatus .....	146
2.8.12	FobSDLinkStatus, FobSDexpLinkStatus .....	147
2.8.13	FobTDCLinkStatus, FobTDCexpLinkStatus.....	147
2.8.14	ICPSensorStatus .....	148
2.8.15	InterruptCycleTime .....	148
2.8.16	InterruptTime .....	149
2.8.17	LicenseInfo.....	150
2.8.18	MultiStationStatus .....	151
2.8.19	PerformanceCounter .....	151
2.8.20	Ping.....	154
2.8.21	TimeSinceLastSync.....	154
2.8.22	TimeSyncStatus.....	155
2.9	Filter-Funktionen .....	156
2.9.1	BP.....	156
2.9.2	HP .....	156
2.9.3	LP .....	157
2.9.4	EnvelopeSpectral .....	158
2.9.5	Preprocess .....	159
2.10	Remanenz-Funktionen.....	160
2.11	Plugins .....	161
<b>3</b>	<b>Support und Kontakt .....</b>	<b>162</b>

# 1 Zu dieser Dokumentation

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

## 1.1 Zielgruppe und Vorkenntnisse

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.

## 1.2 Schreibweisen

In dieser Dokumentation werden folgende Schreibweisen verwendet:

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



## 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 des *ibaPDA*-Systems beschrieben. Sie ist als Leitfaden zur Einarbeitung wie auch als Nachschlagedokument angelegt. Die Teile und Kapitel folgen im Wesentlichen der Vorgehensweise bei der Konfiguration des Systems.

Ergänzend zu dieser Dokumentation können Sie für aktuellste Informationen zur installierten Programmversion die Versionshistorie im Hauptmenü *Hilfe – Änderungen* (Datei [versions.htm](#)) heranziehen. In dieser Datei wird neben der Aufzählung behobener Programmfehler auch auf Erweiterungspunkte des Systems stichwortartig hingewiesen.

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

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

Die Dokumentation des *ibaPDA*-Systems (PDF- und gedruckte Ausgabe) ist in sieben separate Teile gegliedert. Jeder Teil hat seine eigene bei 1 beginnende Kapitel- und Seitennummerierung und wird unabhängig aktualisiert.

<b>Teil 1</b>	Einführung und Installation	Allgemeine Hinweise, Lizenzpolitik, Add-ons Installation und Programmstart Benutzeroberfläche, Systemarchitektur, Client-Server Benutzerverwaltung, Drucken
<b>Teil 2</b>	I/O-Manager	Grundlagen zum I/O-Manager, allgemeine Einstellungen Gruppen und Vektorsignale, Textsignale, Ausgänge, Konfigurationsdateien
<b>Teil 3</b>	Datenschnittstellen und Module	Schnittstellen zur Messdatenerfassung Standardschnittstellen, ibaFOB, Ethernet-basierte Schnittstellen u. a. Für die Schnittstellen, für die es separate Handbücher gibt, wird auf diese verwiesen.
<b>Teil 4</b>	Ausdruckseditor	Alle Funktionen zur Berechnung virtueller Signale
<b>Teil 5</b>	Datenaufzeichnung	Arten der Datenaufzeichnung, Aufzeichnungsprofile, Signalauswahl
<b>Teil 6</b>	Datenvisualisierung	Alle Anzeigearten für Live-Daten, ihre Bedienung und Einstellung
<b>Teil 7</b>	Anhang	Verschiedene Ergänzungen, Fehlerlisten etc.

## 2 Ausdruckseditor (Virtuelle Signale)

Mithilfe von mathematischen Funktionen und booleschen Verknüpfungen können „Virtuelle Signale“ gebildet werden. Diese virtuellen Signale können wie andere Messsignale aufgezeichnet und/oder zur einfachen Realisierung komplexer Triggerbedingungen verwendet werden. Mithilfe der virtuellen Signale können Sie bereits bei der Messung Berechnungen, wie z. B. Summen- oder Differenzbildung, Prüfung auf Toleranzverletzungen u.v.m. durchführen lassen. Oder Sie erzeugen sich Referenzsignale oder -Merkmale um weitere Vergleiche bei der Analyse durchzuführen.

---

### Hinweis



Der Datentyp der resultierenden Analogwerte aus einer Berechnung in einem virtuellen Modul ist immer Float, egal welchen Datentyp die Eingangssignale haben, die innerhalb des Ausdrucks verwendet werden.

---

---

### Hinweis



Im Ausdruckseditor und in allen Ausdrücken und Formeln ist grundsätzlich der Punkt als Dezimaltrennzeichen zu verwenden!

---

---

### Hinweis



Ein Hinweis zur Schreibweise der Funktionsparameter (Argumente) in Tooltips und Hilfetexten:

Ist einem Argument ein Wert zugewiesen, so ist dieser der voreingestellte Wert, der angenommen wird, wenn der Parameter weggelassen wird. Es können nur Parameter weggelassen werden, die einen Voreinstellungswert haben, z. B. *GenerateSignal('Type', 'Amplitude=10', 'T1=1', 'T2=1')*.

Hier können die Argumente 2 (Amplitude), 3 (T1) und 4 (T2) weggelassen werden, wenn man mit den Voreinstellungswerten arbeiten möchte. *GenerateSignal(3,10,1,1)* ist gleichbedeutend mit *GenerateSignal(3)*.

Eine Schreibweise wie *GenerateSignal(3,,2,5)* ist dagegen nicht zulässig. Wenn ein Argument weggelassen wird, dann müssen auch alle nachfolgenden (optionalen) Argumente weggelassen werden. Es sind keine Lücken erlaubt. Werden aber die hinteren Argumente benötigt, muss auch das vordere optionale Argument mit einem Wert oder Ausdruck versehen werden. In diesem Beispiel wäre die korrekte Schreibweise *GenerateSignal(3,10,2,5)*.

---

---

**Hinweis**

Wenn Sie Anführungszeichen in einem statischen Text verwenden wollen, dann schreiben Sie jeweils zwei Anführungszeichen hintereinander.

---

## 2.1 Verschlüsselung bestimmter Textargumente

Bei der Eingabe bestimmter Argumente können Sie eine Verschlüsselung der Texte vornehmen lassen, damit diese Eingaben später nicht mehr lesbar sind. Dies betrifft in erster Linie Eingaben von Benutzernamen und Kennwörtern, wie z. B. bei der ExecuteCommand-Funktion.

Wenn Sie bei der Eingabe zu einem Argument kommen, für das die Verschlüsselung unterstützt wird, öffnet sich ein Fenster mit dem Befehl "Text verschlüsseln". Wenn Sie die Eingabe verschlüsseln wollen, klicken Sie auf "Text verschlüsseln". Es öffnet sich eine Eingabezeile für das Argument. Geben Sie das Argument ein und klicken Sie auf <OK>. Anstelle des lesbaren Textes wird nun eine generierte Zeichenfolge mit dem Präfix `encrypted_` als Argument in die Funktion eingefügt.

## 2.2 Logische Funktionen

### 2.2.1 Vergleichsfunktionen >, >=, <, <=, <>, =

Mit den Vergleichsoperationen > (größer), >= (größer / gleich), < (kleiner), <= (kleiner / gleich), <> (ungleich) und = (gleich) können die Werte zweier Ausdrücke (Operanden) miteinander verglichen werden. Die Operationen liefern als Ergebnis jeweils den booleschen Wert TRUE oder FALSE. Als Operanden können Originalsignale, berechnete Ausdrücke oder einfach konstante Werte eingetragen werden. Das Ergebnis kann als neuer Ausdruck wie ein Signal dargestellt und ausgewertet werden. Somit lassen sich leicht binäre Signale bilden, die ihrerseits wieder als Bedingungen für andere Funktionen verwendet werden 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. D.h. ein Wechsel von TRUE nach FALSE (oder umgekehrt) wird stets im Raster der Messpunkte eingetragen. Die Verbindungslinie zwischen zwei Messpunkten bei der Darstellung von Analogwerten ist nur eine grafische Näherung

---

## 2.2.2 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 (logisches UND), OR (logisches ODER), NOT (logisches NOT, Negation) und XOR (logisches Exklusiv-ODER) können binäre Ausdrücke, z. B. Digitalsignale miteinander verknüpft werden. Entsprechend den Regeln der booleschen Logik liefern die Funktionen als Ergebnis jeweils den Wert TRUE oder FALSE. Als Parameter können Digitalsignale, berechnete (binäre) Ausdrücke oder die Zahlenwerte 0 bzw. 1 eingetragen werden.

Das Ergebnis kann als neuer Ausdruck wie ein Signal dargestellt und ausgewertet werden. Somit lassen sich leicht binäre Signale bilden, die ihrerseits wieder als Bedingungen für andere Funktionen verwendet werden können.

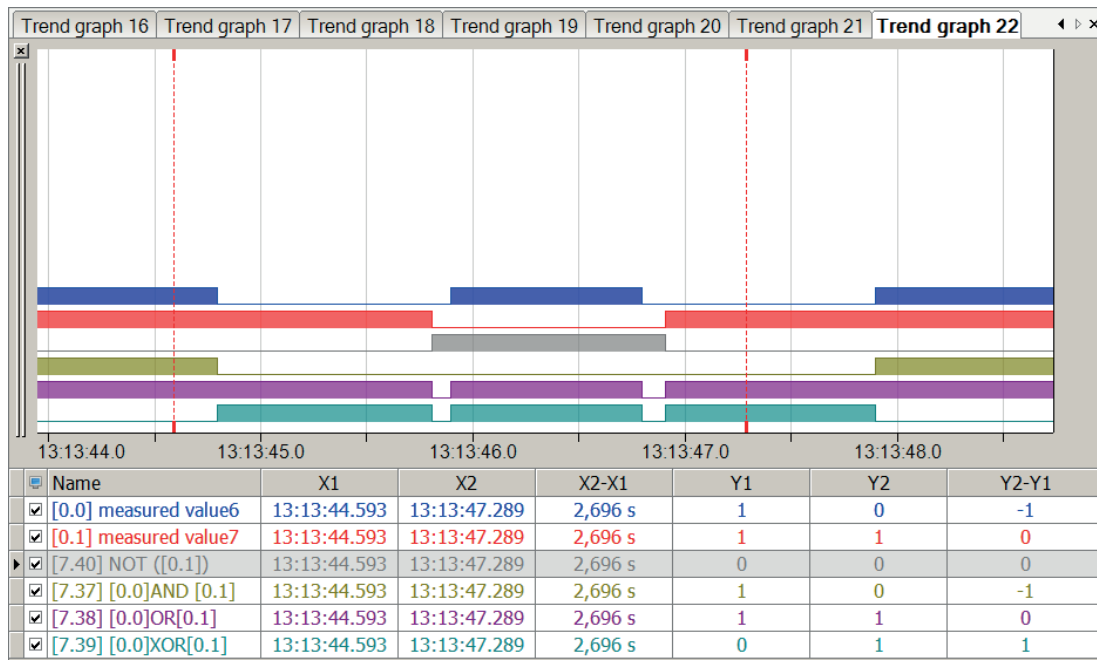
AND			OR			XOR			NOT	
A	B	f (A,B)	A	B	f (A,B)	A	B	f (A,B)	A	f (A)
0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	1	1	0	1	1	0
0	1	0	0	1	1	0	1	1		
1	1	1	1	1	1	1	1	0		

Tab. 1: Logische Funktionen, Wahrheitstabellen

## Beispiel

Grafische Darstellung der booleschen Funktionen

## Lösung



## 2.2.3 Logische Vernüpfungen bitweise

z. B. 'Expression1' bw\_AND 'Expression2'

### Übersicht

Operation	Funktion
bw_AND	Bitweise addieren
bw_OR	Bitweise verodern
bw_XOR	Bitweise exklusiv verodern
bw_NOT	Bitweise negieren

### Beschreibung

Mit diesen Funktionen können Sie zwei Analogwerte bitweise miteinander verknüpfen. Vorzugsweise anzuwenden auf ganzzahlige Datentypen wie BYTE, WORD, INTEGER usw.

Bei Gleitkommazahlen (FLOAT) wird nur der ganzzahlige Teil berücksichtigt, indem der Wert intern in ein 32Bit-Integer gewandelt wird, ohne zu runden.

### Beispiele

Anhand der Vernüpfung von zwei Dezimalzahlen zeigen die folgenden Tabellen das Ergebnis der Operationen.

#### Bitweise addieren (bw\_AND)

[Wert A] bw\_AND [Wert B]

Wert A = 25	0	0	0	1	1	0	0	1
Wert B = 14	0	0	0	0	1	1	1	0
Ergebnis bw_AND (dezimal = 8)	0	0	0	0	1	0	0	0

#### Bitweise verodern (bw\_OR)

[Wert A] bw\_OR [Wert B]

Wert A = 25	0	0	0	1	1	0	0	1
Wert B = 14	0	0	0	0	1	1	1	0
Ergebnis bw_OR (dezimal = 31)	0	0	0	1	1	1	1	1

#### Bitweise exklusiv verodern (bw\_XOR)

[Wert A] bw\_XOR [Wert B]

Wert A = 25	0	0	0	1	1	0	0	1
Wert B = 14	0	0	0	0	1	1	1	0
Ergebnis bw_XOR (dezimal = 23)	0	0	0	1	0	1	1	1

**Bitweise negieren (bw\_NOT)**

Liefert das Komplement der Bits eines Wertes.

`bw_NOT([Wert A])`

Wert A = 25	0	0	0	1	1	0	0	1
Ergebnis bw_NOT (dezimal = -26)	1	1	1	0	0	1	1	0



## 2.2.4 ExtendPulse

`ExtendPulse('Input', 'Time*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

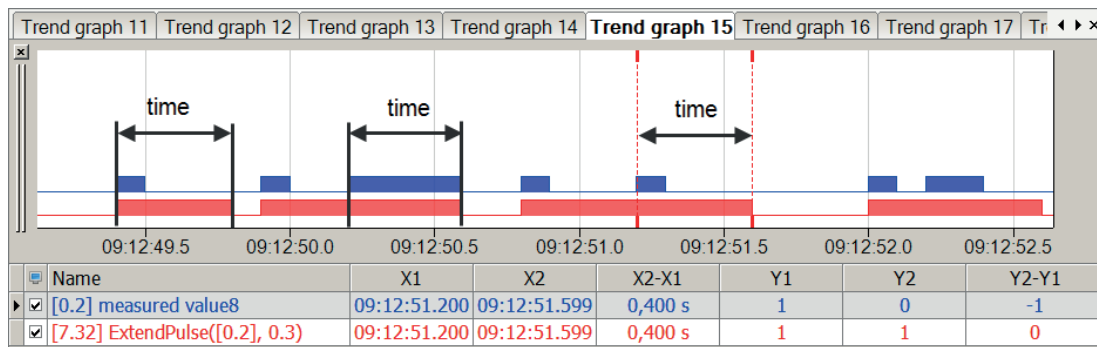
Diese Funktion verlängert einen Impuls an 'Input' auf die Mindestlänge von 'Time' Sekunden. Eine weitere steigende Flanke an 'Input' startet die Zeit neu.

### Beispiel

Verlängerung der Impulse eines Messwerts auf eine Mindestlänge von 0,3.

### Lösung

In der nachfolgenden Abbildung zeigt der blaue Balken den Messwert und der rote Balken zeigt die verlängerte Impulse des Messwerts.



### Hinweis



Durch das Schalten der Flanken im Messpunkttraster können die grafische Darstellung und die Berechnung der Mindestlänge einen Messpunkt über der unter 'Time' angegebenen Mindestlänge liegen.

## 2.2.5 F\_TRIG

`F_TRIG('Expression')`

### Argumente

'Expression'	Digitales Eingangssignal oder digitaler Ausdruck
--------------	--

### Beschreibung

Diese Funktion liefert den Wert TRUE für 1 Sample, wenn der Übergang von TRUE nach FALSE bei 'Expression' erkannt wurde.

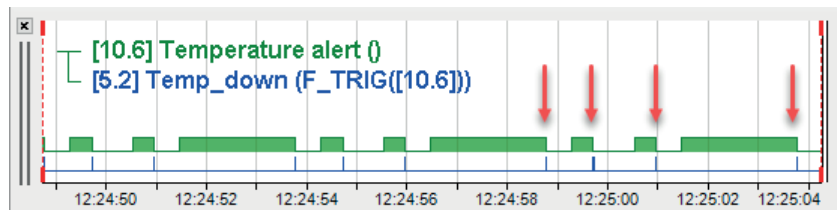
Damit können gezielt fallende Flanken erkannt und signalisiert werden.

### Beispiel

Jedes Mal, wenn das Signal einer Temperaturwarnung abfällt, soll ein Signal erzeugt werden.

### Lösung

Mithilfe der Funktion F\_TRIG ein digitales Signal *Temp\_down* erzeugen, das nur bei einer fallenden Flanke von dem Temperaturwarnsignal *Temperature\_alert* kurz den Wert TRUE annimmt.



## 2.2.6 FALSE

`FALSE()`

### Beschreibung

Gibt den logischen Ausdruck FALSE bzw. Null (0) zurück.

## 2.2.7 If

`If('Condition', 'Expression1', 'Expression2')`

### Argumente

'Condition'	Bedingung als Operation mit den boolschen Ergebnissen TRUE oder FALSE	
'Expression1'	Operation wird ausgeführt, wenn 'Condition' gleich TRUE	
'Expression2'	Operation wird ausgeführt, wenn 'Condition' gleich FALSE	

## Beschreibung

Die If-Funktion dient zur bedingten Ausführung weiterer Berechnungen. Abhängig vom boole- schen Ergebnis einer Bedingung ('Condition'), die selbst eine Operation sein kann, wird beim Ergebnis TRUE die Operation 'Expression1' ausgeführt, beim Ergebnis FALSE entsprechend die Operation 'Expression2'.

Damit lassen sich unterschiedliche Berechnungen prozessgesteuert durchführen. Die Funktion kann natürlich auch verschachtelt genutzt werden, um weitere Verzweigungen zu realisieren. Textsignale werden unterstützt.

### Tipp



Wenn für 'Condition' nur eine Größe eingetragen wird, wird als Bedingung abge- fragt, ob die Größe größer als (TRUE) oder kleiner als (FALSE) 0,5 ist.

### Hinweis



Anstelle der If-Funktion kann auch die Funktion "Switch" verwendet werden. "Switch" bietet darüber hinaus den Vorteil, dass mehr als zwei Fälle abgebildet werden können.

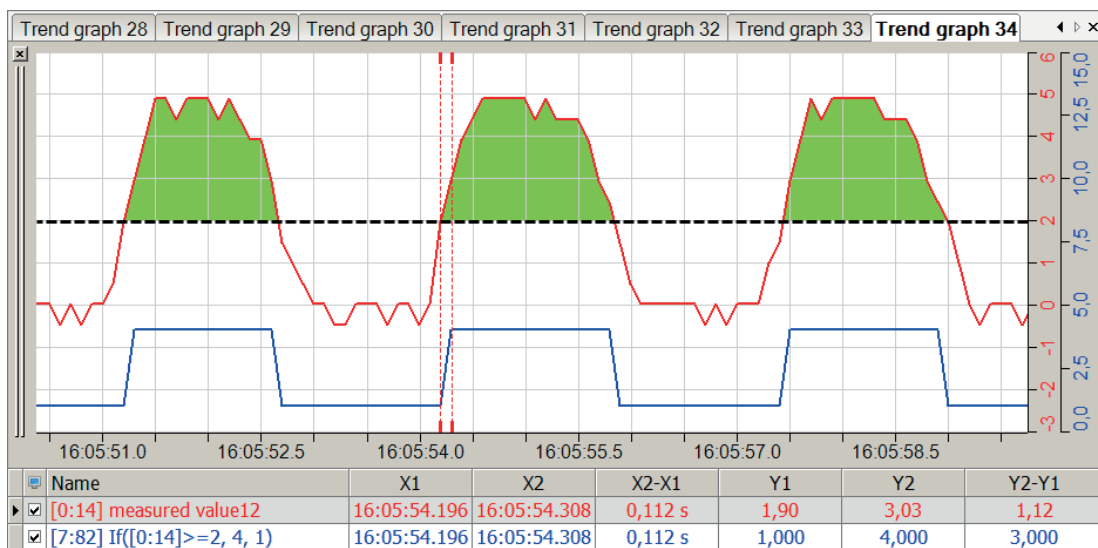
## Beispiel

Erkennen, wann ein Messsignal über einer gesetzten Grenze liegt

## Lösung

Die Grenze wird in 'Condition' als Bedingung mit logischen Operanden formuliert. Ist 'Condition' TRUE, wird der Wert 4 zurückgegeben, bei FALSE der Wert 1.

In der nachfolgenden Abbildung ist oberhalb der gestrichelten Linie die Bedingung erfüllt (grüne Markierung). Unterhalb der gestrichelten Linie ist die Bedingung jedoch nicht erfüllt.



## 2.2.8 OneShot

`OneShot('Expression')`

### Beschreibung

Diese Funktion liefert das Ergebnis TRUE, wenn der aktuelle Messwert von 'Expression' ungleich dem vorigen ist. Sie liefert das Ergebnis FALSE, wenn der aktuelle Messwert gleich dem vorigen ist. Die Funktion unterstützt auch Textsignale.

### Beispiel 1

Erkennung von Wertänderungen

### Aufgabenstellung

Für einen realen Signalverlauf sollen Wertänderungen angezeigt werden.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve das ursprüngliche Signal und der rote Balken zeigt die Bereiche mit einer Wertänderung des Signals.



## Beispiel 2

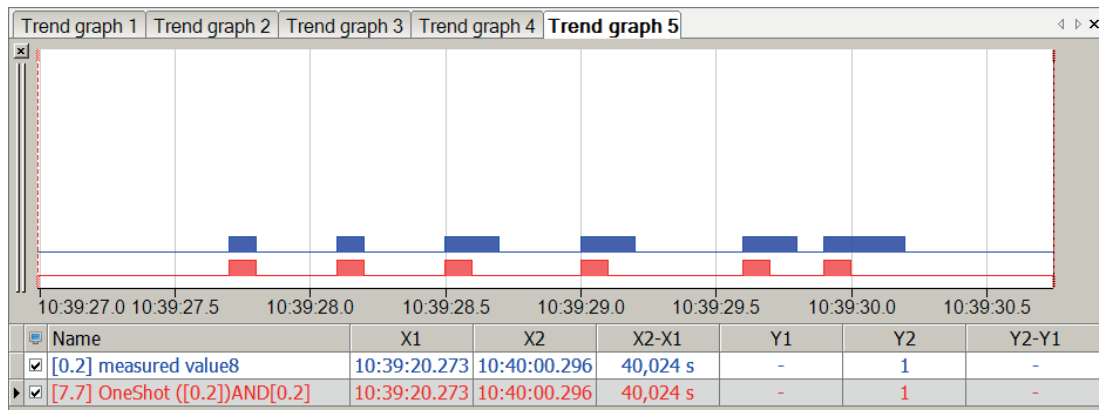
Erkennung von Flanken

### Aufgabenstellung

Es sollen die positiven Flanken eines Digitalsignales gebildet werden.

### Lösung

In der nachfolgenden Abbildung zeigt der blaue Balken das ursprüngliche Signal und der rote Balken zeigt die positiven Flanken.



## 2.2.9 R\_TRIG

`R_TRIG('Expression')`

### Argumente

'Expression'	Digitales Eingangssignal oder digitaler Ausdruck
--------------	--

### Beschreibung

Diese Funktion liefert den Wert TRUE für 1 Sample, wenn der Übergang von FALSE nach TRUE bei 'Expression' erkannt wurde.

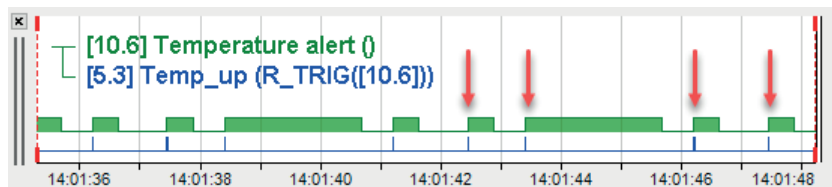
Damit können gezielt steigende Flanken erkannt und signalisiert werden.

### Beispiel

Jedes Mal, wenn das Signal einer Temperaturwarnung auftritt, soll ein Signal erzeugt werden.

### Lösung

Mithilfe der Funktion `R_TRIG` ein digitales Signal *Temp\_up* erzeugen, das nur bei einer steigenden Flanke von dem Temperaturwarnsignal *Temperature\_alert* kurz den Wert TRUE annimmt.



## 2.2.10 SetReset

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

### Argumente

'Set'	Positive Flanke setzt Funktion auf TRUE	
'Reset'	Positive Flanke setzt Funktion auf FALSE	
'Setdomi- nant*'	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

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion wird verwendet, um ein digitales Ergebnis (TRUE/FALSE) mithilfe von positiven Flanken (Übergang von 0 zu 1) der Argumente 'Set' und 'Reset' zu steuern.

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

### Tipp

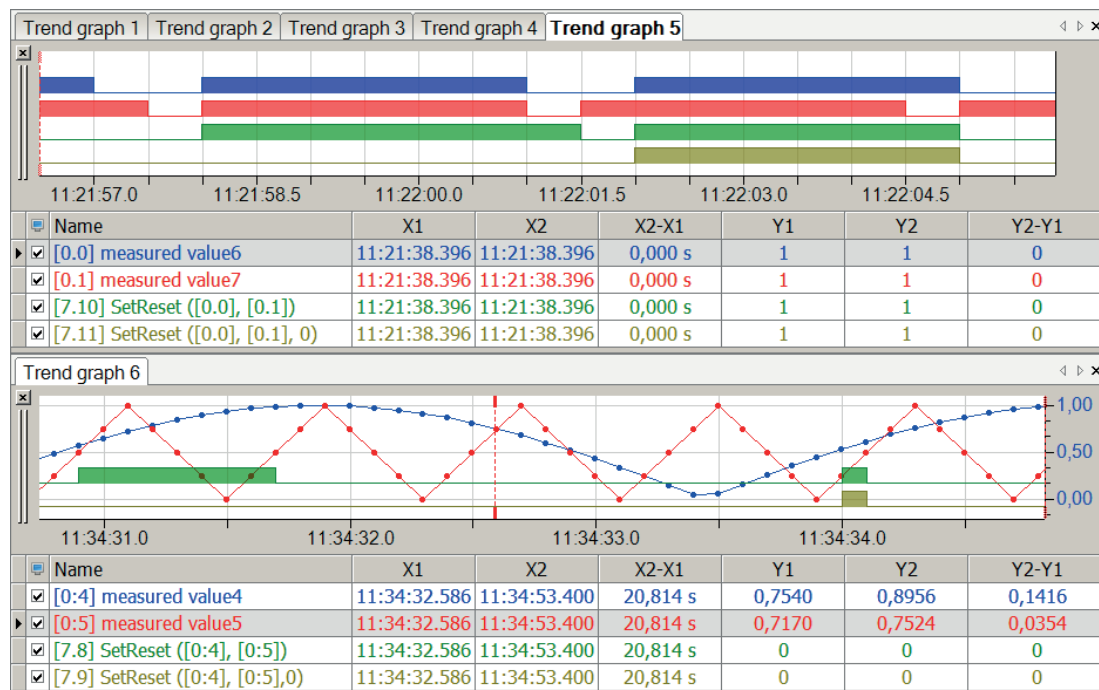


Bei einem analogen Signal entspricht das Überschreiten des Wertes 0,5 einer positiven Flanke.

## Beispiel

Diese Funktion kann dazu verwendet werden, bedingte Berechnungen mit einem Signal zu aktivieren und mit einem anderen Signal zu deaktivieren (z. B. in Verbindung mit IF-Funktion).

Die nachfolgende Abbildung zeigt die Funktion SetReset mit digitalen und analogen Signalen. Bei zeitgleichen Flanken wird der unter 'SetDominant' festgelegte Parameter geschaltet. Bei zeitlich versetzten Flanken hat der Parameter 'SetDominant' keine Auswirkung.





### 2.2.11 Switch

```
Switch('Selector', 'Case1', 'Value1', 'Case2', 'Value2', ..., 'Default')
```

#### Argumente

'Selector'	Ausdruck, welcher auf verschiedene Bedingungen überprüft wird
'CaseN'	Ausdruck, welcher mit 'Selector' verglichen wird
'ValueN'	Ergebnis, falls 'Selector' und 'CaseN' übereinstimmen
'Default'	Ergebnis, falls keiner der 'CaseN' mit 'Selector' übereinstimmt

#### Beschreibung

Diese Anweisung vergleicht einen 'Selector'-Ausdruck mit bis zu 100 Fällen 'CaseN', angelehnt an das SQL Statement CASE. Es werden mindestens 3 Argumente benötigt. Bei einer geraden Anzahl von Argumenten wird automatisch das letzte als 'Default' interpretiert, welches herangezogen wird, wenn keiner der 'CaseN'-Ausdrücke mit dem Ausdruck 'Selector' übereinstimmt.

Falls 'Selector' und 'CaseN' übereinstimmen, wird der dazugehörige 'ValueN'-Ausdruck zurückgegeben. Stimmen mehrere 'CaseN' mit 'Selector' überein, wird automatisch der erste 'CaseN'-Ausdruck ausgewählt.

Als 'Selector' sind folgende Signale zulässig:

- Eine numerische Konstante
- Eine Text-Konstante
- Ein äquidistant oder nicht-äquidistant gesampelter Kanal
- Ein Text-Kanal

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

## 2.2.12 TOF

`TOF('IN', 'PT*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

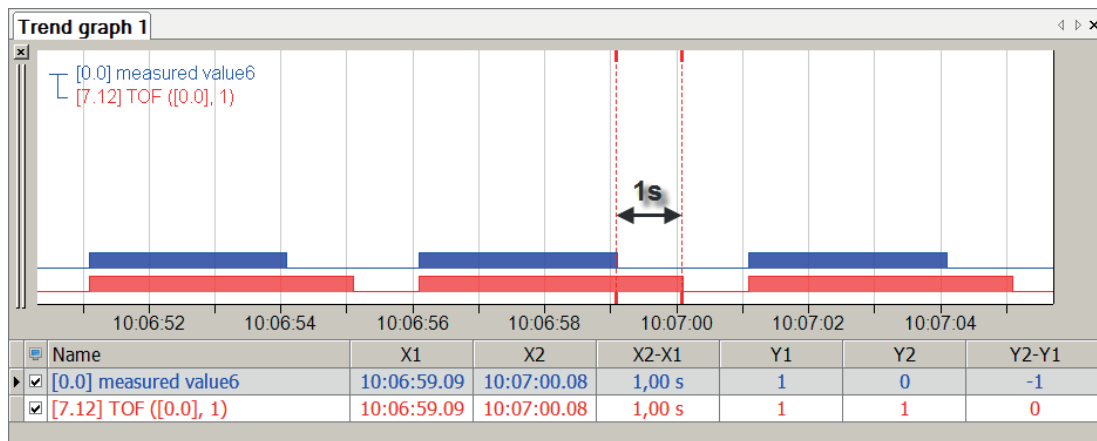
Ausschaltverzögerung. Der Ausgang wird 'pt' Sekunden nach dem Ausschalten des Eingangs 'in' ausgeschaltet.

### Beispiel

Verzögerung des Ausschaltens um eine Sekunde

### Lösung

In der nachfolgenden Abbildung zeigt der blaue Balken den Messwert und der rote Balken zeigt den Ausgangswert mit um eine Sekunde verzögertem Ausgang.



### 2.2.13 Toggle

```
Toggle('Trigger', 'Initial=0')
```

#### Argumente

'Trigger'	Triggersignal (steigende Flanke), mit dem der Ausgang umschaltet
'Initial=0'	Wert zum Start der Erfassung Wird dieses Argument nicht angegeben ist der Startwert 0.

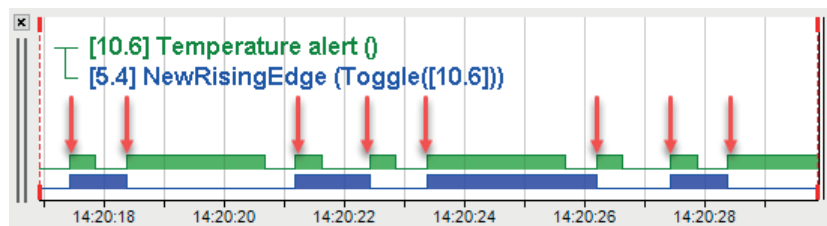
#### Beschreibung

Diese Funktion schaltet den Ausgang mit jeder steigenden Flanke an 'Trigger'-Signal um. Der letzte Parameter 'Initial' ist optional und bestimmt den Wert zum Start der Erfassung.

#### Beispiel

Die steigenden Flanken eines Temperaturwarnsignals *Temperature alert* sollen ein digitales Signal *NewRisingEdge* ein- und ausschalten.

#### Lösung



## 2.2.14 TON

`TON('IN', 'PT*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

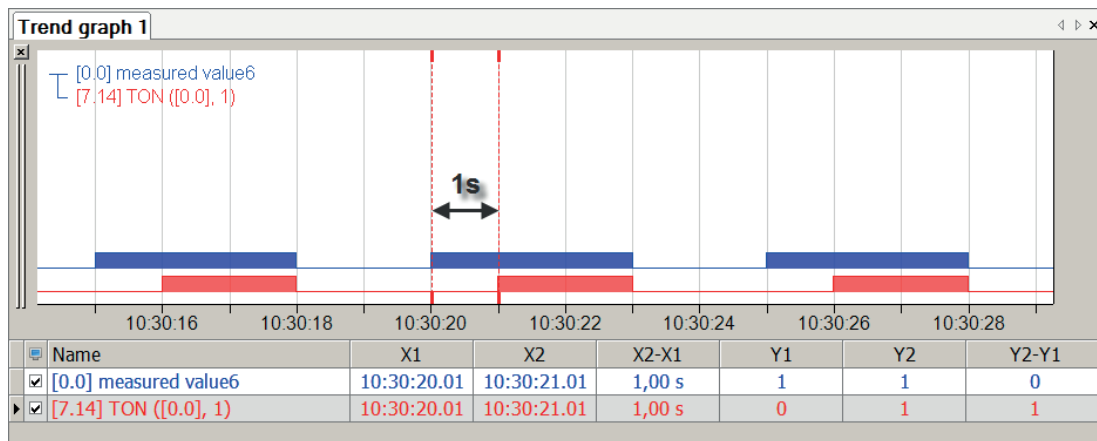
Einschaltverzögerung. Der Ausgang wird 'pt' Sekunden nach dem Einschalten des Eingangs 'in' eingeschaltet.

### Beispiel

Verzögerung des Einschaltens um eine Sekunde

### Lösung

In der nachfolgenden Abbildung zeigt der blaue Balken den Messwert und der rote Balken zeigt den Eingangswert mit um eine Sekunde verzögertem Eingang.



## 2.2.15 TP

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

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

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

### Tipp



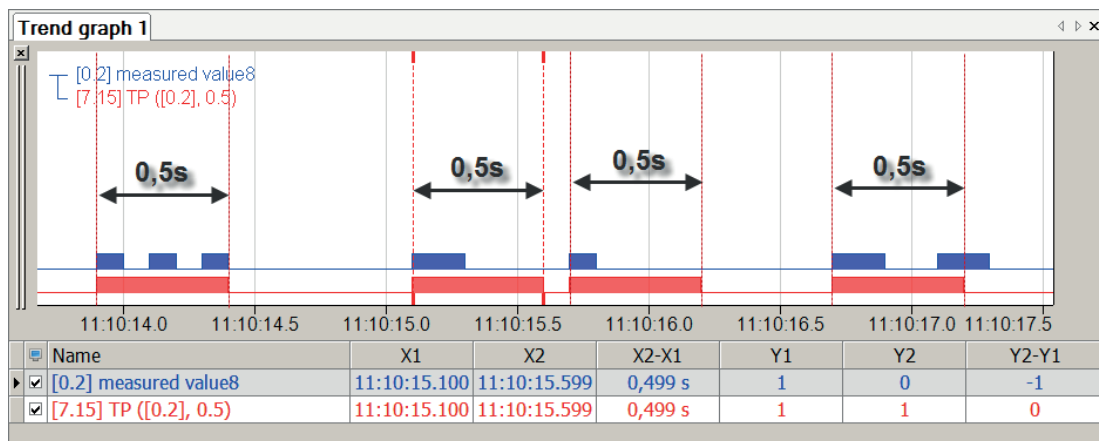
Eine weitere steigende Flanke während des Ausgangs-Impulses verlängert die Schaltung nicht und startet den Impuls nicht neu.

### Beispiel

Erzeugung eines 0,5 Sekunden Impulses aus einem Zufallssignal

### Lösung

In der nachfolgenden Abbildung zeigt der blaue Balken den Messwert und der rote Balken zeigt den Ausgangswert in Impulsen von 0,5 Sekunden.



## 2.2.16 TRUE

`TRUE ()`

### Beschreibung

Gibt den logischen Ausdruck TRUE bzw. 1 zurück.

## 2.3 Mathematische Funktionen

### 2.3.1 Grundrechenarten +, -, \*, /

```
('Expression1')+('Expression2')
```

#### Beschreibung

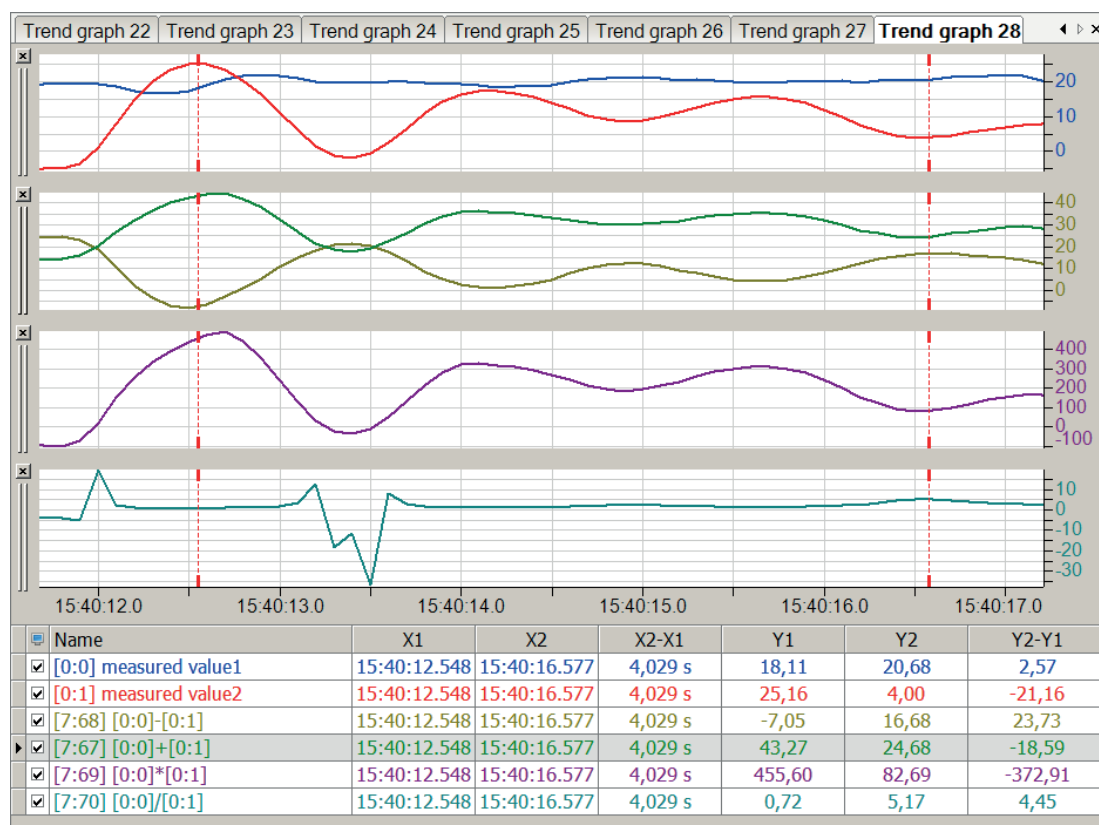
Alle Signale und Ausdrücke können mit den Grundrechenarten (Addition, Subtraktion, Multiplikation und Division) verarbeitet werden. Wenn als Operanden digitale Signale oder Ausdrücke mit den Grundrechenarten verwendet werden, dann übersetzt das Programm die Werte TRUE mit 1.0 und FALSE mit 0.0. Das Ergebnis einer Grundrechenoperation ist stets ein analoger Ausdruck.

#### Beispiel

Grafische Darstellung der Grundrechenarten

#### Lösung

Die nachfolgende Abbildung zeigt die Ergebnisse der verschiedenen Grundrechenarten.



Blau	Messwert 1	Rot	Messwert 2
Grün	Addition	Gelb	Subtraktion
Lila	Multiplikation	Hellblau	Division

## 2.3.2 Abs

`Abs('Expression')`

### Beschreibung

Die Absolutfunktion gibt den Absolutwert ( $= |\text{Betrag}|$ ) von 'Expression' zurück.

### Beispiel

Von einem Messsignal ist nur der absolute Betragswert relevant

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve das ursprüngliche Signal und die rote Kurve den Absolutwert des Signals.



### Tipp



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

### 2.3.3 Add

```
Add('Expression', 'Enable', 'Reset=0')
```

#### Argumente

'Expression'	Analoges Eingangssignal oder analoger Ausdruck	
'Enable'	Digitales Eingangssignal oder digitaler Ausdruck; 'Enable' = 1 aktiviert die Addition	
'Reset'	Optionaler Parameter (Voreinstellung = 0) zum Rücksetzen und Neustarten der Berechnung:	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten und Ergebnis auf 0 setzen

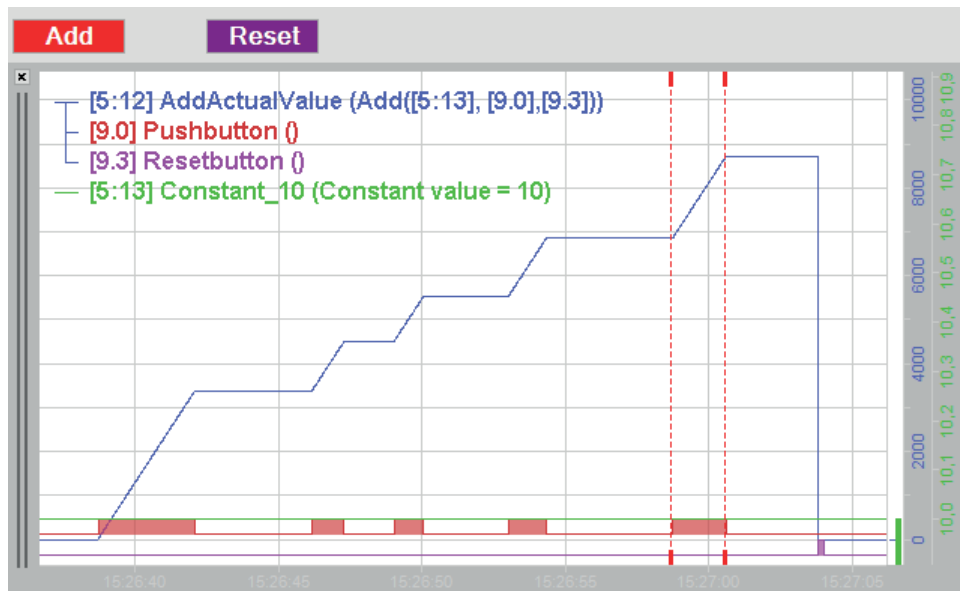
#### Beschreibung

Diese Funktion addiert jeden Zyklus den Wert von 'Expression' zum Momentanwert hinzu, wenn 'Enable' = TRUE bzw. 1 ist. Wenn 'Enable' = FALSE bzw. 0 ist, wird der letzte Momentanwert gehalten. Mit 'Reset' = TRUE bzw. 1 wird der Momentanwert auf 0 gesetzt.

#### Beispiel

Immer wenn ein Drucktaster (Pushbutton) betätigt wird, soll ein Analogwert pro Zyklus um den Wert 10 erhöht werden. Mit einem Reset-Drucktaster soll der Wert wieder auf 0 zurückgesetzt werden.

#### Lösung





### 2.3.4 Ceiling

`Ceiling('Expression')`

#### Beschreibung

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

#### Beispiel

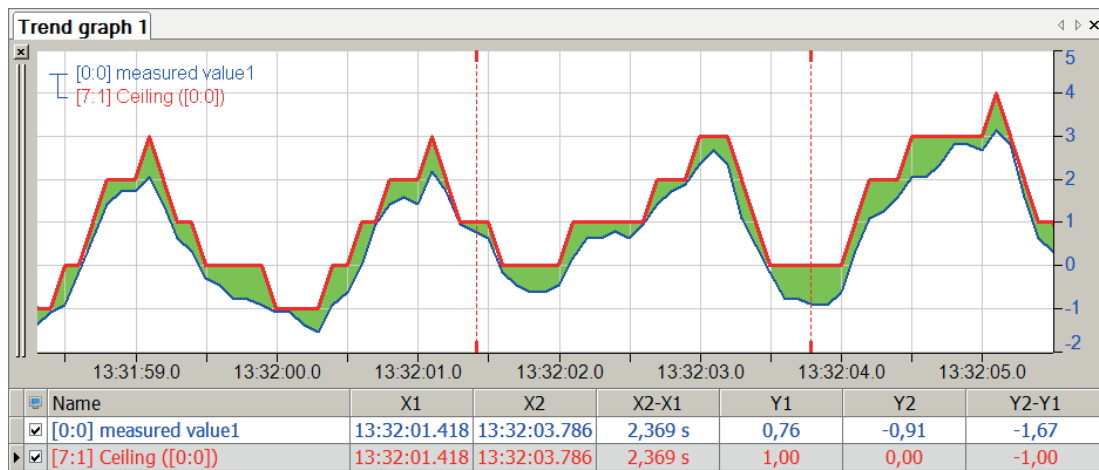
Aufrunden der Werte eines Signalverlaufs auf ganze Zahlen.

#### Aufgabenstellung

Von einem realen Signalverlauf soll für jedes Sample der Messwert auf den nächsten ganzzahligen Wert aufgerundet werden.

#### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve das ursprüngliche Signal und die rote Kurve zeigt den Signalverlauf mit aufgerundeten Werten.



## 2.3.5 Diff

`Diff('Expression')`

### Beschreibung

Diese Funktion liefert als Ergebnis das Differential  $dx/dt$  von 'Expression' zurück.

### Beispiel

Ist 'Expression' ein Längenmesssignal, so kann mit der Diff-Funktion daraus der Geschwindigkeitsverlauf ermittelt werden.

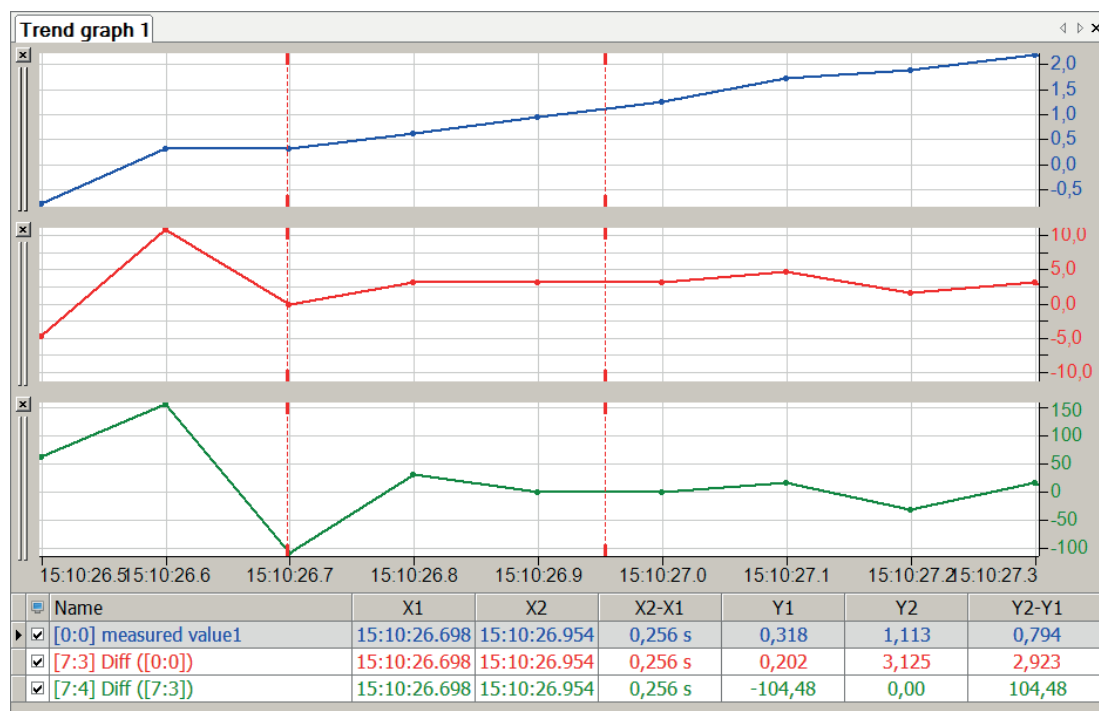
### Aufgabenstellung

Aus einem Längenmesssignal soll der Geschwindigkeitsverlauf und aus diesem der Beschleunigungsverlauf ermittelt werden.

### Lösung

Durch erneutes (iteratives) Ausführen der Diff-Funktion unter Verwendung der Werte der ersten als 'Expression' erhält man den Verlauf der Beschleunigung.

In der nachfolgenden Abbildung zeigt die blaue Kurve die zurückgelegte Wegstrecke, die rote Kurve den berechneten Verlauf der Geschwindigkeit und die grüne Kurve den Verlauf der Beschleunigung.



### Tipp



Ist nur der Verlauf der Beschleunigung von Interesse und die Geschwindigkeit nicht relevant, so kann dieser auch direkt durch rekursives Aufrufen der Funktion `Diff(Diff([0:0]))` ermittelt werden.

### 2.3.6 Eff

Eff('Expression', 'Frequency')

#### Beschreibung

Diese Funktion berechnet den Effektivwert von 'Expression' mit der Grundfrequenz von 'Frequency'.

Die folgende Formel wird für die Berechnung des Effektivwertes verwendet:

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

$e(n)$  : Messpunkt  $n$  von Signal  $e$  ('expr')

$N$  : Anzahl der Messwerte pro Periode

#### Hinweis



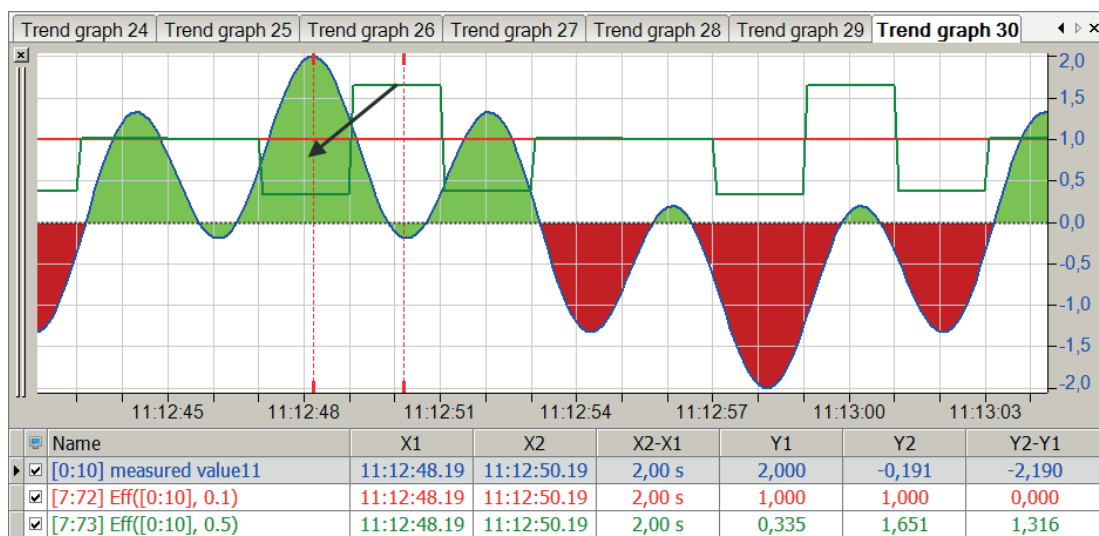
Der Effektivwert wird erst im nächsten Takt der Grundfrequenz ausgegeben.

#### Beispiel

Für einen Wechselspannungsverlauf mit einer Frequenz von 0,1 kHz, der von einer zweiten Wechselspannung mit 0,5 kHz überlagert wird, soll der Effektivwert der Spannung für beide Frequenzen ermittelt werden.

#### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve das Wechselspannungssignal und die grüne Kurve zeigt die Effektivspannung mit der Berechnungsfrequenz 0,5 kHz. Die rote Gerade zeigt die Effektivspannung mit der Berechnungsfrequenz 0,1 kHz.



### 2.3.7 Exp

`Exp('Expression')`

#### Beschreibung

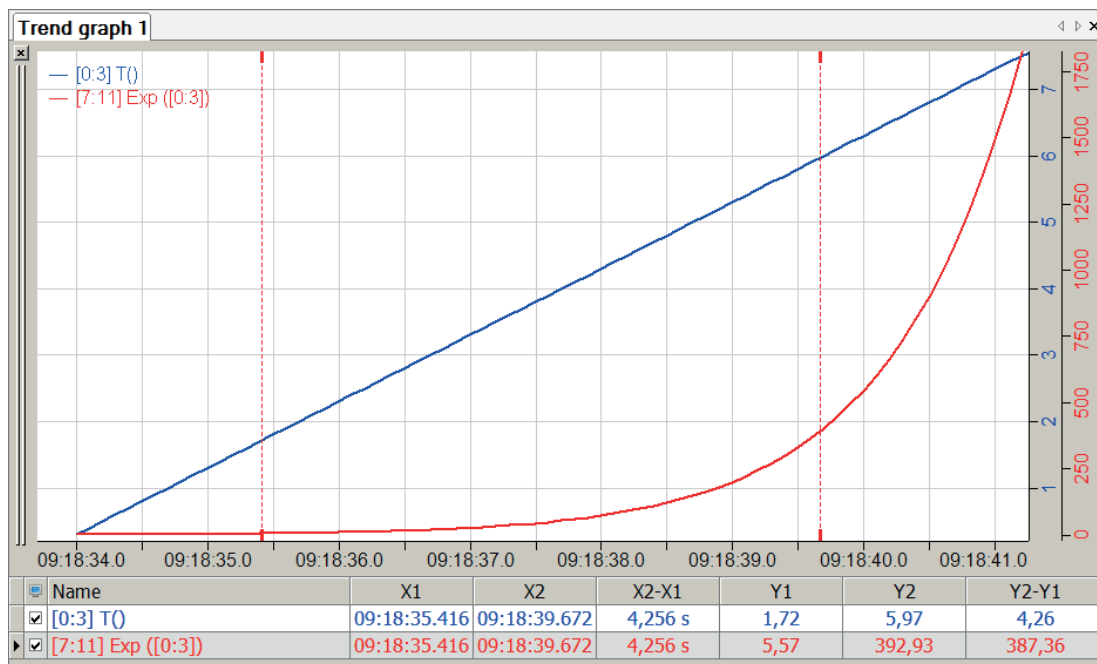
Diese Funktion berechnet das Ergebnis von  $(e)^{\text{'Expression'}}$

#### Beispiel

Grafische Darstellung der Exp-Funktion

#### Lösung

In der nachfolgenden Grafik zeigt die blaue Kurve die Zeitfunktion und die rote Kurve zeigt die Exponentialfunktion der Zeit.



### 2.3.8 Floor

`Floor('Expression')`

#### Beschreibung

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

#### Beispiel

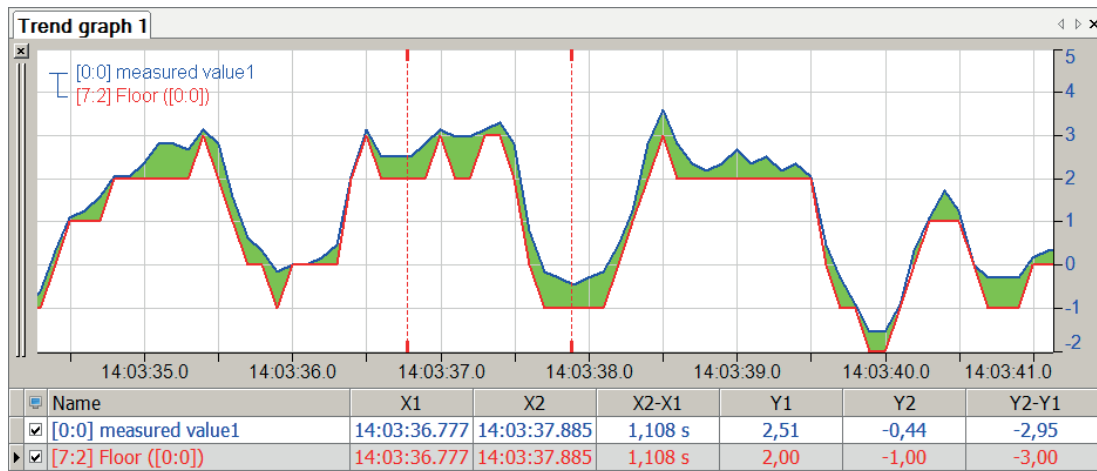
Abrunden der Werte eines Signalverlaufs auf ganze Zahlen

#### Aufgabenstellung

Von einem realen Signalverlauf soll für jedes Sample der Messwert auf den nächsten ganzzahligen Wert abgerundet werden.

## Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve das ursprüngliche Signal und die rote Kurve zeigt Signalverlauf mit abgerundeten Werten.



## 2.3.9 Int

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

### Argumente

'Expression'	Messwert	
'Reset'	Optionaler digitaler Parameter, der zum Rücksetzen des Integrals bzw. Unterdrücken des Integrationsvorgangs verwendet werden kann. '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 ( $y \cdot dt$ ) von 'Expression' zurück. Der Parameter 'Reset' kann zum Rücksetzen des Integrals bzw. Unterdrücken des Integrationsvorgangs verwendet werden, z. B. um bei periodischen oder reversierenden Vorgängen dasselbe Signal mehrfach zu integrieren. 'Reset' kann selbst auch ein Ausdruck sein.

Beispiele:

<code>Int([0:0])</code>	Es erfolgt kein Reset ('Reset' weggelassen)
<code>Int([0:0], If(Mod(T(),20)=0,TRUE(),FALSE()))</code>	Das Integral wird alle 20 Sekunden zurückgesetzt.
<code>Int([0:0], [3.1])</code>	z. B. mit <code>[3.1] = If([0:0]&gt;10, 1, 0)</code> Das Integral wird zurückgesetzt, sobald der Ausdruck [3.1] TRUE zurückgibt, also wenn der Ausdruck [0:0] den Grenzwert 10 überschreitet.

## Beispiel 1

### Tipp



Diese Funktion kann in einem virtuellen Remanenzmodul verwendet werden. Ihre Ergebniswerte können damit über ein Stoppen und Neustarten der Messung erhalten werden.

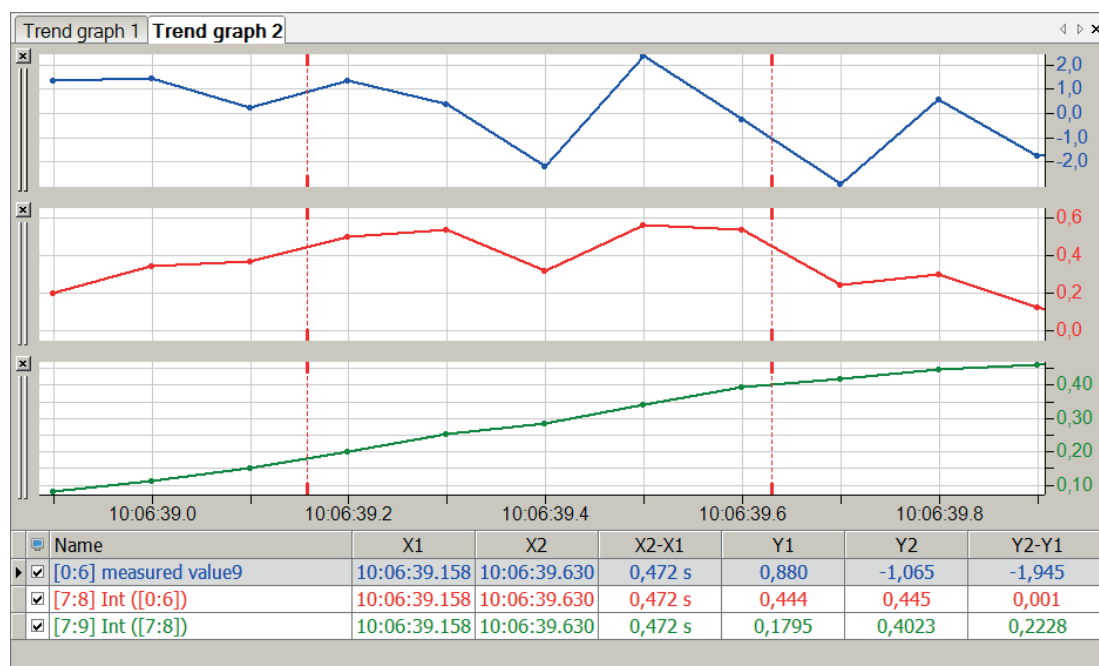
Ist 'Expression' ein Beschleunigungssignal so kann durch iteratives Ausführen der Int-Funktion daraus der zurückgelegte Weg ermittelt werden.

### Aufgabenstellung

Mithilfe eines Beschleunigungssensors sollen die Geschwindigkeit sowie der zurückgelegte Weg bestimmt werden

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve die gemessene Beschleunigung, die rote Kurve den berechneten Verlauf der Geschwindigkeit und die grüne Kurve die berechnete zurückgelegte Wegstrecke.



### Tipp



Ist nur die zurückgelegte Wegstrecke von Interesse, so kann diese auch durch rekursives Aufrufen der Funktion `Int(Int([0:6]))` ermittelt werden.

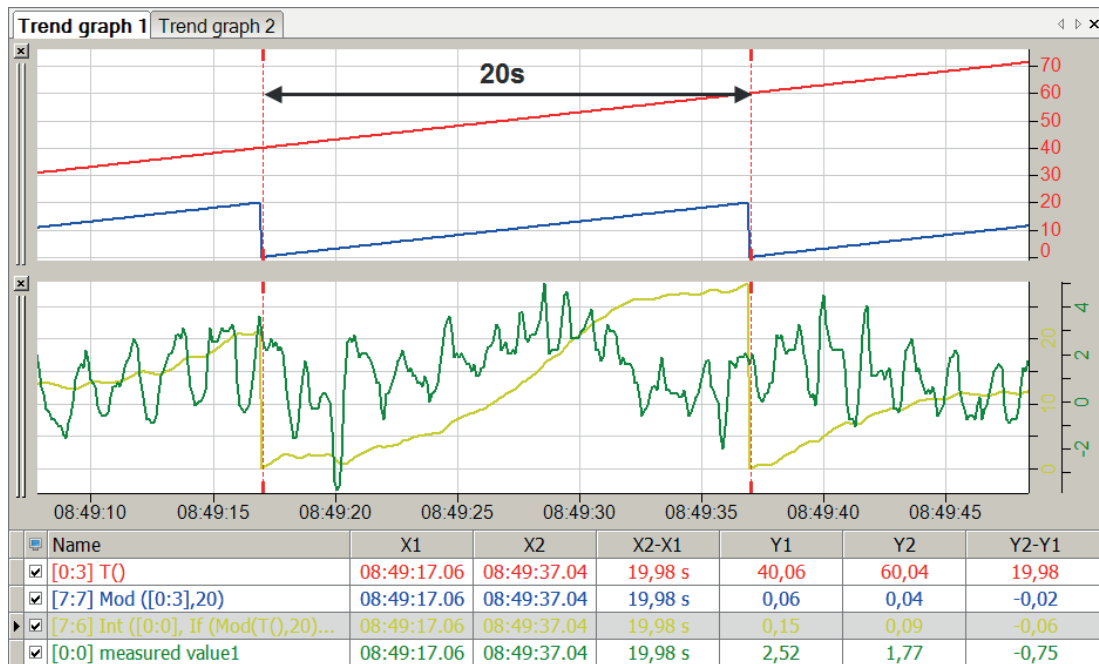
## Beispiel 2

Das Integral soll in einem Intervall von 20 Sekunden neu gestartet werden

### Aufgabenstellung

Die Modulo-Funktion kann genutzt werden, um das Integral zurückzusetzen, da alle 20 Sekunden gilt  $T() \bmod 20 = 0$ .

### Lösung



Rot	Zeitfunktion	Blau	Modulo20 der Zeitfunktion
Grün	Messwert	Gelb	Integral des Messwertes mit 20 Sekunden-Reset

### 2.3.10 Log

`Log('Expression')`

#### Beschreibung

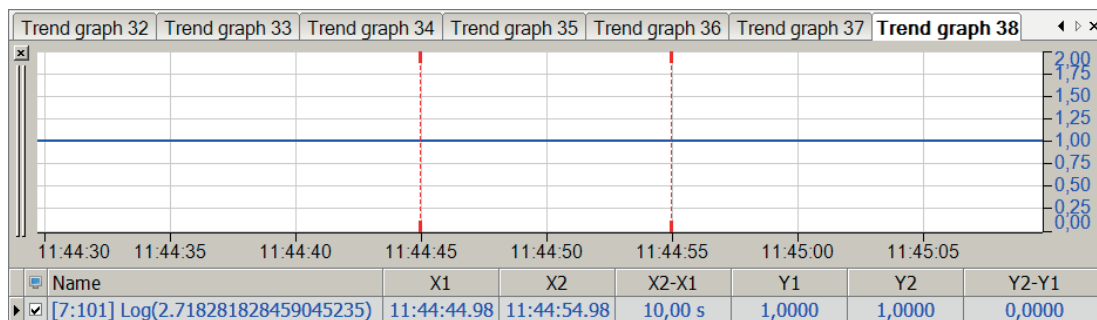
Diese Funktion liefert als Ergebnis den natürlichen Logarithmus ( $\ln x$ ) von 'Expression'.

#### Beispiel

Berechnung eines bekannten natürlichen Logarithmus.

#### Lösung

Logarithmus von der Eulerschen Zahl  $e$  muss konstant 1 ergeben:  $\ln e = 1$



#### Tipp



Negative Werte für 'Expression' geben zwar keine Fehlermeldung, allerdings auch kein Ergebnis.

### 2.3.11 Log10

`Log10('Expression')`

#### Beschreibung

Diese Funktion liefert als Ergebnis den dekadischen Logarithmus ( $\lg x$ ) von 'Expression'.

#### Beispiel

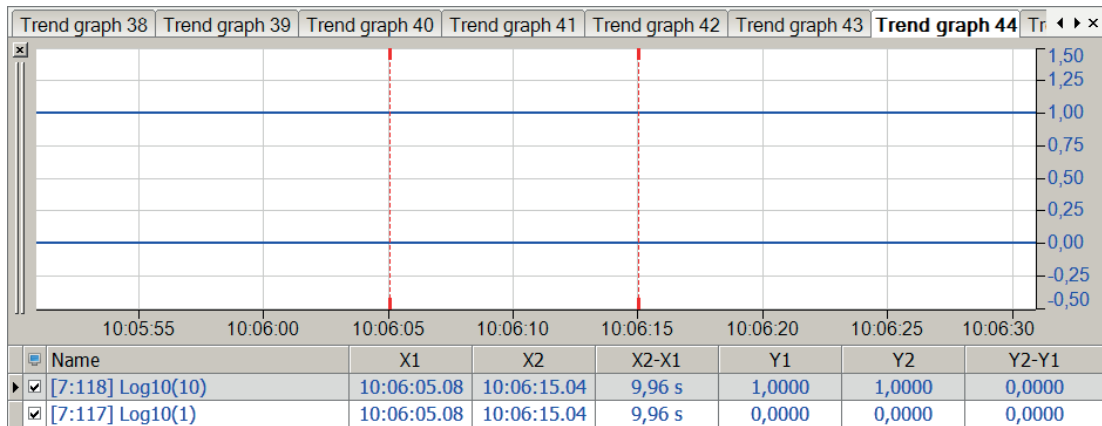
Berechnung von bekannten dekadischen Logarithmen.

#### Lösung

$$\lg 1 = \text{Log10}(1) = 0$$

$$\lg 10 = \text{Log10}(10) = 1$$



**Tipp**

Negative Werte für 'Expression' geben zwar keine Fehlermeldung, allerdings auch kein Ergebnis.

**2.3.12 Mod**

`Mod('Expression1', 'Expression2')`

**Beschreibung**

Diese Funktion liefert als Ergebnis den Modulo von 'Expression1' und 'Expression2'. Die Funktion verwendet intern die C-Funktion `fmod`, die es gestattet, Gleitkommawerte für 'Expression1' und 'Expression2' zu verwenden.

Der Modulo  $r$  ist der Rest der Division  $\text{Ausdruck1} / \text{Ausdruck2}$ , so dass umgekehrt gilt:

$\text{Ausdruck1} = \text{Ausdruck2} * i + r$ , wobei  $i$  eine ganze Zahl (Integer) ist.

Modulo  $r$  hat stets das gleiche Vorzeichen wie 'Expression1' und der Absolutwert von  $r$  ist stets kleiner als der Absolutwert von 'Expression2'.

Wenn 'Expression1' < 'Expression2' ist, dann liefert die Funktion den Wert 'Expression1' als Ergebnis. Der Rest kann in der mathematischen Sprache auch als "Ausdruck1 modulo Ausdruck2" bezeichnet werden.

Beispiele:  $\text{Mod}(7, 3) = 1$ ; Sieben durch drei ist gleich zwei, Rest 1.

$\text{Mod}(20.0, 10.5) = 9.5$

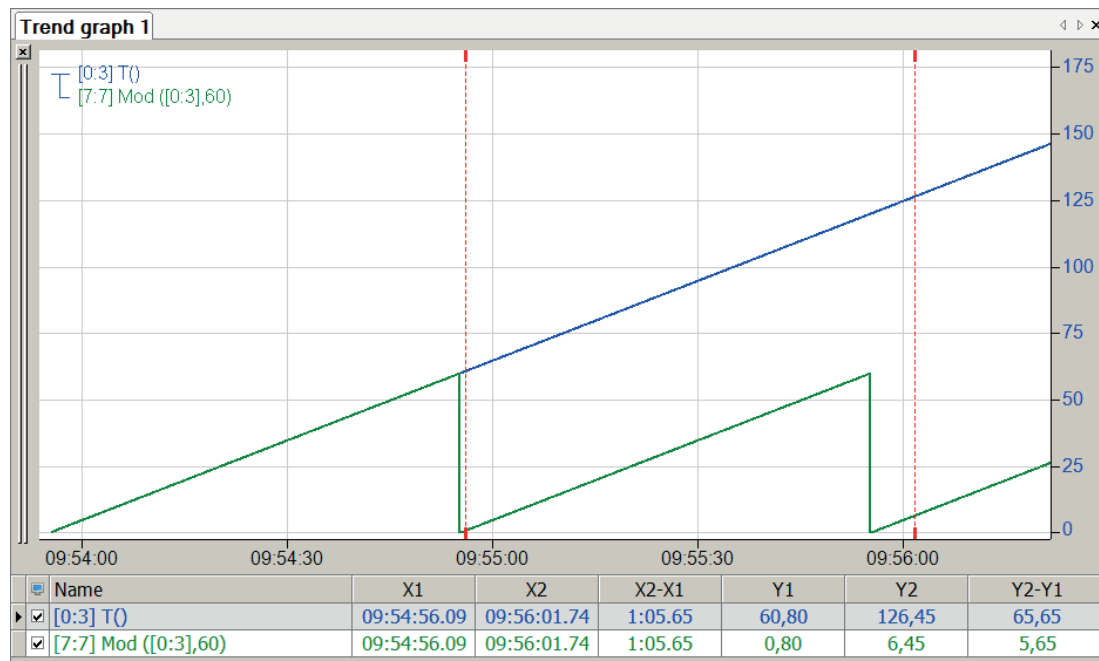
### Beispiel

Von der Zeitfunktion sind nur die Sekunden relevant, Stunden und Minuten sollen abgeschnitten werden

### Lösung

Mithilfe des Modulo60 der Zeitfunktion bleiben nur die Sekunden als Rest.

In der nachfolgenden Abbildung zeigt die blaue Kurve die Zeitfunktion und die grüne Kurve zeigt Modulo60 der Zeitfunktion.



### 2.3.13 Pow

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

#### Argumente

'Expression1'	Basis
'Expression2'	Exponent

#### Beschreibung

Diese Funktion potenziert 'Expression1' (Basis) mit 'Expression2' (Exponent): ('Expression1')^Expression2'

### Beispiel

Berechnung einiger wichtiger Potenzen

**Lösung**

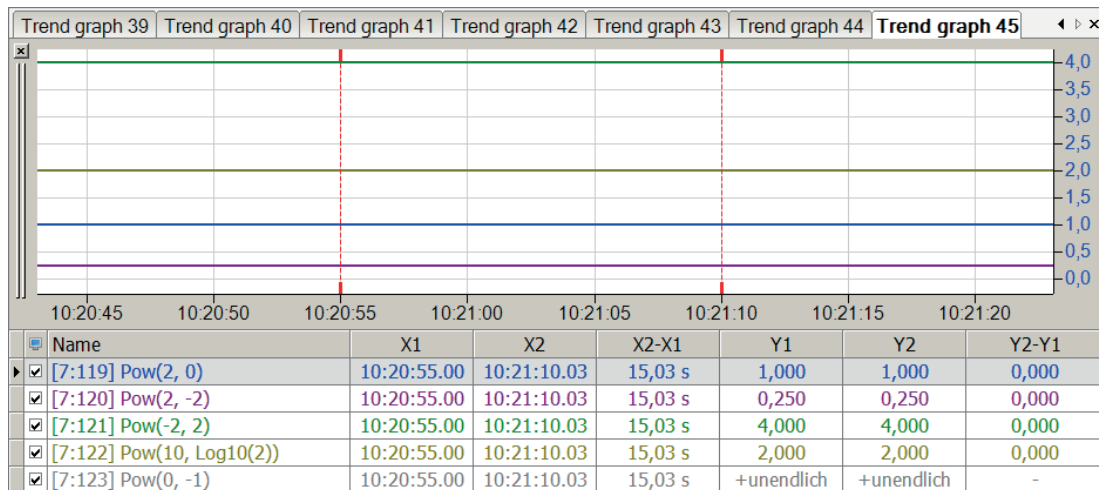
$$(2)0 = \text{Pow}(2, 0) = 1$$

$$(2)^{-2} = \text{Pow}(2, -2) = 0,25$$

$$(-2)^2 = \text{Pow}(-2, 2) = 4$$

$$(10)\lg 2 = \text{Pow}(10, \lg 2) = 2$$

$$(0)^{-1} = \text{Pow}(0, -1) = +\infty \text{ (unendlich)}$$

**Tipp**

Die Potenzierung von 0 mit -1, was einem Teilen durch 0 entspricht, gibt keine Fehlermeldung sondern den Grenzwert +unendlich zurück.

**2.3.14 Round**

`Round('Expression')`

**Beschreibung**

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

**Beispiel**

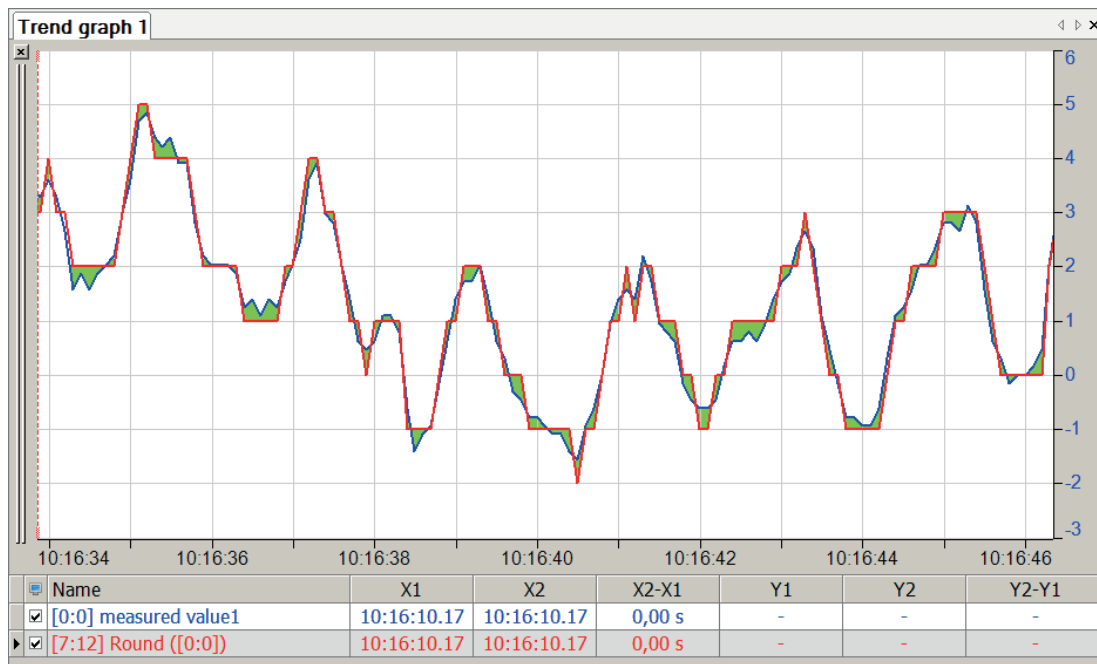
Runden der Werte eines Signalverlaufs auf ganze Zahlen

**Aufgabenstellung**

Von einem realen Signalverlauf soll für jedes Sample der Messwert auf den nächsten ganzzahligen Wert gerundet werden.

**Lösung**

In der nachfolgenden Abbildung zeigt die blaue Kurve das ursprüngliche Signal und die rote Kurve zeigt Signalverlauf mit abgerundeten Werten.



### 2.3.15 Sqrt

`Sqrt('Expression')`

#### Beschreibung

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

#### Beispiel

Berechnung einiger bekannter Quadratwurzeln

#### Lösung

$$\sqrt{4} = \text{Sqrt}(4) = 2$$

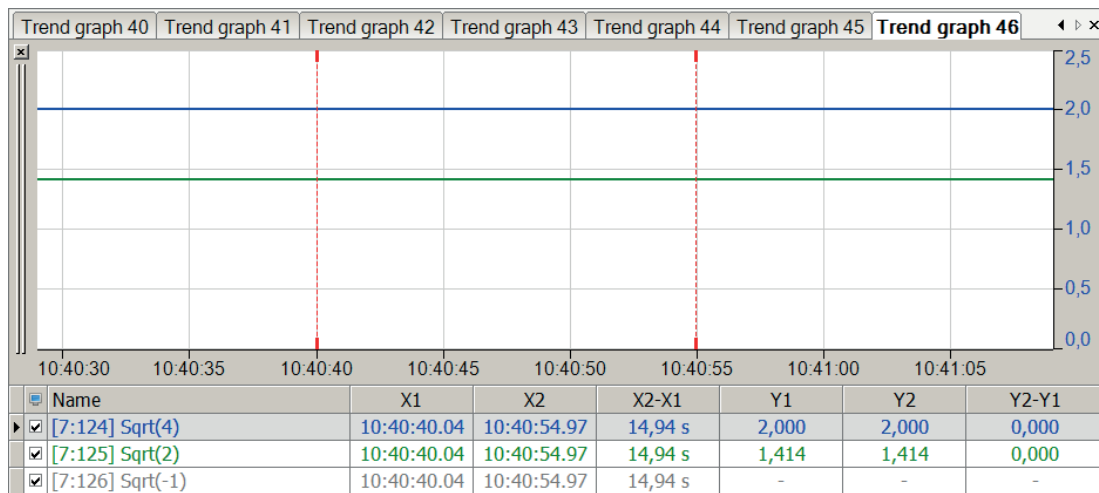
$$\sqrt{2} = \text{Sqrt}(2) = 1,41421356...$$

$$\sqrt{-1} = \text{Sqrt}(-1) = i \text{ (komplexe Rechnung notwendig)}$$

#### Tipp



Negative Werte für 'Expression' geben zwar keine Fehlermeldung, allerdings auch kein Ergebnis.



### 2.3.16 Truncate

`Truncate('Expression')`

#### Beschreibung

Die Funktion Truncate schneidet die Nachkommastellen eines Gleitkommawertes ab. Negative Zahlen werden somit auf den nächsten Integer-Wert aufgerundet, positive Zahlen abgerundet.

#### Beispiel

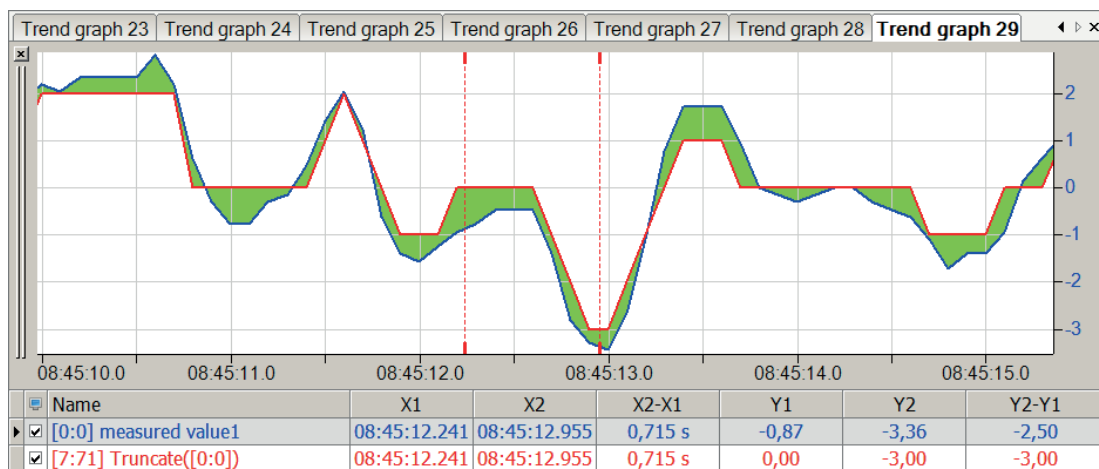
Abschneiden der Nachkommastellen der Werte eines Signalverlaufs

#### Aufgabenstellung

Von einem realen Signalverlauf soll für jedes Sample der Messwert ohne Nachkommastellen ausgegeben werden.

#### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve das ursprüngliche Signal und die rote Kurve zeigt den Signalverlauf mit abgeschnittenen Nachkommastellen.



## 2.3.17 Trigonometrische Funktionen

`Funktion('Expression')`

### 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.

Funktion	Beschreibung
<code>Sin('Expression')</code>	Diese Funktion liefert als Ergebnis den Sinus von 'Expression' in rad.
<code>Cos('Expression')</code>	Diese Funktion liefert als Ergebnis den Cosinus von 'Expression' in rad.
<code>Tan('Expression')</code>	Diese Funktion liefert als Ergebnis den Tangens von 'Expression' in rad.
<code>Asin('Expression')</code>	Diese Funktion liefert als Ergebnis den Arkussinus von 'Expression' in rad.
<code>Acos('Expression')</code>	Diese Funktion liefert als Ergebnis den Arkuscosinus von 'Expression' in rad.
<code>Atan('expression')</code>	Diese Funktion liefert als Ergebnis den Arkustangens von 'Expression' in rad.

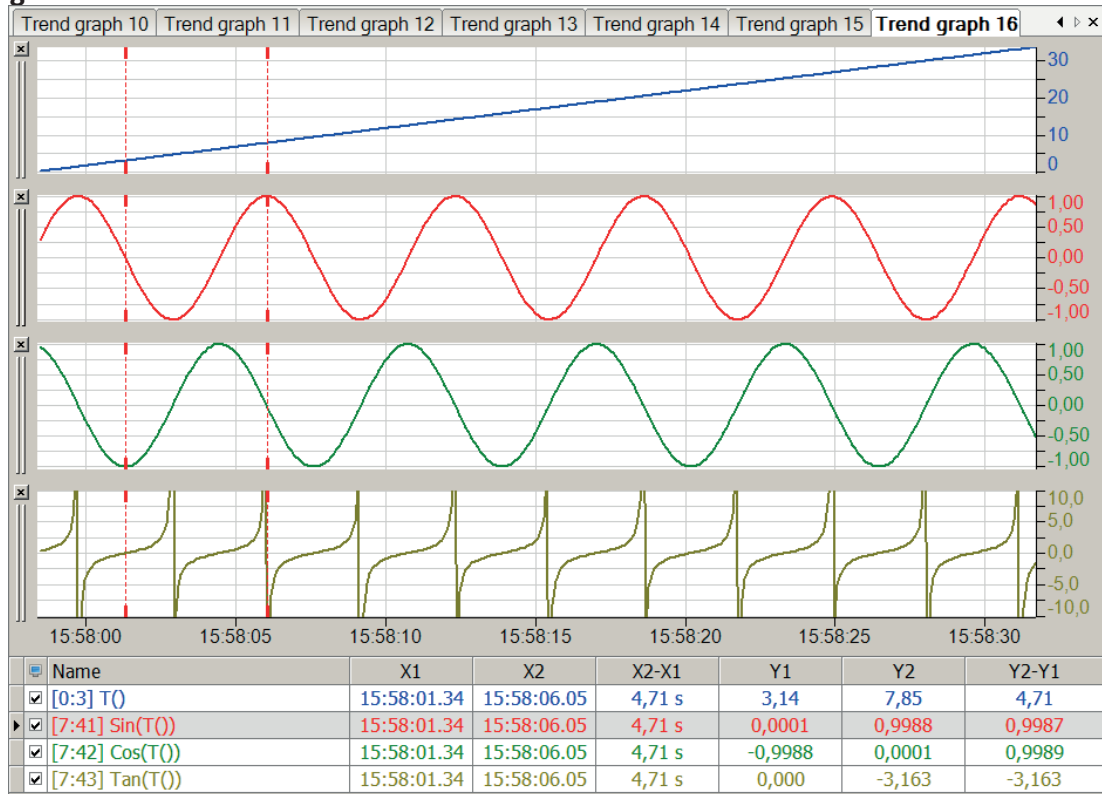
### Tipp



Zur Erzeugung von Signalen kann auch die `GenerateSignal`-Funktion verwendet werden.

**Beispiel**

Darstellung der trigonometrischen Funktionen

**Lösung**

Blau	Zeitfunktion T() als Basis für trigonometrische Funktionen	Rot	Sinusfunktion der Zeit
Grün	Cosinusfunktion der Zeit	Gelb	Tangensfunktion der Zeit

Pi ( )

**Beschreibung**

Die Zahl Pi ist als Konstante ( $p = 3,1415927\dots$ ) für diverse Berechnungen im System hinterlegt. Mit dieser Funktion fügen Sie die Zahl Pi in Ihre Berechnung ein.

## 2.4 Statistische Funktionen

### 2.4.1 Avg

`Avg('Expression', 'Reset=0')`

#### Argumente

'Expression'	Messwert für den der Mittelwert gebildet wird	
'Reset'	Optional Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung:	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten und Ergebnis auf Momentanwert von 'Expression' setzen

#### Beschreibung

Diese Funktion liefert als Ergebnis den Mittelwert von 'Expression' seit Beginn der Messung bzw. seit dem letzten Reset. Das Signal wird in den Arbeitsspeicher geschrieben. Der arithmetische Mittelwert wird kontinuierlich berechnet. Wenn 'Reset' = 1 (TRUE) ist, dann ist das Ergebnis gleich dem Istwert von 'Expression'.

#### Hinweis



Das Ergebnis der Avg-Funktion wird erst im nachfolgenden Intervall ausgegeben.



## 2.4.2 Avg2

Avg2('Expression1', 'Expression2', ...)

### Argumente

'Expression'	Messwert, für den der Durchschnittswert gebildet wird
--------------	---

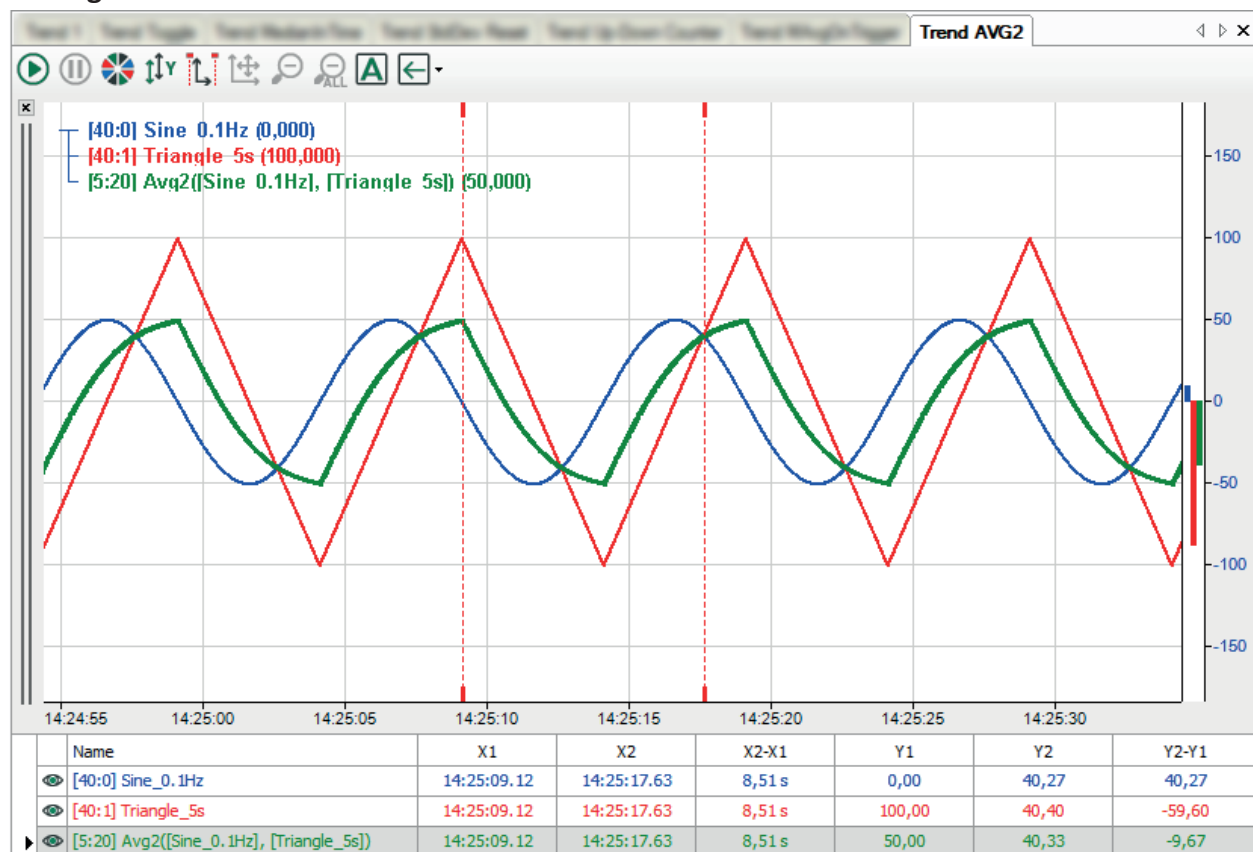
### Beschreibung

Diese Funktion liefert als Ergebnis den Durchschnittswert aller Argumente 'Expression1', 'Expression2', 'Expression3' usw. In jedem Erfassungszyklus wird aus den Momentanwerten aller Argumente der arithmetische Mittelwert gebildet. Bis zu 1000 Argumente sind zulässig.

### Beispiel

Aus zwei Signalen, einem Sinus- und einem Dreiecksignal, soll der Mittelwert gebildet werden.

### Lösung



### 2.4.3 AvgInTime

```
AvgInTime('Expression','Interval',' Reset=0')
```

#### Argumente

'Expression'	Messwert für den der Mittelwert gebildet wird	
'Interval'	Angabe der Intervalllänge in Sekunden	
'Reset'	Optional Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung:	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset' = 2	Berechnung anhalten und Ergebnis beibehalten
	'Reset' = 3	Jetzt berechnen und danach Berechnung anhalten

#### Beschreibung

Diese Funktion liefert als Ergebnis den Mittelwert je Zeitabschnitt der Länge 'Interval' von 'Expression'. Das Signal wird in den Arbeitsspeicher geschrieben. Nach Ablauf eines Intervalls wird der arithmetische Mittelwert für dieses Intervall berechnet.

#### Tipp



Das Ergebnis der AvgInTime-Funktion wird erst im nachfolgenden Intervall ausgegeben.

#### Beispiel

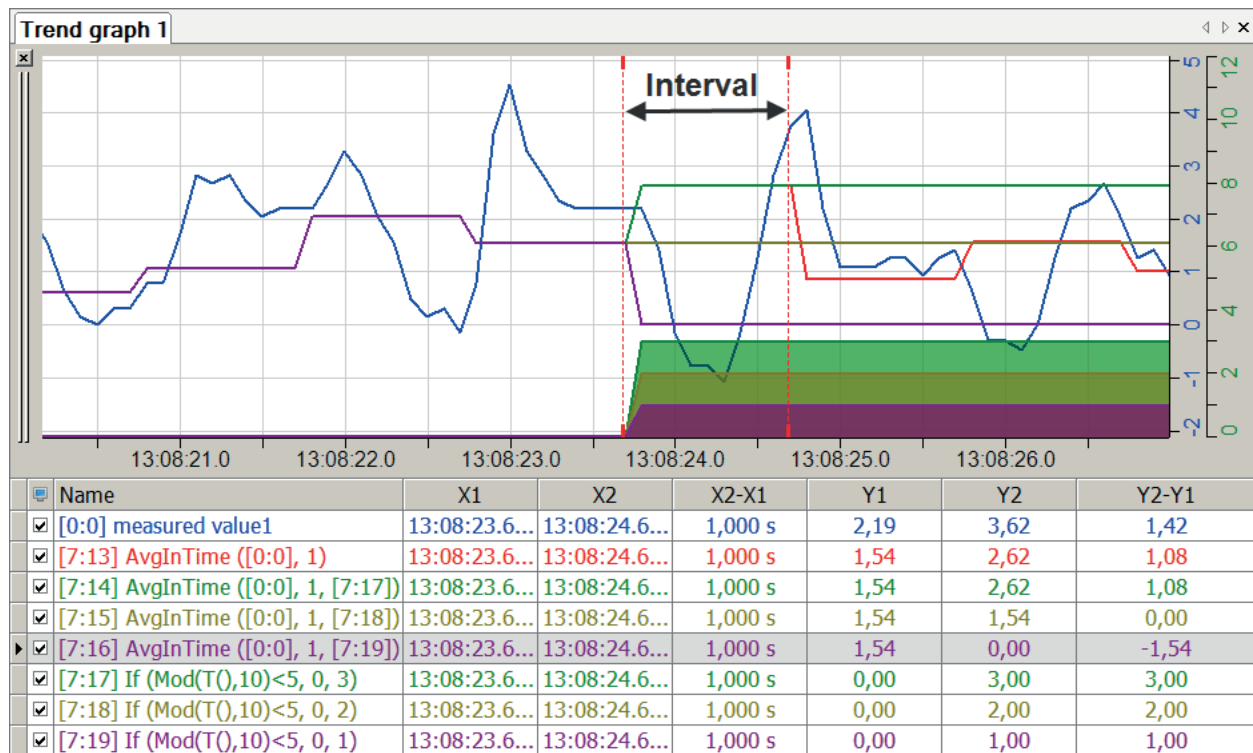
Auswirkungen des Parameters 'Reset'

#### Aufgabenstellung

Der Parameter 'Reset' soll in 5 Sekunden Abständen von 0 auf 1, 2, oder 3 geschaltet werden, um die Auswirkungen des Parameters sichtbar zu machen.

#### Lösung

Drei If-Abfragen geben über eine Modulus-Funktion der Zeit in 5 Sekunden Intervallen den Wert 0 als TRUE und die Werte 1, 2, 3 als FALSE an den 'Reset'-Parameter der AvgInTime-Funktion weiter.



Rot	Durchgehende Berechnung ohne 'Reset'-Angabe	Grün	('Reset' = 3) Wert wird vor Umschaltung berechnet
Gelb	('Reset' = 2) Wert bleibt konstant während Umschaltung	Lila	('Reset' = 1) Wert =0 während Umschaltung

## 2.4.4 KurtosisInTime

KurtosisInTime('Expression','Interval',' Reset=0')

### Argumente

'Expression'	Messwert, für den die Kurtosis gebildet wird	
'Interval'	Angabe der Länge des Intervalls in Sekunden, über das die Kurtosis berechnet werden soll.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten und Ergebnis beibehalten
	'Reset'=3	Jetzt berechnen und danach Berechnung anhalten

### Beschreibung

Die Berechnung der Kurtosis (Wölbung) 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 ergebenden Verteilungsfunktion. Schädigungen mit ausgeprägtem diskretem Charakter lassen den Kurtosis-Faktor somit stark ansteigen. Sein Absolutwert lässt somit bereits Aussagen über eine Schädigung zu.

## 2.4.5 MAvg

`MAvg('Expression', 'WindowInterval', 'UpdateInterval=timebase', 'Reset=0')`

### Argumente

'Expression'	Messwert, für den der Mittelwert gebildet wird	
'WindowInterval'	Angabe der Länge des Intervalls in Sekunden, über das der Mittelwert gebildet wird; muss ein Vielfaches von 'UpdateInterval' sein.	
'UpdateInterval'	Optionaler Parameter (Voreinstellung = Zeitbasis); gibt vor, in welchem Takt die Berechnung durchgeführt wird.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung.	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten und Ergebnis beibehalten

### Beschreibung

Diese Funktion liefert als Ergebnis den gleitenden arithmetischen Mittelwert von 'Expression' berechnet über ein 'WindowInterval' in Sekunden. Die Berechnung wird im Takt 'UpdateInterval' ausgeführt. 'UpdateInterval' ist ein optionaler Parameter und wird in Sekunden angegeben. Wenn 'UpdateInterval' nicht angegeben wird, dann wird er gleich der Zeitbasis gesetzt (Default), d. h. so klein wie möglich. Die Berechnung erfolgt dann fortschreitend um jeweils einen Messpunkt. Die Berechnung kann auch in größeren Abständen erfolgen, wenn für 'UpdateInterval' ein (vielfacher) Wert der Zeitbasis eingegeben wird.

'WindowInterval' bestimmt den Zeitraum, für den der Mittelwert jedes Mal berechnet wird. 'WindowInterval' muss ein Vielfaches von 'UpdateInterval' sein. Wenn nicht, dann wird 'WindowInterval' automatisch geändert, zum ersten Vielfachen von 'UpdateInterval' größer als oder gleich 'WindowInterval'.

### 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.

## Beispiel

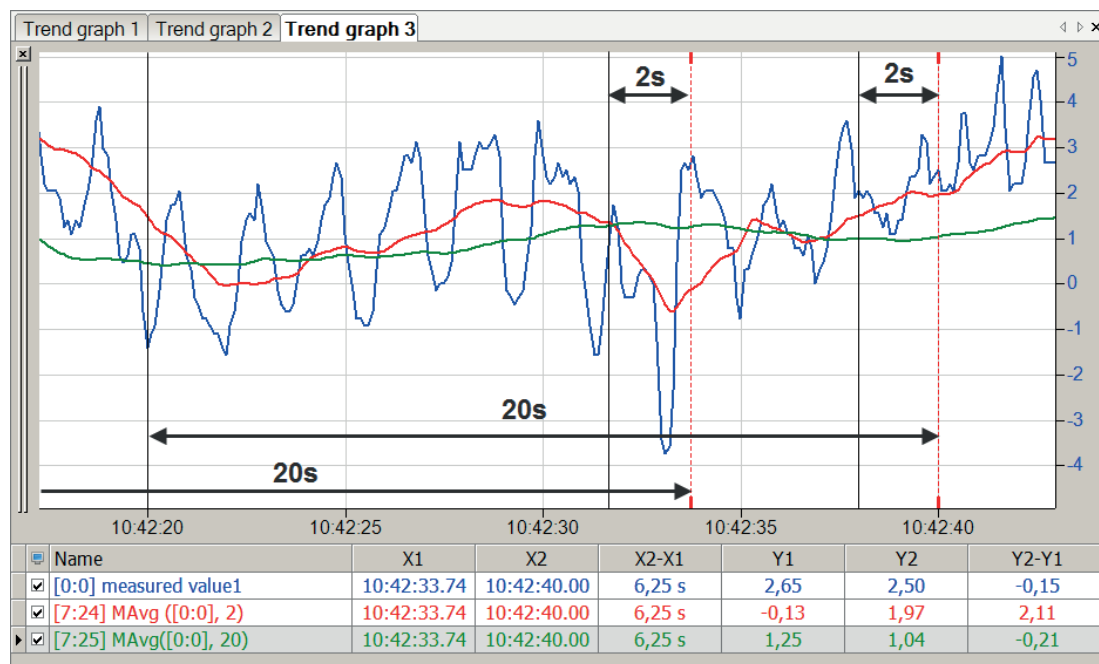
Für einen Messwert sind die Mittelwerte der vorangegangenen 2 und 20 Sekunden relevant.

## Aufgabenstellung

Die MAVg-Funktion wird einmal mit einer Intervallzeit von 2 Sekunden und einmal mit einer Intervallzeit von 20 Sekunden aufgerufen.

## Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert, die rote Kurve zeigt den Mittelwert über ein Intervall von zwei Sekunden und die grüne Kurve zeigt den Mittelwert über ein Intervall von 20 Sekunden.



## 2.4.6 MAvgOnTrigger

`MAvgOnTrigger('Expression', 'Trigger', 'NumberOfValues*', 'Reset=0')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Argumente

'Expression'	Messwert, für den der Mittelwert gebildet wird	
'Trigger'	Digitales Signal oder Ausdruck als Auslöser für die Ausführung des Befehls	
'NumberOf- Vlaues*'	Anzahl der Samples, die für die Berechnung des Durchschnitts verwendet werden;  Parameter wird zum Start der Erfassung übernommen, daher festen Wert und keinen Ausdruck eintragen.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung.	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten, alle gepufferten Daten löschen und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten und Ergebnis beibehalten

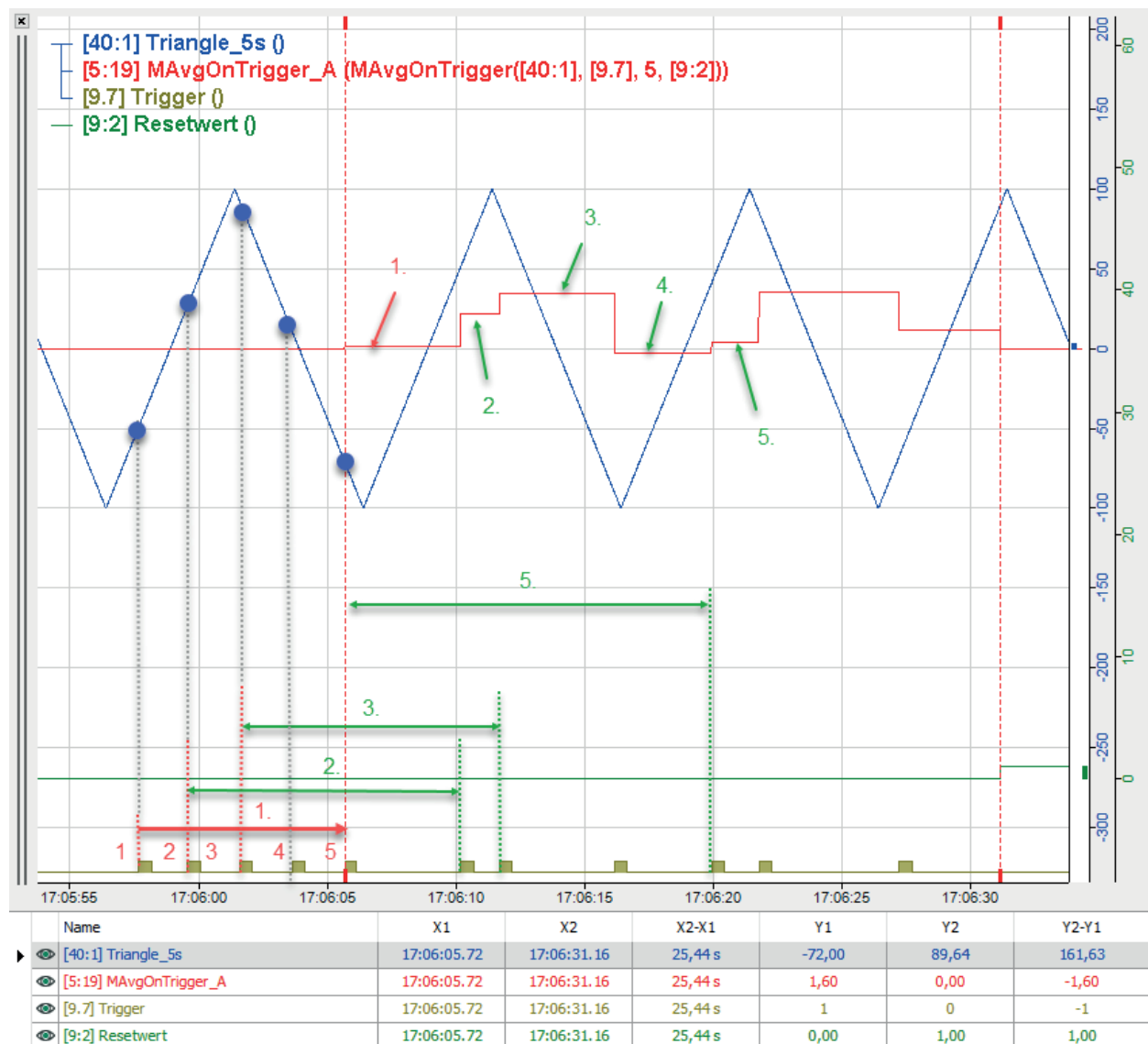
### Beschreibung

Diese Funktion liefert als Ergebnis den gleitenden arithmetischen Mittelwert von 'Expression', der bei jeder steigenden Flanke von 'Trigger' berechnet wird. In die Berechnung fließen nur so viele Werte ein, wie mit 'NumberOfValues' vorgegeben wurde. Nach dem Start der Erfassung oder nach einem Zurücksetzen mit 'Reset' =1 muss 'Trigger' zunächst 'NumberOfValues' mal auslösen, bevor der erste Mittelwert berechnet wird. Mit jedem weiteren 'Trigger' wird der Mittelwert für die jeweils letzten 'NumberOfValues' getriggerten Samples berechnet.

Damit ist es möglich, gleitende Mittelwerte ereignisgesteuert berechnen zu lassen. Die Berechnung kann dann z. B. pro Umdrehung einer Maschine oder pro gefertigtes Produkt erfolgen, anstatt nur linear über die Zeit.

## Beispiel

Anwendung von MAvgOnTrigger auf ein Dreieckssignal.



Zur Veranschaulichung wurde im Beispiel nur eine 'NumberOfValues' = 5 gewählt.

Nach dem Start der Erfassung wird erst mit dem 5. Trigger ein erster Mittelwert (rot 1.) aus den 5 erfassten Samples (blaue Punkte) berechnet. Der Abstand der Triggersignale ist dabei unerheblich. Aus Darstellungsgründen wurden im Beispiel die ersten 5 Trigger dicht bei einander ausgelöst.

Mit jedem weiteren Trigger werden das dann aktuelle und die letzten 4 Samples in die Berechnung einbezogen und ergaben dann die entsprechenden Mittelwerte (grün 2., 3., 4., 5. usw.)



## 2.4.7 Max

`Max('Expression', 'Reset=0')`

### Argumente

'Expression'	Messwert, für den der Maximalwert ermittelt werden soll	
'Reset'	Optional digitaler Parameter, der zum Rücksetzen des Maximums verwendet werden kann, z. B. um Einpendelvorgänge des Messsignals in der Startphase zu ignorieren. 'Reset' kann selbst auch ein Ausdruck sein.	
	'Reset' = TRUE	Berechnung anhalten und Ergebnis auf Momentanwert von 'Expression' setzen.
	'Reset' = 0	Berechnung durchführen; zuletzt erkanntes Maximum wird ausgegeben. (Voreinstellung)

### Beschreibung

Diese Funktion liefert als Ergebnis den Maximalwert 'Expression'. Es wird als konstanter Wert (horizontale Linie) im Signalstreifen angezeigt. Jeder Wert wird mit dem vorherigen verglichen. Ist der neue Wert größer als der alte, wird der größere Wert in die Kurve eingetragen. Ist der neue Wert gleich oder kleiner als der alte, wird der alte Wert eingetragen. Mit dem digitalen Signal 'Reset' kann wahlweise die Maximalwert-Berechnung gestoppt und das Ergebnis wieder auf den Istwert des Eingangssignals zurückgesetzt werden. Ohne das Reset-Signal besteht keine Möglichkeit die Anzeige zurückzusetzen, es sei denn, die Messung wird angehalten und neu gestartet. 'Reset' kann auch als Ausdruck formuliert werden.

Beispiele:

<code>Max([0:0])</code>	Es erfolgt kein Reset.
<code>Max([0:0],If(Mod(T(),20)=0,TRUE(),FALSE()))</code>	Der Maximalwert wird alle 20 Sekunden zurückgesetzt.
<code>Max([0:0],[3.1])</code>	z. B. mit <code>[3.1] = If([0:0]&lt;10, 1, 0)</code> Der Maximalwert wird zurückgesetzt, sobald der Ausdruck <code>[3.1]</code> TRUE zurückgibt, also wenn der Ausdruck <code>[0:0]</code> den Grenzwert 10 unterschreitet..

### Beispiel

#### Tipp



Diese Funktion kann in einem virtuellen Remanenzmodul verwendet werden. Ihre Ergebniswerte können damit über ein Stoppen und Neustarten der Messung erhalten werden.

Für einen Messwert soll das Maximum ermittelt werden. Dabei soll die Start-Phase in der Berechnung unberücksichtigt bleiben.

## Aufgabenstellung

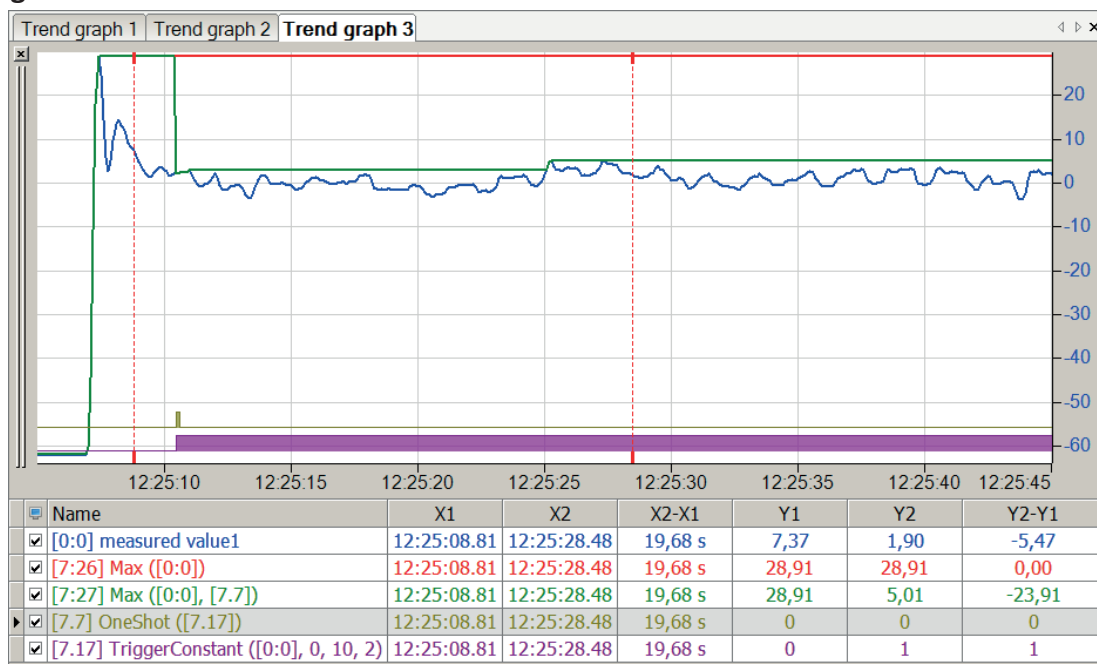
Um Ausschläge während der Startphase zu löschen wird der Maximalwert durch Nutzung der 'Reset'-Funktion zurückgesetzt. Dies kann über die TriggerConstant-Funktion erfolgen, die das Einpendeln des Messwertes abwartet. Das Setzen des 'Reset'-Parameters erfolgt über die Flanken-Erkennung ('OneShot') des Triggers.

### Hinweis



Der 'Reset'-Parameter darf nicht dauerhaft auf TRUE gesetzt werden, da der Maximalwert dadurch permanent zurückgesetzt werden und somit dem Messwert entsprechen würde.

## Lösung



Bla	Messwert	Rot	Verfälschter Maximalwert ohne 'Reset'
Grün	Maximalwert mit Reset nach Startphase	Gelb	One-Shot Funktion zur Auslösung des 'Reset'
Lila	Trigger nach der Startphase		

## 2.4.8 Max2

Max2('Expression1', 'Expression2', ...)

### Beschreibung

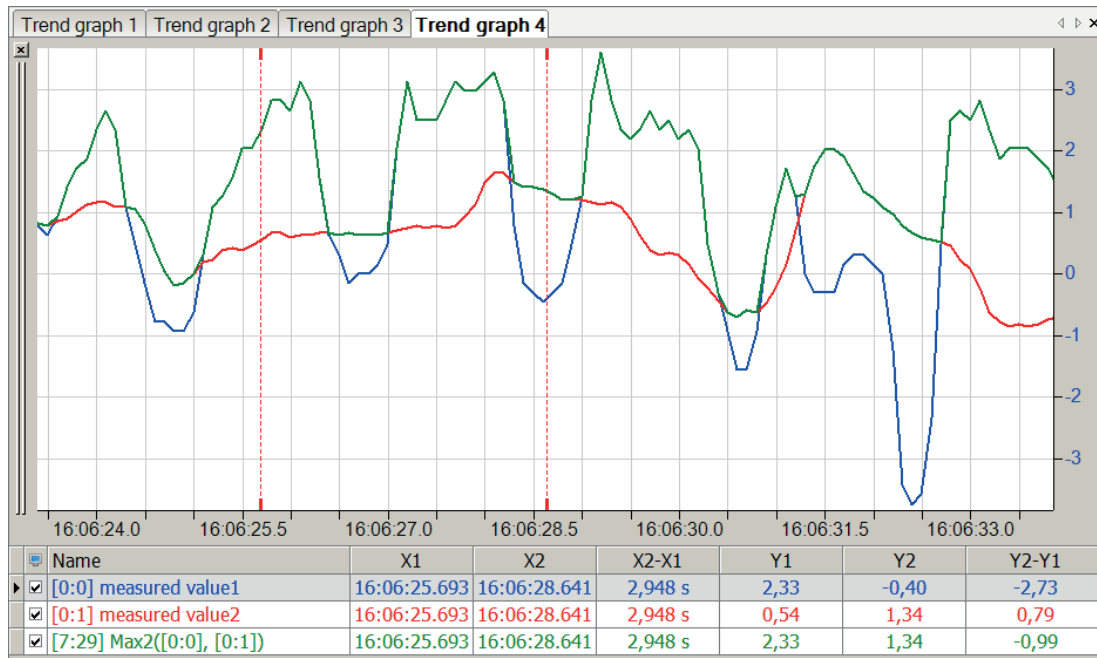
Diese Funktion liefert als Ergebnis das Maximum aller Ausdrücke 'Expression1', 'Expression2' usw. Die Ausdrücke bzw. Signale werden Messwert für Messwert verglichen und der jeweils größte Wert wird als Ergebnis übergeben. Bis zu 1000 Argumente sind zulässig.

## Beispiel

Von zwei Messwerten ist lediglich der größere Wert relevant.

## Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert 1, die rote Kurve zeigt den Messwert 2 und die grüne Kurve zeigt den Verlauf der Maximalwerte.



## 2.4.9 MaxInTime

MaxInTime('Expression', 'Interval', ' Reset=0')

### Argumente

'Expression'	Messwert, für den das Maximum gebildet wird	
'Interval'	Angabe der Länge des Intervalls in Sekunden, über das das Maximum berechnet werden soll.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten und Ergebnis beibehalten
	'Reset'=3	Jetzt berechnen und danach Berechnung anhalten

### Beschreibung

Diese Funktion liefert als Ergebnis das Maximum von 'Expression' innerhalb jedes Intervalls der Länge 'Interval' (in s).

**Hinweis**

Das Ergebnis der MaxInTime-Funktion wird erst im nachfolgenden Intervall ausgegeben.

**Beispiel**

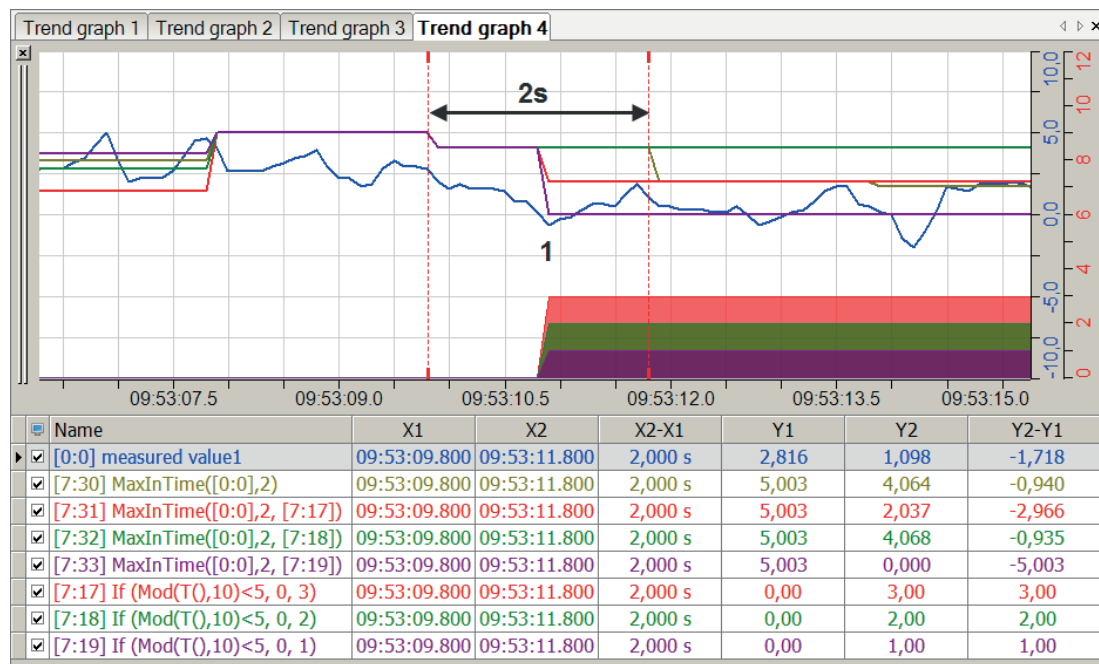
Auswirkungen des Parameters 'Reset'

**Aufgabenstellung**

der Parameter 'Reset' soll in 5-Sekunden Abständen von 0 auf 1, 2, oder 3 geschaltet werden um die Auswirkungen des Parameters sichtbar zu machen

**Lösung**

Drei If-Abfragen geben über eine Modulo-Funktion der Zeit in 5-Sekunden-Intervallen den Wert 0 als TRUE und die Werte 1, 2, 3 als FALSE an den 'Reset'-Parameter der MaxInTime-Funktion weiter.



Blau	Messwert	Gelb	Durchgehende Berechnung ohne 'Reset'-Angabe
Rot	('Reset'=3) Wert wird vor Unterbrechung berechnet, auch innerhalb eines Intervalls (1)	Grün	('Reset'=2) Wert bleibt konstant während Unterbrechung
Lila	('Reset'=1) Wert=0 während Unterbrechung, auch innerhalb eines Intervalls (1)		

## 2.4.10 Median2

Median2('Expression1', 'Expression2', ...)

### Argumente

'Expression'	Messwert(e), für den/die der Median gebildet wird
--------------	---

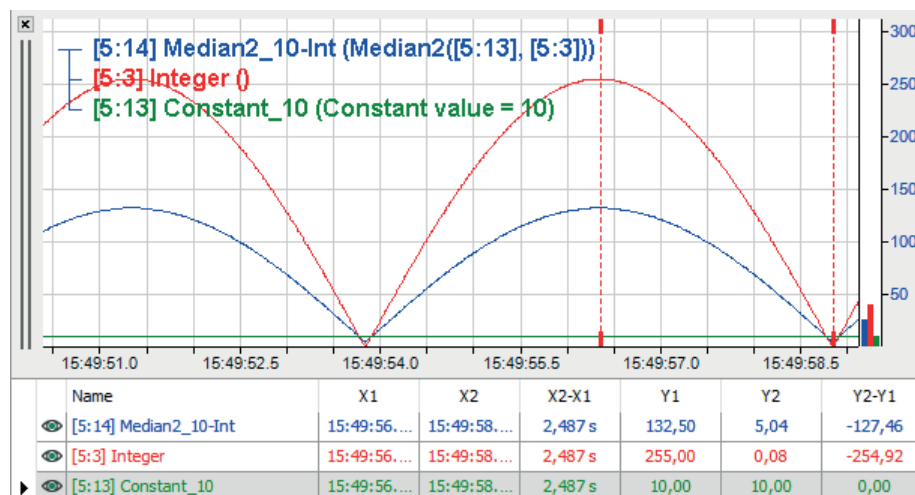
### Beschreibung

Diese Funktion liefert als Ergebnis den Median aller Argumente. Die Istwerte der Argumente werden in jedem Zyklus addiert und die Summe durch zwei geteilt.

### Beispiel

Von einer Konstanten und einem periodischen Wert soll der Median gebildet werden.

### Lösung



Das Beispiel zeigt am Maximum der Kurve einen Median von 132,5  $((10 + 255)/2)$ .

## 2.4.11 MedianInTime

MedianInTime('Expression', 'Interval', ' Reset=0')

### Argumente

'Expression'	Messwert, für den der Median gebildet wird	
'Interval'	Angabe der Intervalllänge in Sekunden	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung:	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset' = 2	Berechnung anhalten, alle gepufferten Daten löschen und Ergebnis beibehalten
	'Reset' = 3	Jetzt berechnen und danach Berechnung anhalten

## Beschreibung

Diese Funktion liefert als Ergebnis den Median je Zeitabschnitt der Länge 'Interval' von 'Expression'. Das Signal wird in den Arbeitsspeicher geschrieben. Nach Ablauf eines Intervalls wird der arithmetische Mittelwert für dieses Intervall berechnet.

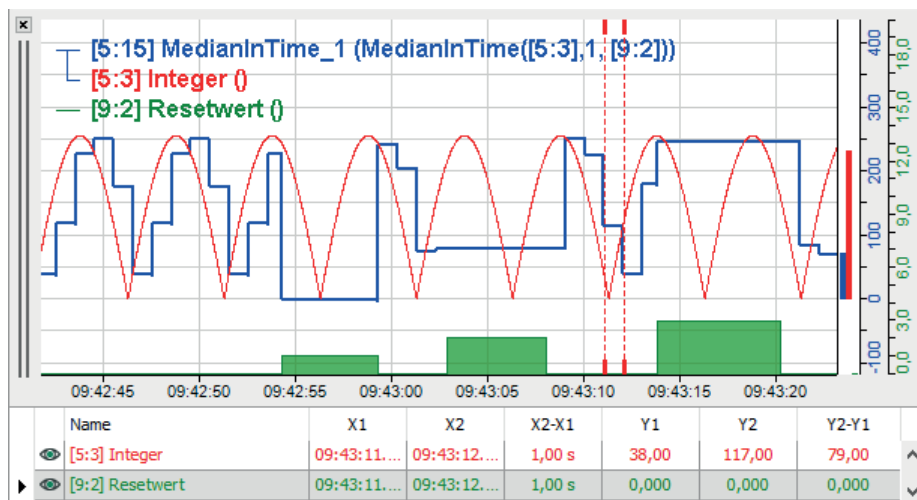
## Tipp



Das Ergebnis der MedianInTime-Funktion wird erst im nachfolgenden Intervall ausgegeben.

## Beispiel

Die nachfolgende Abbildung zeigt die Berechnung des MedianInTime (blau) von einem Signal (rot) und die Auswirkungen des Parameters 'Reset' (grün).



## 2.4.12 Min

`Min('Expression', 'Reset=0')`

### Argumente

'Expression'	Messwert, für den der Minimalwert ermittelt werden soll	
'Reset'	Optional digitaler Parameter, der zum Rücksetzen des Minimums verwendet werden kann, z. B. um Einpendelvorgänge des Messsignals in der Startphase zu ignorieren. 'Reset' kann selbst auch ein Ausdruck sein.	
	'Reset' = TRUE	Berechnung anhalten und Ergebnis auf Momentanwert von 'Expression' setzen.
	'Reset' = 0	Berechnung durchführen; zuletzt erkanntes Minimum wird ausgegeben. (Voreinstellung)

### Beschreibung

Diese Funktion liefert als Ergebnis den Minimalwert des Signals 'Expression'. Es wird als konstanter Wert (horizontale Linie) im Signalstreifen angezeigt. Jeder Wert wird mit dem vorherigen verglichen. Ist der neue Wert kleiner als der alte, trägt Min einen kleineren Wert in die Kurve ein. Ist der neue Wert gleich oder größer als der alte, wird der alte Wert eingetragen. Mit dem digitalen Signal 'Reset' kann wahlweise die Minimalwert-Berechnung gestoppt und das Ergebnis wieder auf den Istwert des Eingangssignals zurückgesetzt werden. Ohne das Reset-Signal besteht keine Möglichkeit die Anzeige zurückzusetzen, es sei denn, die Messung wird angehalten und neu gestartet. 'Reset' kann auch als Ausdruck formuliert werden.

Beispiele:

<code>Min([0:0])</code>	Es erfolgt kein Reset.
<code>Min([0:0],If(Mod(T(),20)=0,TRUE(),FALSE()))</code>	Der Minimalwert wird alle 20 Sekunden zurückgesetzt.
<code>Min([0:0],[3.1])</code>	z. B. mit <code>[3.1] = If([0:0]&lt;10, 1, 0)</code> Der Minimalwert wird zurückgesetzt, sobald der Ausdruck <code>[3.1]</code> TRUE zurückgibt, also wenn der Ausdruck <code>[0:0]</code> den Grenzwert 10 unterschreitet..

### Beispiel

#### Tipp



Diese Funktion kann in einem virtuellen Remanenzmodul verwendet werden. Ihre Ergebniswerte können damit über ein Stoppen und Neustarten der Messung erhalten werden.

Für einen Messwert soll das Minimum ermittelt werden. Dabei soll die Start-Phase in der Berechnung unberücksichtigt bleiben.

## Aufgabenstellung

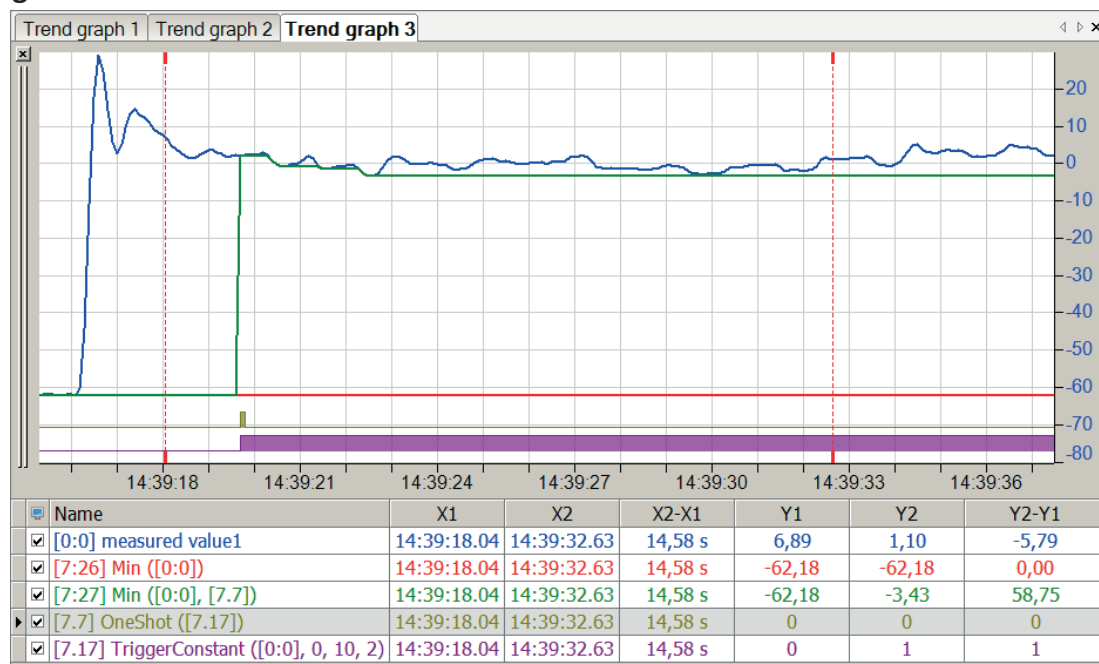
Um Ausschläge während der Startphase zu löschen wird der Minimalwert durch Nutzung der 'Reset'-Funktion zurückgesetzt. Dies kann über die TriggerConstant-Funktion erfolgen, die das Einpendeln des Messwertes abwartet. Das Setzen des 'Reset'-Parameters erfolgt über die Flanken-Erkennung ('OneShot') des Triggers.

### Hinweis



Der 'Reset'-Parameter darf nicht dauerhaft auf TRUE gesetzt werden, da der Minimalwert dadurch permanent zurückgesetzt werden und somit dem Messwert entsprechen würde.

## Lösung



Blau	Messwert	Rot	Verfälschter Minimalwert ohne 'Reset'
Grün	Minimalwert mit Reset nach Startphase	Gelb	One-Shot Funktion zur Auslösung des 'Reset'
Lila	Trigger nach der Startphase		

Abb. 1: Blaue Kurve: Messwert; rote Kurve: verfälschter Minimalwert ohne 'Reset'; grüne Kurve: Minimalwert mit Reset nach Startphase; gelber Balken: One-Shot Funktion zur Auslösung des 'Reset'; lila Balken: Trigger nach der Startphase



## 2.4.13 Min2

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

### Beschreibung

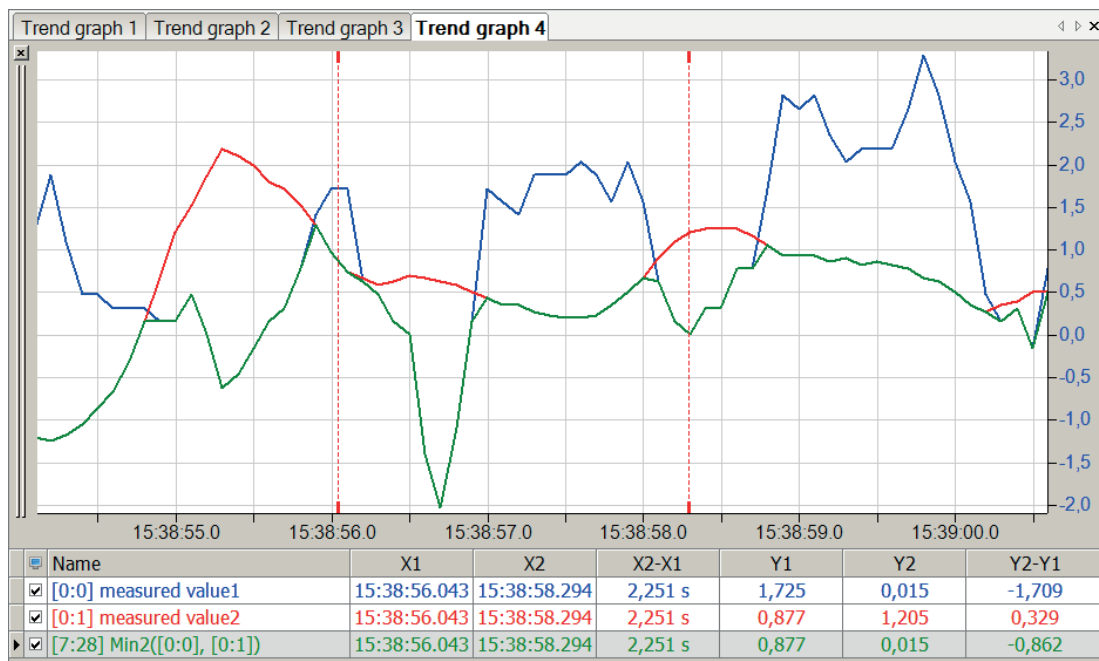
Diese Funktion liefert als Ergebnis das Minimum aller Ausdrücke 'Expression1', 'Expression2' usw. Die Ausdrücke bzw. Signale werden Messwert für Messwert verglichen und der jeweils kleinste wird als Ergebnis übergeben. Bis zu 1000 Argumente sind zulässig.

### Beispiel

Von zwei Messwerten ist lediglich der kleinere Wert relevant.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert 1, die rote Kurve zeigt den Messwert 2 und die grüne Kurve zeigt den Verlauf der Minimalwerte.



## 2.4.14 MinInTime

```
MinInTime('Expression','Interval',' Reset=0')
```

### Argumente

'Expression'	Messwert, für den das Minimum gebildet wird	
'Interval'	Angabe der Länge des Intervalls in Sekunden, über das das Minimum berechnet werden soll.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten und Ergebnis beibehalten
	'Reset'=3	Jetzt berechnen und danach Berechnung anhalten

### Beschreibung

Diese Funktion liefert als Ergebnis das Minimum von 'Expression' innerhalb jedes Intervalls der Länge 'Interval' (in s).

### Hinweis



Das Ergebnis der MinInTime-Funktion wird erst im nachfolgenden Intervall ausgegeben.

### Beispiel

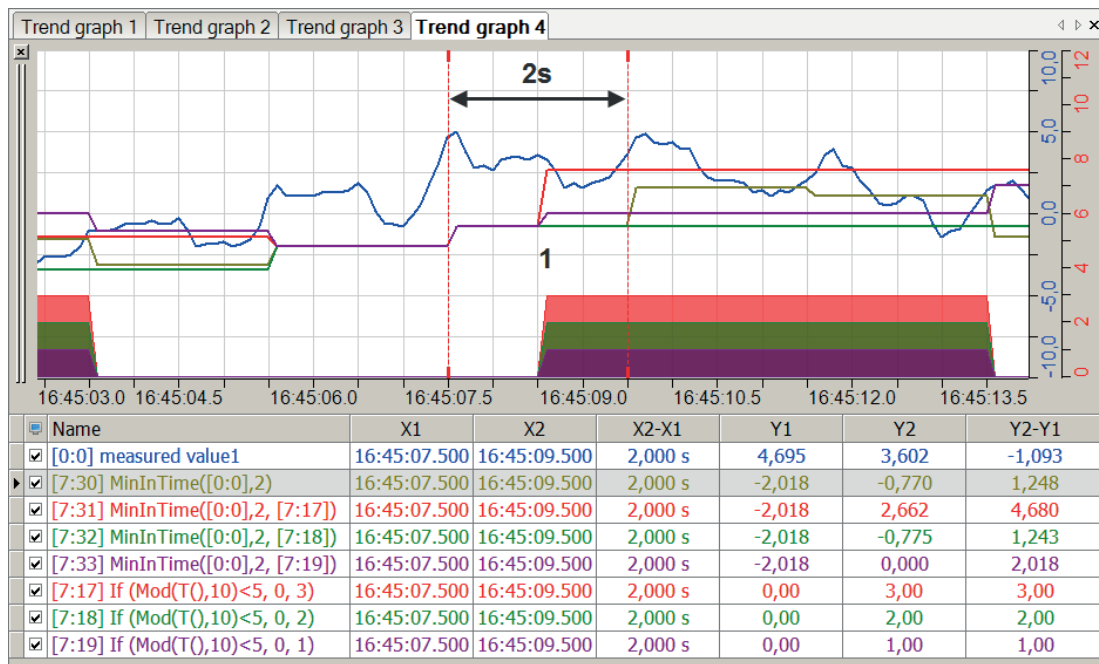
Auswirkungen des Parameters 'Reset'

### Aufgabenstellung

der Parameter 'Reset' soll in 5-Sekunden Abständen von 0 auf 1, 2, oder 3 geschaltet werden um die Auswirkungen des Parameters sichtbar zu machen

### Lösung

Drei If-Abfragen geben über eine Modulo-Funktion der Zeit in 5-Sekunden-Intervallen den Wert 0 als TRUE und die Werte 1, 2, 3 als FALSE an den 'Reset'-Parameter der MinInTime-Funktion weiter.



Blau	Messwert	Gelb	durchgehende Berechnung ohne 'Reset'-Angabe
Rot	('Reset'=3) Wert wird vor Unterbrechung berechnet, auch innerhalb eines Intervalls (1)	Grün	('Reset'=2) Wert bleibt konstant während Unterbrechung
Lila	('Reset'=1) Wert=0 während Unterbrechung, auch innerhalb eines Intervalls (1)		

### 2.4.15 MKurtosis

`MKurtosis('Expression', 'WindowInterval', 'UpdateInterval=timebase', 'Reset=0')`

#### Argumente

'Expression'	Messwert, für den die Kurtosis gebildet wird	
'WindowInterval'	Angabe der Länge des Intervals in Sekunden, über das die Kurtosis gebildet wird; muss ein Vielfaches von 'UpdateInterval' sein.	
'UpdateInterval'	Optionaler Parameter (Voreinstellung = Zeitbasis); gibt vor, in welchem Takt die Berechnung durchgeführt wird.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung.	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten und Ergebnis beibehalten

#### Beschreibung

Liefert den Kurtosis-Wert von 'Expression' alle 'UpdateInterval' Sekunden zurück, basierend auf einem gleitenden Fenster von 'WindowInterval' Sekunden.

Der Parameter 'UpdateInterval' ist optional. Wenn nicht angegeben, dann ist er gleich der Zeitbasis der Funktion (d. h. so klein wie möglich).

'WindowInterval' muss ein Vielfaches von 'UpdateInterval' sein. Wenn nicht, dann wird 'WindowInterval' automatisch geändert, zum ersten Vielfachen von 'UpdateInterval' größer als oder gleich 'WindowInterval'.

Für Informationen zur Kurtosis, siehe Kapitel [↗ KurtosisInTime](#), Seite 52

## 2.4.16 MMax

```
MMax('Expression', 'WindowInterval', 'UpdateInterval=timebase', 'Reset=0')
```

### Argumente

'Expression'	Messwert, für den das Maximum gebildet wird	
'WindowInterval'	Angabe der Länge des Intervalls in Sekunden, über das das Maximum berechnet werden soll; muss ein Vielfaches von 'UpdateInterval' sein.	
'UpdateInterval'	Optionaler Parameter (Voreinstellung = Zeitbasis); gibt vor, in welchem Takt die Berechnung durchgeführt wird.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten, zurücksetzen und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten, zurücksetzen und Ergebnis beibehalten

### Beschreibung

Diese Funktion liefert das Maximum von 'Expression' alle 'UpdateInterval' Sekunden zurück, basierend auf einem gleitenden Fenster von 'WindowInterval' Sekunden.

Der Parameter '*UpdateInterval*' ist optional. Wenn nicht angegeben, dann ist er gleich der Zeitbasis der Funktion (d. h. so klein wie möglich).

'WindowInterval' muss ein Vielfaches von '*UpdateInterval*' sein. Wenn nicht, dann wird 'WindowInterval' automatisch geändert, zum ersten Vielfachen von '*UpdateInterval*' größer als oder gleich 'WindowInterval'.

## 2.4.17 MMedian

```
MMedian('Expression', 'WindowInterval', 'UpdateInterval=timebase', 'Reset=0')
```

### Argumente

'Expression'	Messwert, für den der Median gebildet wird	
'WindowInterval'	Angabe der Länge des Intervalls in Sekunden, über das der Median berechnet werden soll; muss ein Vielfaches von 'UpdateInterval' sein.	
'UpdateInterval'	Optionaler Parameter (Voreinstellung = Zeitbasis); gibt vor, in welchem Takt die Berechnung durchgeführt wird.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung:	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten, alle gepufferten Daten löschen und Ergebnis auf 0 setzen
	'Reset' = 2	Berechnung anhalten, alle gepufferten Daten löschen und Ergebnis beibehalten

**Beschreibung**

Diese Funktion liefert als Ergebnis den Median von 'Expression' alle 'UpdateInterval'-Sekunden, basierend auf einem gleitenden Fenster von 'WindowInterval'-Sekunden.

Der Parameter 'UpdateInterval' ist optional. Wenn nicht angegeben, dann ist er gleich der Zeitbasis der Funktion (d. h. so klein wie möglich).

Der Parameter 'WindowInterval' muss ein Vielfaches von '*UpdateInterval*' sein. Wenn nicht, dann wird 'WindowInterval' automatisch geändert zum ersten Vielfachen von '*UpdateInterval*' größer als oder gleich 'WindowInterval'.

'WindowInterval' und 'UpdateInterval' können durch Ausdrücke vorgegeben werden. Änderungen dieser Intervallwerte werden erst nach einem Reset übernommen.

---

**Tipp**

Das Ergebnis der MMedian-Funktion wird erst im nachfolgenden Intervall ausgegeben.

---

## 2.4.18 MMin

```
MMin('Expression','WindowInterval','UpdateInterval=timebase','Reset=0')
```

### Argumente

'Expression'	Messwert, für den das Minimum gebildet wird	
'WindowInterval'	Angabe der Länge des Intervalls in Sekunden, über das das Minimum berechnet werden soll; muss ein Vielfaches von 'UpdateInterval' sein.	
'UpdateInterval'	Optionaler Parameter (Voreinstellung = Zeitbasis); gibt vor, in welchem Takt die Berechnung durchgeführt wird.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten, zurücksetzen und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten, zurücksetzen und Ergebnis beibehalten

### Beschreibung

Diese Funktion liefert das Minimum von 'Expression' alle 'UpdateInterval' Sekunden zurück, basierend auf einem gleitenden Fenster von 'WindowInterval' Sekunden.

Der Parameter 'UpdateInterval' ist optional. Wenn nicht angegeben, dann ist er gleich der Zeitbasis der Funktion (d. h. so klein wie möglich).

'WindowInterval' muss ein Vielfaches von 'UpdateInterval' sein. Wenn nicht, dann wird 'WindowInterval' automatisch geändert, zum ersten Vielfachen von 'UpdateInterval' größer als oder gleich 'WindowInterval'.

### 2.4.19 MSkewness

`MSkewness('Expression', 'WindowInterval', 'UpdateInterval=timebase', 'Reset=0')`

#### Argumente

'Expression'	Messwert, für den die Skewness gebildet wird	
'WindowInterval'	Angabe der Länge des Intervalls in Sekunden, über das die Skewness gebildet wird; muss ein Vielfaches von 'UpdateInterval' sein.	
'UpdateInterval'	Optionaler Parameter (Voreinstellung = Zeitbasis); gibt vor, in welchem Takt die Berechnung durchgeführt wird.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung.	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten und Ergebnis beibehalten

#### Beschreibung

Liefert den Skewness-Wert von 'Expression' alle 'UpdateInterval' Sekunden zurück, basierend auf einem gleitenden Fenster von 'WindowInterval' Sekunden.

Der Parameter 'UpdateInterval' ist optional. Wenn nicht angegeben, dann ist er gleich der Zeitbasis der Funktion (d. h. so klein wie möglich).

'WindowInterval' muss ein Vielfaches von 'UpdateInterval' sein. Wenn nicht, dann wird 'WindowInterval' automatisch geändert, zum ersten Vielfachen von 'UpdateInterval' größer als oder gleich 'WindowInterval'.

Für Informationen zur Skewness, siehe Kapitel [↗ SkewnessInTime](#), Seite 73

### 2.4.20 MStdDev

`MStdDev('Expression', 'WindowInterval', 'UpdateInterval=timebase', 'Reset=0')`

#### Argumente

'Expression'	Messwert, für den die Standardabweichung gebildet wird	
'WindowInterval'	Angabe der Intervalllänge in Sekunden, über das die Standardabweichung berechnet werden soll; muss ein Vielfaches von 'UpdateInterval' sein.	
'UpdateInterval'	Optionaler Parameter (Voreinstellung = Zeitbasis); gibt vor, in welchem Takt die Berechnung durchgeführt wird.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung:	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten, alle gepufferten Daten löschen und Ergebnis auf 0 setzen
	'Reset' = 2	Berechnung anhalten, alle gepufferten Daten löschen und Ergebnis beibehalten



### Beschreibung

Diese Funktion liefert als Ergebnis die Standardabweichung von 'Expression' alle 'UpdateInterval'-Sekunden, basierend auf einem gleitenden Fenster von 'WindowInterval'-Sekunden.

Der Parameter 'UpdateInterval' ist optional. Wenn nicht angegeben, dann ist er gleich der Zeitbasis der Funktion (d. h. so klein wie möglich).

Der Parameter 'WindowInterval' muss ein Vielfaches von 'UpdateInterval' sein. Wenn nicht, dann wird 'WindowInterval' automatisch geändert zum ersten Vielfachen von 'UpdateInterval' größer als oder gleich 'WindowInterval'.

'WindowInterval' und 'UpdateInterval' können durch Ausdrücke vorgegeben werden. Änderungen dieser Intervallwerte werden erst nach einem Reset übernommen.

### Tipp



Das Ergebnis der MStdDev-Funktion wird erst im nachfolgenden Intervall ausgegeben.

## 2.4.21 SkewnessInTime

`SkewnessInTime('Expression', 'Interval', 'Reset=0')`

### Argumente

'Expression'	Messwert, für den die Skewness gebildet wird	
'Interval'	Angabe der Länge des Intervalls in Sekunden, über das die Skewness berechnet werden soll.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset' = 2	Berechnung anhalten und Ergebnis beibehalten
	'Reset' = 3	Jetzt berechnen und danach Berechnung anhalten

### Beschreibung

Ähnlich wie der Kurtosis-Faktor so eignet sich der Skewness-Faktor zur Bewertung und Analyse von Schwingungen. Der Skewness-Faktor kann dann angewendet werden, wenn die Symmetrie-Eigenschaften eines Schwingungssignals überprüft werden sollen (z. B. Beschleunigungssignal).

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

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 Seite konzentriert ist. Eine linksschiefe (bzw. rechtssteile) Verteilung hat demnach den Hauptanteil auf der rechten Seite. Der Grad der Schiefe wird durch das dritte Moment der Verteilung bestimmt.

Für die Berechnung der Skewness wird ähnlich wie bei der Funktion KurtosisInTime verfahren.

## 2.4.22 StdDev

`StdDev('Expression', 'Reset=0')`

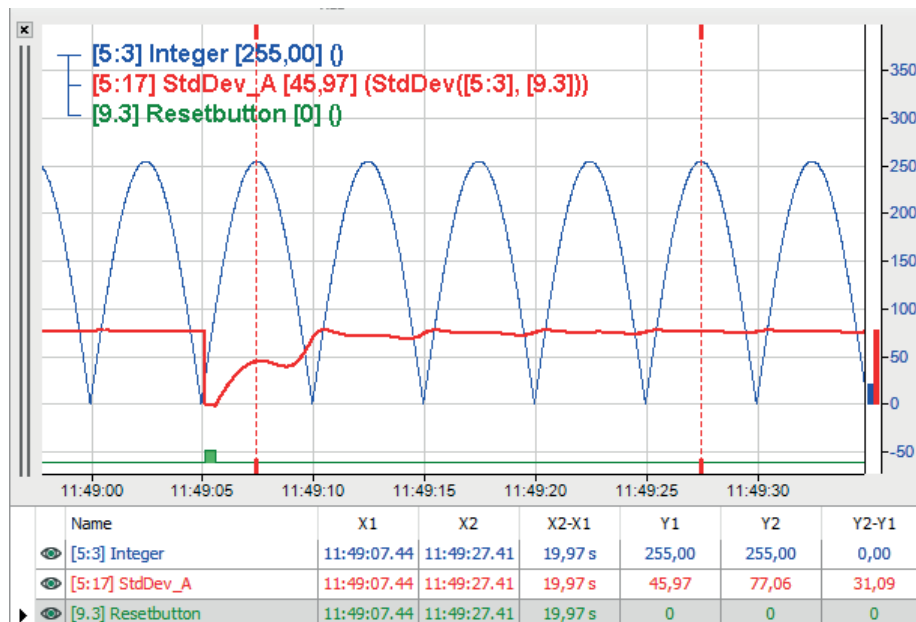
### Argumente

'Expression'	Messwert, für den die Standardabweichung berechnet werden soll	
'Reset'	Optionaler Parameter (Voreinstellung = 0) zum Stoppen und Neustarten der Berechnung	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten und Ergebnis auf 0 setzen

### Beschreibung

Diese Funktion liefert als Ergebnis die Standardabweichung von 'Expression' zurück. Wenn 'Reset' TRUE ist, dann wird die Berechnung zurückgesetzt und das Ergebnis ist 0.

### Beispiel



### 2.4.23 StdDev2

`StdDev2('Expression1', 'Expression2', ...)`

#### Argumente

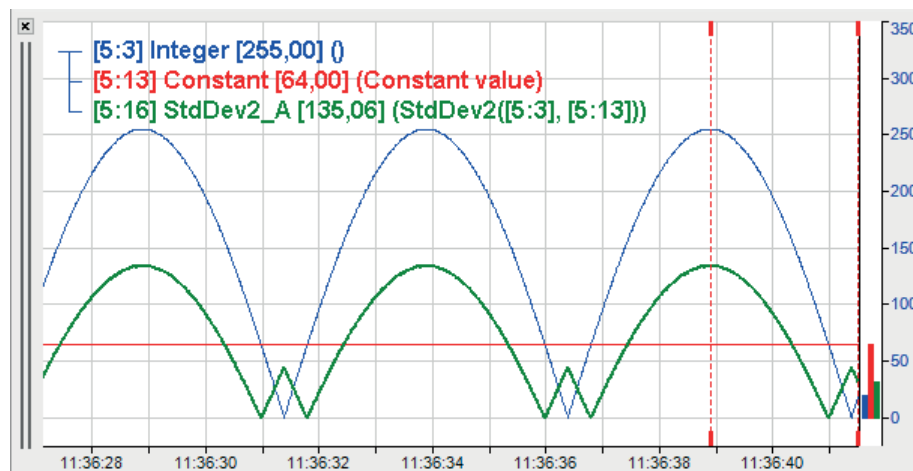
'Expression'	Messwert(e), für den/die die Standardabweichung gebildet wird
--------------	---

#### Beschreibung

Diese Funktion liefert als Ergebnis die Standardabweichung aller Argumente.

#### Beispiel

Standardabweichung von einer Konstanten und einem variablen Wert.



### 2.4.24 StddevInTime

`StddevInTime('Expression', 'Interval', 'Reset=0')`

#### Argumente

'Expression'	Messwert, für den die Standardabweichung gebildet wird	
'Interval'	Angabe der Länge des Intervalls in Sekunden, über das die Standardabweichung berechnet werden soll.	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten und Ergebnis auf 0 setzen
	'Reset' = 2	Berechnung anhalten und Ergebnis beibehalten
	'Reset' = 3	Jetzt berechnen und danach Berechnung anhalten

#### Beschreibung

Diese Funktion liefert als Ergebnis die Standardabweichung von 'Expression' über jedes Zeitintervall der Länge 'Interval'. Mit dem optionalen Parameter 'Reset' kann die Berechnung angehalten werden.

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

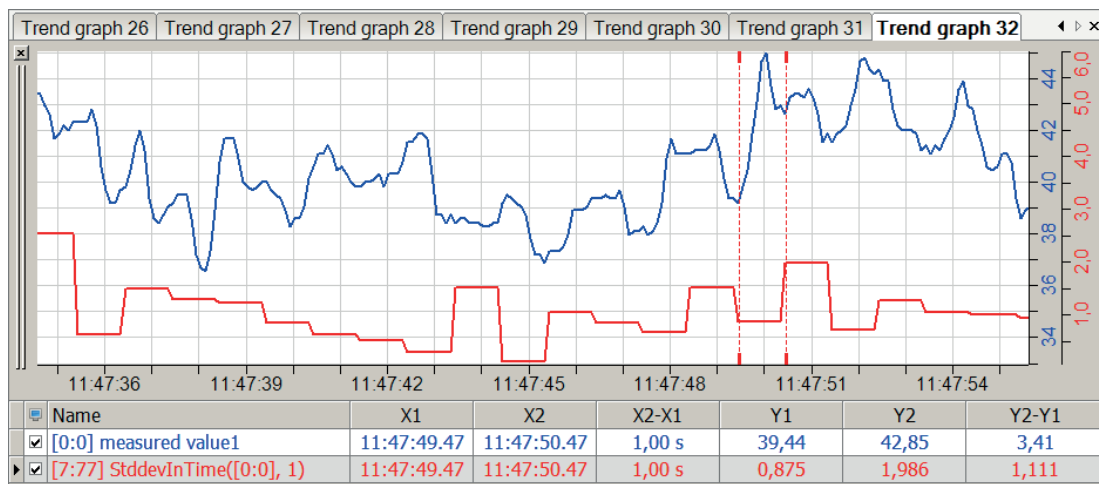
n = Anzahl Messungen

### Beispiel

Für einen Signalverlauf soll die Standardabweichung in Zeitintervallen von einer Sekunde bestimmt werden. Ein Reset ist nicht notwendig.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve die Messwerte und die rote Kurve zeigt die Standardabweichung in den Intervallen der Länge eine Sekunde.



### Hinweis



Die Standardabweichung wird immer für das vorhergehende Intervall angegeben.

## 2.5 Trigger-Funktionen

### 2.5.1 PeriodicTrigger

`PeriodicTrigger('Interval*', 'StartTime*', 'UseSystemTime*')`

#### Argumente

'Interval*'	Intervall in Minuten, in dem der Wert TRUE zurückgegeben wird	
'StartTime*'	Angabe in Minuten; ergibt nach Anwendung der Modulo-Funktion die Startzeit ('StartTime' Modulo 'Interval')	
'UseSystemTime*'	Legt fest, ob Systemzeit oder interner hochauflösender Timer verwendet wird.	
	'UseSystemTime' = 0	Keine Systemzeit, sondern interner Timer wird verwendet
	'UseSystemTime' > 0	Systemzeit wird verwendet.

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Diese Funktion liefert den Wert TRUE alle 'Interval' Minuten, beginnend bei 'StartTime' Modulo 'Interval'-Minuten. Der Merker 'UseSystemTime' legt fest, ob die Systemzeit oder der interne hochauflösende Timer zu verwenden ist.

#### Beispiel

Darstellung eines Triggers alle 0 Sekunden

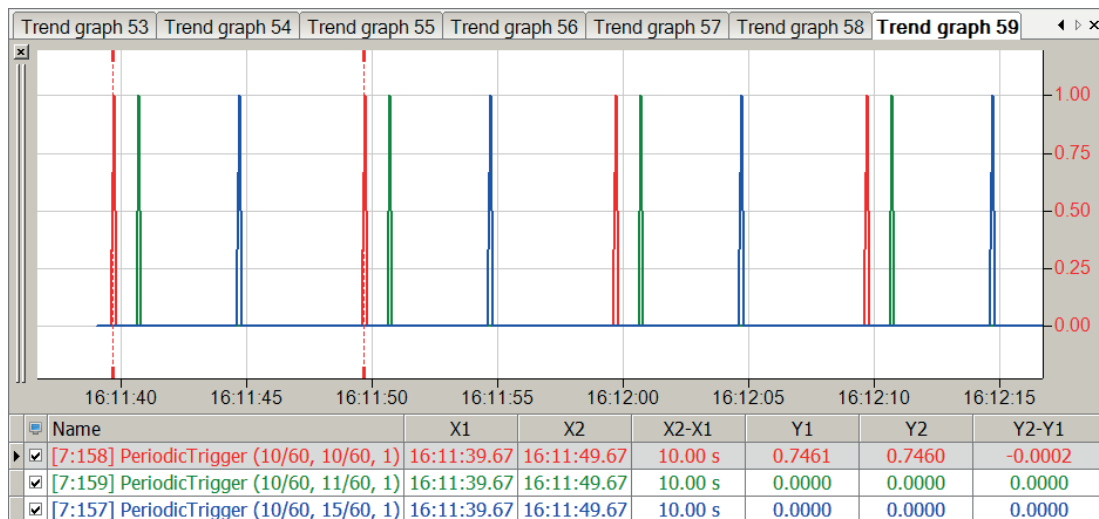
#### Lösung

Darstellung folgender Funktionen:

- `PeriodicTrigger(10/60, 10/60, 1)`
- `PeriodicTrigger(10/60, 11/60, 1)`
- `PeriodicTrigger(10/60, 15/60, 1)`

mit 10/60 min (entspricht 10 s) für 'Interval', wiederum 10/60 min bzw. 11/60 min (11 s) und 15/60 min (15 s) für 'StartTime' und 1 (Systemzeit) für 'UseSystemTime'.

Wie die folgende Abbildung zeigt, unterscheiden sich die Trigger alle 10 s nach ihrer Startzeit. Nach obigem Berechnungsschema startet das Trigger Intervall der roten Kurve bei 0 s, das der grünen Kurve bei 1 s und das der blauen Kurve bei 5 s nach dem Start.



### Hinweis



Negative Werte für 'StartTime' sind ungültig und führen zu einer Fehlermeldung.

## 2.5.2 TriggerChangeRate

`TriggerChangeRate('Expression', 'DeltaY*', 'DeltaT*', 'DeadTime*')`

### Argumente

'Expression'	Messwert
'DeltaY*'	Geforderter Wertabstand zum Auslösen des Triggers
'DeltaT*'	Betrachtetes Zeitintervall; Maximalwert = 1.000.000 × Modulzeitbasis
'DeadTime*'	Zeitangabe in Sekunden, die die Triggerbedingung bis zur Auslösung erfüllt sein muss. Ist keine Verzögerung erwünscht, so ist der Wert 0 einzutragen.

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Die Funktion liefert TRUE, solange die Änderung des Messwerts 'Expression' (dy) innerhalb des Intervalls 'DeltaT' größer als 'DeltaY' ist.

### Hinweis

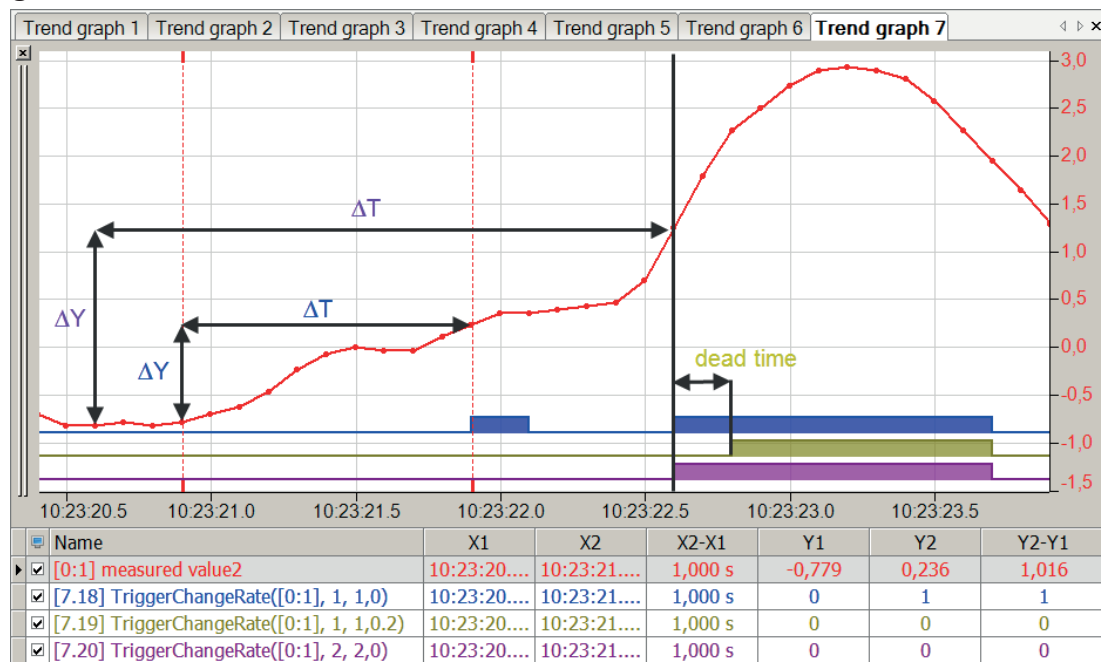


Die Funktion vergleicht den Abstand der Werte mit dem Zeitunterschied DeltaT mit DeltaY, die Steigung der Kurve kann zwischen den Werten auch abweichen. Der zu betrachtende Zeitunterschied DeltaT ist auf das 1-millionenfache der Modulzeitbasis begrenzt.

## Beispiel

Unterschiede in Triggerauslösung und Auswirkung der 'DeadTime'

## Lösung



Rot	Messwert	Blau	Triggerauslösung mit $\Delta Y=1$ und $\Delta T=1$
Gelb	Triggerauslösung mit $\Delta Y=1$ und $\Delta T=1$	Lila	Triggerauslösung mit $\Delta Y=2$ und $\Delta T=2$

## 2.5.3 TriggerConstant

`TriggerConstant('Expression', 'Level*', 'Epsilon*', 'DeadTime*')`

### Argumente

'Expression'	Messwert
'Level*'	Wertangabe der Mittellinie des Bereichs in dem der Trigger auslösen soll
'Epsilon*'	Wertangabe des Abstandes beider Bereichsgrenzen zur Mittellinie
'DeadTime*'	Zeitangabe in Sekunden, die die Triggerbedingung bis zur Auslösung erfüllt sein muss. Ist keine Verzögerung erwünscht, so ist der Wert 0 einzutragen.

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Die Funktion liefert TRUE, solange 'Expression' mindestens für die Dauer der 'DeadTime' innerhalb des Bereichs ['Level' - 'Epsilon', 'Level' + 'Epsilon'] bleibt.

### Hinweis



Im Gegensatz zu den Funktionen TriggerEdge und TriggerLevel wird das Triggersignal über die gesamte Dauer der Niveauüber- oder unterschreitung ausgegeben und nicht als Einzelimpuls.

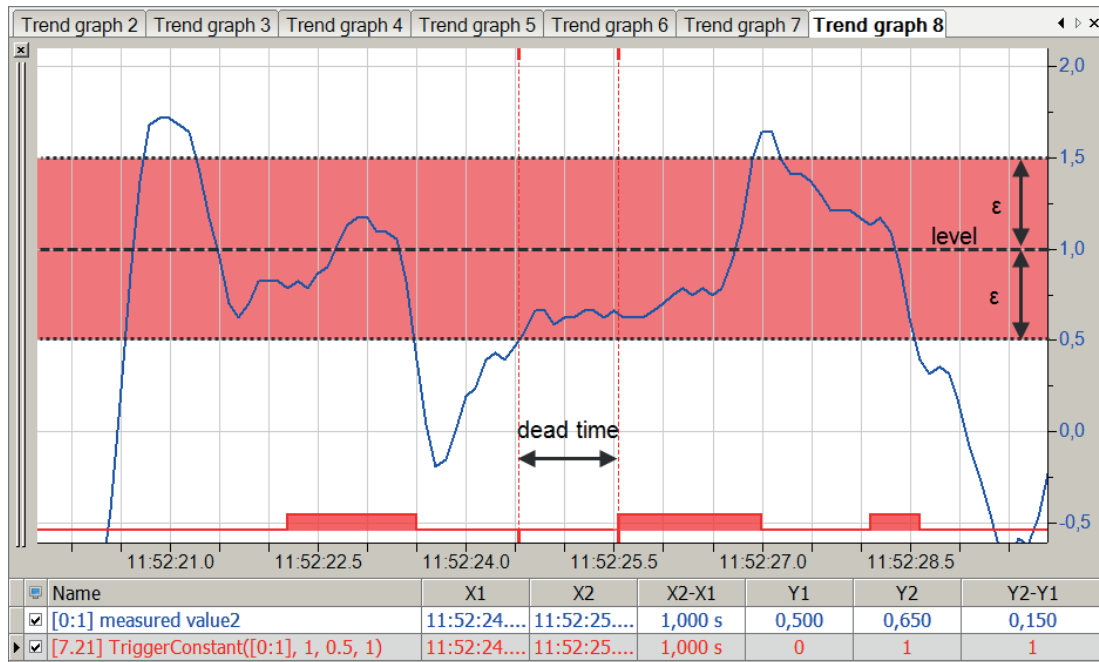


## Beispiel

Der Trigger soll ausgelöst werden, wenn sich der Messwert länger als eine Sekunde in einem Bereich zwischen 0,5 und 1,5 befindet.

## Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve das ursprüngliche Signal und der rote Balken zeigt den ausgelösten Trigger.



## 2.5.4 TriggerEdge

`TriggerEdge('Expression', 'Level*', 'EdgeType*', 'DeadTime*')`

### Argumente

'Expression'	Messwert	
'Level*'	Angabe des Niveauwertes	
'EdgeType*'	Angabe, ob steigende, fallende oder beide Flanken, d. h. Durchgänge des Niveauwertes gezählt werden	
	'EdgeType' < 0	nur fallende Flanken (Durchgang in negativer Richtung)
	'EdgeType' > 0	nur steigende Flanken (Durchgang in positiver Richtung)
	'EdgeType' = 0	fallende und steigende Flanken
'DeadTime*'	Zeitangabe in Sekunden, die der Messwert den Niveauwert über- oder unterschritten haben muss, um den Trigger auszulösen.	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Löst aus, wenn 'Expression' 'Level' über- oder unterschreitet und auf dieser 'Level'-Seite für mindestens 'DeadTime' Sekunden bleibt. Wenn 'Expression' ein Digitalsignal ist, dann ist 'Level' auf 0.5 fest eingestellt. 'EdgeType' bestimmt welche Flanken bzw. welcher Durchgang gezählt werden:

### Hinweis



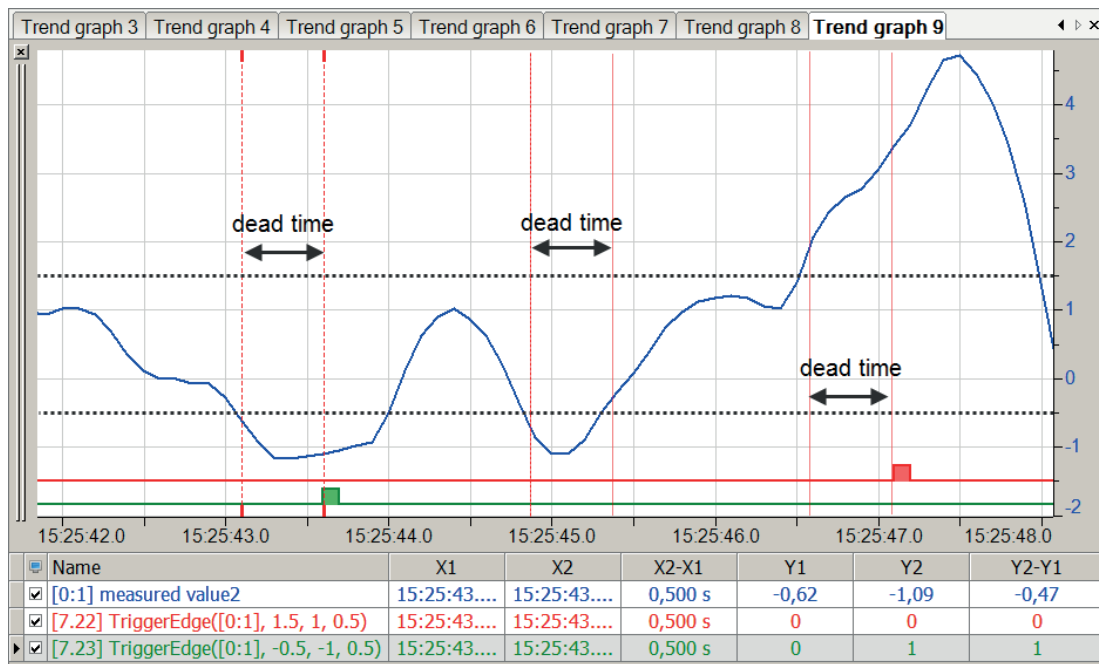
Im Gegensatz zu den Funktionen TriggerLevel und TriggerConstant wird beim Durchgang durch den Niveauwert nur ein Impuls ausgegeben.

### Beispiel

Es soll jeweils ein Trigger ausgelöst werden, wenn sich der Messwert für 0,5 Sekunden oberhalb von 1,5 oder unterhalb von -0,5 befindet.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert und der rote Balken zeigt den Trigger, wenn der Messwert länger als 0,5 Sekunden über 1,5 ist. Der grüne Balken zeigt den Trigger, wenn der Messwert länger als 0,5 Sekunden unter -0,5 ist.



## 2.5.5 TriggerLevel

TriggerLevel('Expression', 'Level\*', 'LevelType\*', 'DeadTime\*')

### Argumente

'Expression'	Messwert	
'Level*'	Angabe des Niveauwertes	
'LevelType*'	Angabe, welche Seite von 'Level' betrachtet wird	
	'LevelType' = 0	unterhalb Niveau
	'LevelType' = 1	oberhalb Niveau
'DeadTime*'	Zeitangabe in Sekunden, die der Messwert auf der betrachteten Seite des Niveauwertes verbleiben muss, um den Trigger auszulösen.	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Löst aus, wenn 'Expression' für mindestens 'DeadTime' Sekunden über bzw. unter 'Level' bleibt. 'LevelType' bestimmt, welche 'Level'-Seite überwacht wird.

### Hinweis



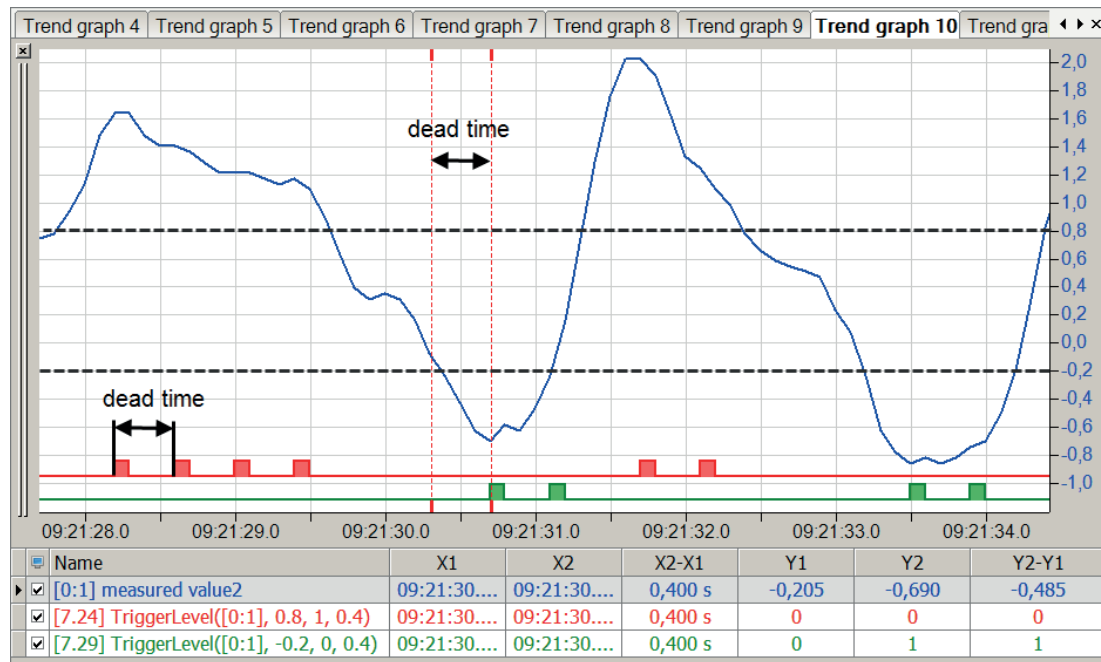
Im Gegensatz zu den Funktionen TriggerEdge und TriggerConstant werden über die Zeitdauer des Über- oder Unterschreitens periodisch Triggerimpulse in 'Dead-Time'-Intervallen ausgegeben.

### Beispiel

Es soll jeweils ein Trigger ausgelöst werden, wenn sich der Messwert für 0,4 Sekunden oberhalb von 0,8 und unterhalb von -0,2 befindet. Dabei soll ein Triggerimpuls für die gesamte Dauer der Über- oder Unterschreitung ausgegeben werden.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert und der rote Balken zeigt den Trigger alle 0,4 Sekunden, wenn der Messwert über 0,8 ist. Der grüne Balken zeigt den Trigger alle 0,4 Sekunden, wenn der Messwert unter -0,2 ist.



## 2.5.6 TriggerHarmonicLevel

TriggerHarmonicLevel('Expression', "'LimitProfile\*'", 'Harmonic\*')

### Argumente

'Expression'	Messsignal
""LimitProfile*""	Name des Grenzwertprofils, das in der PQU definiert ist
'Harmonic*'	Ordnung (Harmonische), deren Grenzwert überwacht werden soll, Werte 0 bis 50

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

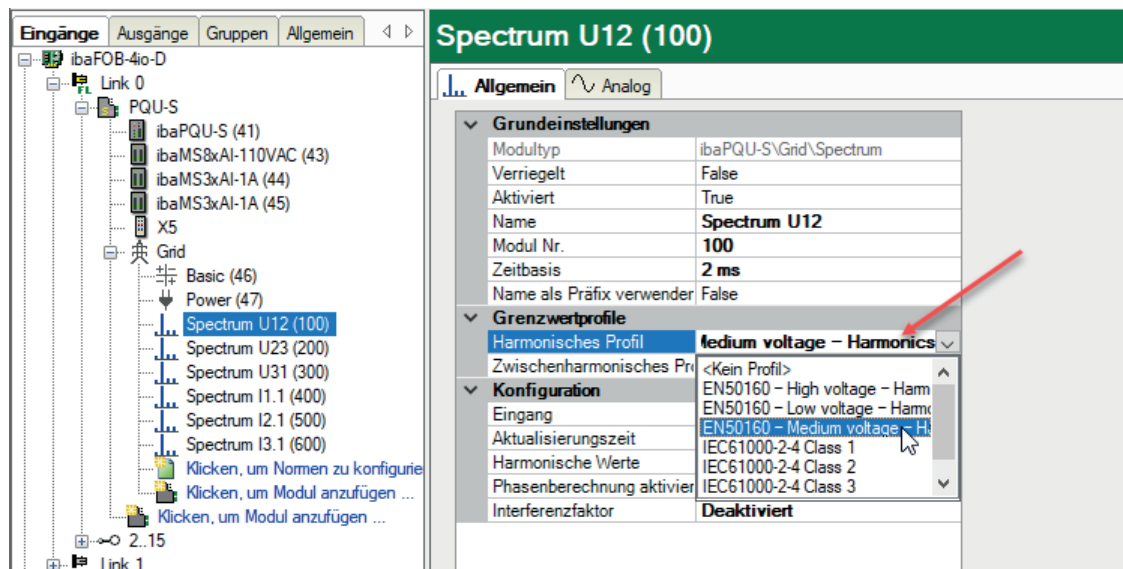
### Beschreibung

Trigger löst aus (Ergebnis True), wenn der Signalwert oberhalb der im Profil ""LimitProfile"" der PQU definierten harmonischen Grenze liegt. Der Parameter 'Harmonic' bestimmt, welcher Grenzwert aus dem Grenzwertprofil verwendet wird.

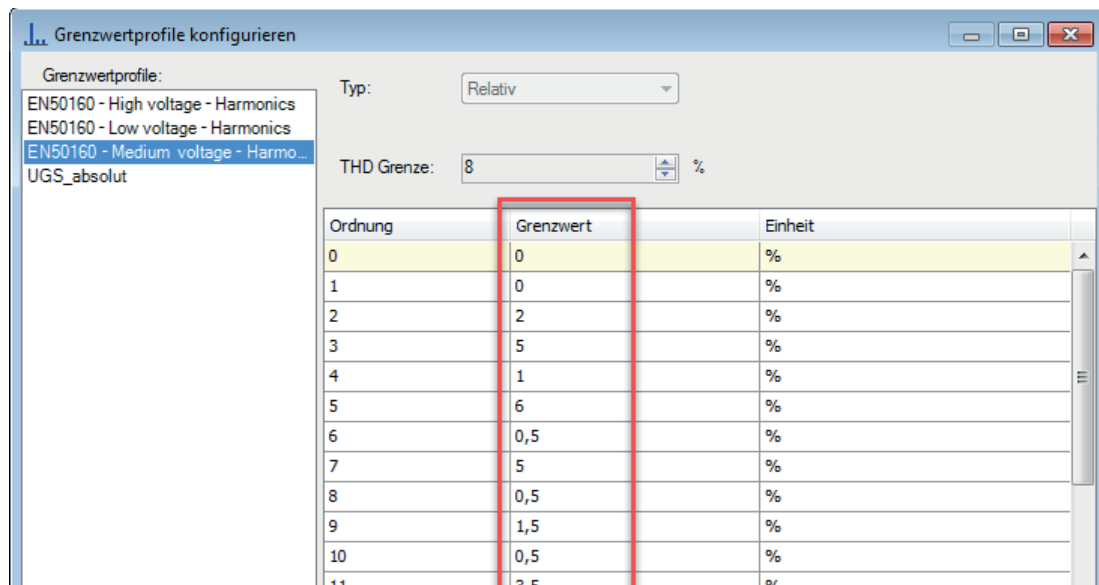
### Beispiel

Bei einer Messung an einer Mittelspannungsschaltanlage soll die Leiterspannung U12 auf ihren Oberschwingungsgehalt überwacht werden. Bei Überschreitung des nach EN50160 vorgeschriebenen Grenzwerts für die 3. Harmonische im 10 min-Intervall soll ein Triggersignal ausgelöst werden.

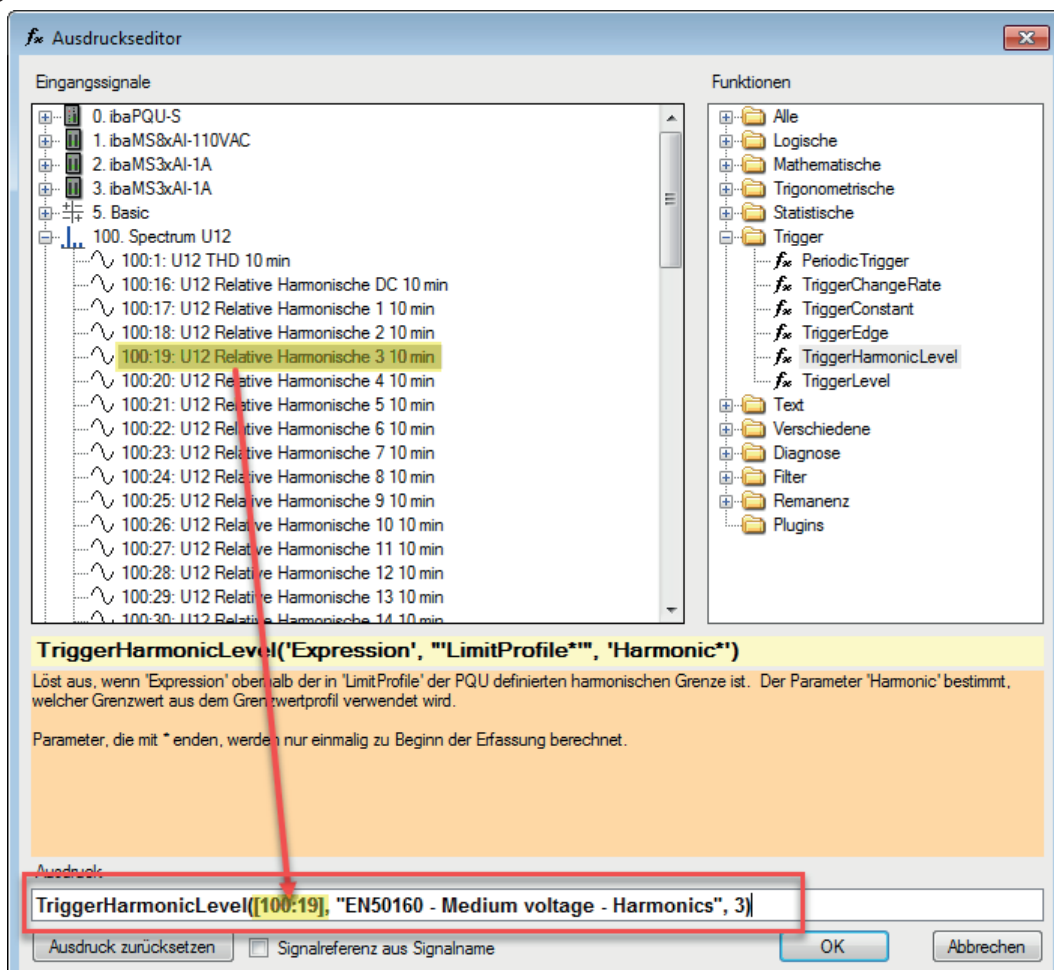
Das Grenzwertprofil ist im I/O-Manager in den Spectrum-Modulen konfiguriert. Der Name des Profils lautet "EN50160 – Medium voltage – Harmonics".



Die Grenzwerte können mit *Profile konfigurieren* eingesehen werden. In unserem Beispiel liegt der Grenzwert für die 3. Harmonische bei 5 %.



## Lösung



Der Trigger löst aus, wenn der 10 min-Wert der 3. Harmonischen von U12 über 5 % liegt.

## 2.6 Text-Funktionen

### 2.6.1 CharValue

```
CharValue('Text', 'CharNumber=0')
```

#### Argumente

'Text'	Textsignal	
'CharNumber'	Position des Zeichens im Text, Default = 0	

#### Beschreibung

Diese Funktion liefert den ASCII-Dezimalwert des Zeichens an der Position 'CharNumber' in einem Text zurück. Die Voreinstellung liefert das erste Zeichen (Position = 0).

Beispiel: Wenn 'Text' den Wert "A\_Text für Test" hat, dann ist das Ergebnis von CharValue('Text', 0) der ASCII-Wert 65 für den Großbuchstaben A.

#### Beispiel

Darstellung des in der Beschreibung genannten Beispiels

#### Lösung

Abfrage des ASCII-Dezimalwertes des ersten Zeichens im Text "A\_Text für Test". A hat in der ASCII Tabelle den Wert 65.



#### Tipp



Siehe ASCII Tabelle unter <http://www.ascii-code.com/>.

## 2.6.2 CompareText

```
CompareText('Text1', 'Text2', 'CaseSensitive=1')
```

Anmerkung:

Der ursprüngliche Name dieser Funktion war TextCompare. Sie wurde später in CompareText umbenannt. Aus Kompatibilitätsgründen ist die Funktion TextCompare weiter verwendbar, die Argumente sind identisch.

### Argumente

'Text1'	Angabe des ersten Vergleichstextes	
'Text2'	Angabe des zweiten Vergleichstextes	
'CaseSensitive*'	Optionaler Parameter (Voreinstellung =1) zur Festlegung, ob ein Case-sensitiver Vergleich (Berücksichtigung von Groß- und Kleinschreibung) gemacht werden soll	
	'CaseSensitive' = 1	Groß- und Kleinschreibung berücksichtigen
	'CaseSensitive' <> 1	Groß- und Kleinschreibung ignorieren

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion vergleicht 2 Texte alphabetisch miteinander. Das bedeutet, dass die Zeichen der 2 Strings einzeln miteinander verglichen werden, beginnend mit dem linken Zeichen. Die Funktion verwendet die gegenwärtige Kultureinstellung (Windows), um kulturspezifische Informationen zu Schreibweise und alphabetischer Reihenfolge zu erhalten. Der Vergleich berücksichtigt weder mehrstellige Zahlenwerte, noch Wortbedeutung oder die Länge des Strings. Ein Leerzeichen wird beim Vergleich berücksichtigt. Mit dem optionalen Parameter 'CaseSensitive=1' kann bestimmt werden ob Groß- und Kleinschreibung berücksichtigt werden soll (=1 oder keine Angabe), oder nicht (<>1).

Dynamischer Text kann mithilfe eines Textsignals verwendet werden. Ein Textsignal wird in rechteckige Klammern [ ] gesetzt. Es ist ebenso möglich einen konstanten Text einzugeben, indem der Text einfach in Anführungszeichen gesetzt wird.

Als Ergebnis gibt die Funktion einen Analogwert aus.

### Ergebnisse

-1	wenn die Zeichen des ersten Textes im Alphabet vor den Zeichen des zweiten Textes stehen
0	wenn die beiden Texte gleich sind
1	wenn die Zeichen des ersten Textes im Alphabet nach den Zeichen des zweiten Textes stehen



**Beispiel**

Die folgende Tabelle zeigt einige Beispiele:

Text1	Text2	Ergebnis		Anmerkung
		CompareText ("Text1","Text2",0)	CompareText ("Text1","- Text2",1)	
1234 abcd	1234 abcd	0	0	1 = 2
1234 abcd	1234 bcde	-1	-1	1 < 2 "a" steht vor "b"
1234 Abcd	1234 abcd	0	1	1 = 2 (Groß-/Kleinschreibung nicht berücksichtigt) 1 > 2 (Groß-/Kleinschreibung berücksichtigt) "A" steht nach "a"
12340 abcd	1234 _ abcd	1	1	1 > 2 "0" steht nach " _"
1234 0abcd	1234 abcd	-1	-1	1 < 2 "0" steht vor "a"
12034 abcd	1234 abcd	-1	-1	1 < 2 "0" steht vor "3"
1234 abcd	1y34 abcd	-1	-1	1 < 2 "2" steht vor "y"
1z34 abcd	1Y34 abcd	1	1	1 > 1 "z" steht nach "Y"

### 2.6.3 ConcatText

```
ConcatText('Text1', Text2', ...)
```

#### Argumente

'Textn'	Textsignal
---------	------------

#### Beschreibung

Diese Funktion liefert als Ergebnis eine Verkettung von 'Text1', 'Text2' usw.

Wenn Sie Anführungszeichen in statischen Texten verwenden wollen, schreiben Sie zwei Anführungszeichen hintereinander.

#### Beispiel

Die Werte dreier Textsignale sollen aneinander gereiht werden.

Textsignale sind [37:0], [37:1] und [37:2].

#### Lösung

```
ConcatText([37:0],[37:1],[37:2])
```

Text 1	Customer
Text 2	ACME
Text 3	4200
ConcatText	Customer ACME 4200

### 2.6.4 ConvertFromText

```
ConvertFromText('Expression', 'DecimalPoint*=0', 'Begin=0', 'End'=-1 (end of text)')
```

#### Argumente

'Expression'	Name des Textsignals	
'DecimalPoint*'	Dezimalzeichen	
	DecimalPoint = 0	Punkt
	DecimalPoint = 1	Komma
'Begin'	Index des ersten Zeichens des gewünschten Textes, Voreinstellung = 0	
'End'	Index des letzten Zeichens des gewünschten Textes, Voreinstellung = -1 (Ende des Textes)	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

## Beschreibung

Die Funktion parst eine Gleitkommazahl aus einem Text und liefert den numerischen Wert als Analogsignal. Steht an erster Stelle des durch die Argumente 'Beginn' und 'End' definierten Bereiches keine Zahl, wird 0 ausgegeben. Ausnahme: wenn nur Leerzeichen bis zum ersten numerischen Zeichen vorhanden sind. Ansonsten wird der Text bis zur ersten nichtnumerischen Stelle bzw. maximal bis 'End' gelesen. Führende Nullen vor einer Zahl dürfen nicht durch Leerzeichen oder nicht-numerische Zeichen unterbrochen werden.

## Beispiel

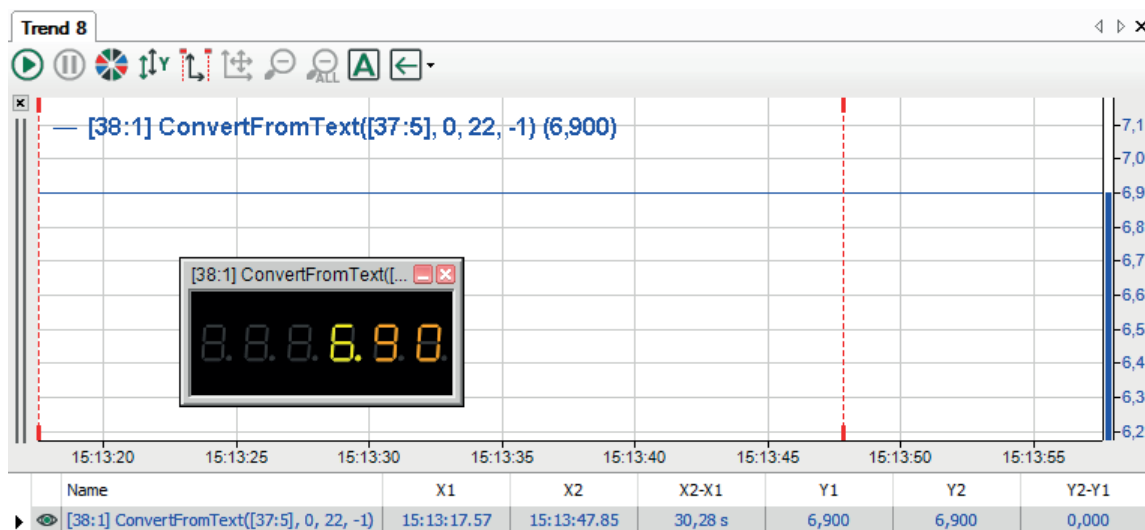
Einlesen eines definierten Texts

## Lösung

Die Funktion lautet: `ConvertFromText('Text', 0,22,-1)`

Der Inhalt des Textsignals lautet: `Voltage target value: 6.9 V`

Der Spannungswert beginnt bei der Position 22. Als Ende-Index wird -1 verwendet, damit auch Werte mit mehr Vor- und/oder Nachkommstellen erfasst werden.



## 2.6.5 ConvertToText

```
ConvertToText('Expression','IntegerDigits*=1','FractionalDigits*='',PlusSign*=2,'DecimalPoint*=0')
```

### Argumente

'Expression'	Ausdruck (Gleitkommawert), der in Text gewandelt werden soll	
'IntegerDigits*'	Mindestanzahl der Vorkommastellen	
'FractionalDigits*'	Anzahl der Nachkommastellen	
'PlusSign*'	Darstellung positiver Werte (Pluszeichen)	
	'PlusSign' = 0	Leerzeichen
	'PlusSign' = 1	"+"
	'PlusSign' = 2	Nichts
'DecimalPoint*'	Dezimaltrennzeichen	
	'DecimalPoint*' = 0	Punkt
	'DecimalPoint*' = 1	Komma

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt die Textdarstellung eines Gleitkommawerts zurück. Die Mindestanzahl der Vorkommastellen kann mit dem Argument 'IntegerDigits' definiert werden. Mit dem Argument 'FractionalDigits' legen Sie die Anzahl der Nachkommastellen fest. Wenn 'FractionalDigits' kleiner als null ist, werden nur Stellen ausgegeben, die ungleich null sind. Wenn 'FractionalDigits' größer null ist, werden auch Stellen ausgegeben, die gleich null sind. Der Parameter 'PlusSign' bestimmt die Ausgabe des positiven Vorzeichens.

### Beispiele

Beispiele für den Gleitkommawert [FloatValue] = 42,471130

'Expression'	'Expression'	'IntegerDigits'	'FractionalDigits'	'PlusSign'	'DecimalPoint'
	[FloatValue] Wert: 42,471130	1	4	2	1
Formel	<code>ConvertToText([FloatValue],1,4,2,1)</code>				
Ergebnis	42,4711				

'Expression'	'Expression'	'IntegerDigits'	'Fractional-Digits'	'PlusSign'	'DecimalPoint'
	[FloatValue] Wert: 42,471130	1	6	1	1
Formel	ConvertToText ([FloatValue], 1, 6, 1, 1)				
Ergebnis	+42,471130				

'Expression'	'Expression'	'IntegerDigits'	'Fractional-Digits'	'PlusSign'	'DecimalPoint'
	[FloatValue] Wert: 42,471130	1	-6	0	1
Formel	ConvertToText ([FloatValue], 1, -6, 0, 1)				
Ergebnis	42,47113				

## 2.6.6 CountText

CountText('Text', 'CountOnlyDifferent\*=0', 'Reset=0')

### Argumente

'Text' Name des Textsignals

'CountOnlyDifferent\*' Optionaler Parameter (Voreinstellung = 0)

'CountOnlyDifferent' <> 0 Neuer Text wird nur gezählt, wenn er sich vom vorigen unterscheidet.

'CountOnlyDifferent' = 0 Jeder neue Text wird gezählt.

'Reset' Optionaler Parameter, der zum Rücksetzen des Zählerstands verwendet werden kann. 'Reset' kann selbst auch ein Ausdruck sein.

'Reset' > 0 Zähler wird zurückgesetzt

'Reset' = 0 Zählung freigegeben (Voreinstellung)

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion zählt den Empfang bzw. die Änderung eines Textsignals und liefert den Zählerstand zurück.

## 2.6.7 DeleteText

```
DeleteText('Text','StartPos','Lenght=100000')
```

### Argumente

'Text'	Textsignal oder "Text", aus dessen Text Zeichen gelöscht werden sollen
'StartPos'	Analogsignal oder Zahlenwert; gibt die Startposition im Text an, ab der gelöscht werden soll. Zählung beginnt bei null (0 = 1. Zeichen).
'Length'	Analogsignal oder Zahlenwert; gibt die Anzahl der zu löschenden Zeichen an, Default=100000

### Beschreibung

Diese Funktion entfernt aus einem Text ab der Position 'StartPos' eine Anzahl 'Length' Zeichen. Wird 'Length' nicht angegeben, dann werden alle Zeichen von 'StartPos' bis Ende entfernt.

### Beispiel

Aus einem Text sollen ab Position 4 vier Zeichen entfernt werden.

Argumente	'Text'	'StartPos'	'Length'
	[TextSignal] Wert: ABCDE1234XYZ	4	4
Formel	<code>DeleteText([TextSignal],4,4)</code>		
Ergebnis	ABCD4XYZ		

## 2.6.8 FindText

```
FindText("'Text1'", "'Text2'", ' CaseSensitive=1*')
```

### Argumente

'Text1'	Angabe des ersten Vergleichstextes	
'Text2'	Angabe des zweiten Vergleichstextes	
'CaseSensitive*'	Optionaler Parameter (Voreinstellung =1) zur Festlegung, ob ein Case-sensitiver Vergleich (Berücksichtigung von Groß- und Kleinschreibung) gemacht werden soll	
	'CaseSensitive' <> 0	Groß- und Kleinschreibung berücksichtigen
	'CaseSensitive' = 0	Groß- und Kleinschreibung ignorieren

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

## Beschreibung

Diese Funktion prüft, ob 'Text2' in 'Text1' enthalten ist und gibt als Ergebnis den Index (Stelle in 'Text1, erstes Zeichen =Index 0) zurück, wo 'Text2' zum ersten Mal gefunden wurde. Leerzeichen werden bei der Suche berücksichtigt. Mit dem optionalen Parameter 'CaseSensitiv=1' kann bestimmt werden ob Groß- und Kleinschreibung berücksichtigt werden soll ( =1 oder keine Angabe), oder nicht (<>1).

Dynamischer Text kann mithilfe eines Textsignals verwendet werden. Ein Textsignal wird in rechteckige Klammern [] gesetzt. Es ist ebenso möglich einen konstanten Text einzugeben, indem der Text einfach in Anführungszeichen gesetzt wird.

Als Ergebnis gibt die Funktion einen Analogwert aus.

## Ergebnisse

-1	wenn der zweite Text nicht im ersten Text enthalten ist
0....n	Position, wo 'Text2' das erste Mal in 'Text1' steht

## Beispiel

Die folgende Tabelle zeigt einige Beispiele:

Text1	Text2	Ergebnis		Anmerkung
		FindText ("Text1","Text2",0)	FindText ("Text1","-Text2",1)	
Die Sonne scheint	Sonne	4	4	
Der Mond scheint	Sonne	-1	-1	
Die sonne scheint	Sonne	4	-1	

## 2.6.9 InsertText

```
InsertText('Text1', 'Text2', 'Pos')
```

### Argumente

'Text1'	Textsignal oder "Text", in dessen Text ein anderer Text eingefügt werden soll	
'Text2'	Textsignal oder "Text", dessen Text in 'Text1' eingefügt werden soll	
'Pos'	Analogsignal oder Zahlenwert; gibt die Position an, an der der Text eingefügt werden soll.	
	'Pos' $\leq 0$	'Text2' wird 'Text1' vorangestellt.
	'Pos' $\geq$ Länge von 'Text1'	'Text2' wird an 'Text1' hinten angehängt.

### Beschreibung

Diese Funktion fügt in einen 'Text1' an der Position 'Pos' einen 'Text2' ein.

### Beispiel

Argumente	'Text1'	'Text2'	'Pos'
	[TextSignal_1] Wert: 123_XYZ	[TextSignal2] Wert: abcd_	4
Formel	<code>InsertText([TextSignal1],[TextSignal2],4)</code>		
Ergebnis	123_abcd_XYZ		



## 2.6.10 MidText

```
MidText('Text','StartPos','Lenght=100000')
```

### Argumente

'Text'	Textsignal oder "Text", aus dessen Text Zeichen zurückgegeben werden sollen
'StartPos'	Analogsignal oder Zahlenwert; gibt die Startposition im Text an, ab der die Zeichen zurückgegeben werden sollen. Zählung beginnt bei null (0 = 1. Zeichen).
'Length'	Analogsignal oder Zahlenwert; gibt die Anzahl der zurückzugebenden Zeichen an, Default=100000

### Beschreibung

Diese Funktion gibt aus einem Text ab der Position 'StartPos' eine Anzahl 'Length' Zeichen zurück. Wird 'Length' nicht angegeben, dann werden alle Zeichen von 'StartPos' bis Ende zurückgegeben.

### Beispiel

In einer standardisierten Auftragsnummer (Text) steht an der 6. Stelle immer eine vierstellige Kundenkennung. Diese Kundenkennung soll ausgelesen werden.

Argumente	'Text'	'StartPos'	'Length'
	[TextSignal_Order] Wert: 4711_AC-ME_1234XYZ	5	4
Formel	<code>MidText([TextSignal_Order],5,4)</code>		
Ergebnis	ACME		

## 2.6.11 ReplaceText

```
ReplaceText('Text', 'SearchText', 'ReplaceText')
```

### Argumente

'Text'	Textsignal oder "Text" mit zu ersetzendem Inhalt
'SearchText'	Textsignal mit dem zu ersetzenden Text oder "Text"
'ReplaceText'	Textsignal mit dem ersetzenden Text oder "Text"

### Beschreibung

Diese Funktion ersetzt jedes Vorkommen von 'SearchText' innerhalb von 'Text' durch 'ReplaceText'. Alle Argumente können sowohl Textsignale oder Statischer Text sein. Wenn Sie statischen Text direkt in die Formel eingeben, setzen Sie ihn in Anführungszeichen. Mit dem Einsatz von Textsignalen können Sie sowohl den zu ersetzenden Text ('SearchText') als auch den ersetzenden Text ('ReplaceText') dynamisch gestalten. Auch der Basistext, der zu ersetzende Inhalte enthält ('Text') kann als Textsignal dynamisch sein.

Falls Sie Anführungszeichen in statischen Texten verwenden wollen, schreiben Sie zwei Anführungszeichen hintereinander.

### Beispiele

Die nachfolgende Tabellen zeigen vereinfacht verschiedene Anwendungen.

Argumente	'Text'	'SearchText'	'ReplaceText'
	[TextSignal] Wert: AAA BBB CCC	"BBB"	[ReplaceTextSignal] Wert: XXX
Formel	ReplaceText([TextSignal], "BBB", [ReplaceTextSignal])		
Ergebnis	AAA XXX CCC		

Argumente	'Text'	'SearchText'	'ReplaceText'
	[TextSignal] Wert: AAA BBB CCC	[SearchTextSignal] Wert: BB CC	"XXX"
Formel	ReplaceText([TextSignal], [SearchTextSignal], "XXX")		
Ergebnis	AAA BXXXC		

Argumente	'Text'	'SearchText'	'ReplaceText'
	"Hello Name"	"Name"	[ReplaceTextSignal] Wert: John
Formel	<code>ReplaceText("Hello Name", "Name", [ReplaceTextSignal])</code>		
Ergebnis	Hello John		

## 2.6.12 TextLength

`TextLength('Text')`

### Argumente

'Text'	Textsignal oder "Text", dessen Textlänge in Anzahl Zeichen zurückgegeben werden soll.
--------	---

### Beschreibung

Diese Funktion gibt die Anzahl der Zeichen eines Texts zurück.

### Beispiel

Argumente	'Text'
	"Wie viele Zeichen hat dieser Text?"
Formel	<code>TextLength("Wie viele Zeichen hat dieser Text?")</code>
Ergebnis	34

## 2.6.13 TrimText

```
TrimText('Text', 'TrimOption=0')
```

### Argumente

'Text'	Textsignal, das verarbeitet werden soll	
'TrimOption'	Optionale Angabe , welche Leerzeichen aus dem Text entfernt werden sollen	
	'TrimOption' = 0	Default; es werden sowohl führende Leerzeichen als auch Leerzeichen am Ende des Textes entfernt.
	'TrimOption' = 1	Es werden nur führende Leerzeichen entfernt.
	'TrimOption' = 2	Es werden nur Leerzeichen am Ende entfernt.
	'TrimOption' = 3	Es werden alle Leerzeichen entfernt, inkl. Leerzeichen im Innern des Textes

### Beschreibung

Diese Funktion entfernt aus einem Text die Leerzeichen. Welche Leerzeichen entfernt werden sollen, steuern Sie über die Option 'TrimOption'.

### Beispiel

Folgender Text aus dem Textsignal 'TextToTrim' soll in verschiedenen Stufen von den Leerzeichen befreit werden. Die senkrechten Zeichen markieren Anfang und Ende des Textes inkl. Leerzeichen, sind selbst aber nicht Bestandteil des Textes.

| Lorem ipsum dolor sit amet |

### Lösung

<code>TrimText('TextToTrim')</code>	Lorem ipsum dolor sit amet
<code>TrimText('TextToTrim',1)</code>	Lorem ipsum dolor sit amet
<code>TrimText('TextToTrim',2)</code>	Lorem ipsum dolor sit amet
<code>TrimText('TextToTrim',3)</code>	Loremipsumdolorsitamet

## 2.7 Verschiedene Funktionen

### 2.7.1 ClientInfo

```
ClientInfo('"ClientAddress*"', 'InfoType*')
```

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Diese Funktion gibt Informationen über den mit 'ClientAddress' spezifizierten Client zurück. Der Parameter 'ClientAddress' kann die IP-Adresse des Rechners sein oder der Rechnername (Groß-Kleinschreibung nicht relevant). Diese Informationen können genutzt werden zur Steuerung anderer Funktionen oder zu Anzeige- und Diagnosezwecken.

Geben Sie den Informationstyp ('InfoType') an, den Sie erhalten wollen.

Folgende Informationstypen werden unterstützt:

Informationstypen	Mögliche Ergebnisse
0	0 = Keine Verbindung zum Server 1 = Client ist mit Server verbunden
1	Anzahl der vom Client angeforderten Signale
2	0 = Client belegt keine Client-Lizenz 1 = Client belegt eine Client-Lizenz
3	0 = Client belegt keine QPanel-Lizenz 1 = Client belegt eine QPanel-Lizenz

Tab. 2: Informationstypen und mögliche Ergebnisse der Funktion ClientInfo

## 2.7.2 ClientInfoText

```
ClientInfoText('ClientAddress*', 'InfoType*')
```

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt textuelle Informationen über den mit 'ClientAddress' spezifizierten Client zurück. Der Parameter 'ClientAddress' kann die IP-Adresse des Rechners sein oder der Rechnername (Groß-Kleinschreibung nicht relevant). Diese Informationen können genutzt werden zur Steuerung anderer Funktionen oder zu Anzeige- und Diagnosezwecken.

Geben Sie den Informationstyp ('InfoType') an, den Sie erhalten wollen.

Folgende Informationstypen werden unterstützt:

Informationstypen	Mögliche Ergebnisse
0	Rechnername des Clients
1	IP-Adresse des Client-Rechners
2	Windows-Benutzername des Clients
3	Name des aktuell angemeldeten ibaPDA-Benutzers
4	Zeitpunkt des Verbindungsaufbaus mit dem Client
5	Version der ibaPDA-Client software

Tab. 3: Informationstypen und mögliche Ergebnisse der Funktion ClientInfoText

## 2.7.3 Count

```
Count('Expression', 'Level*', 'Hysteresis*', 'EdgeType*', 'Reset=0')
```

### Argumente

'Expression'	Messwert	
'Level*'	Angabe des Niveauwertes	
'Hysteresis*'	Angabe eines Hysteresebandes	
'EdgeType*'	Angabe, ob steigende, fallende oder steigende und fallende Flanken gezählt werden sollen	
	'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'	Optional digitaler Parameter, der zum Rücksetzen des Zählers verwendet werden kann. 'Reset' kann selbst auch ein Ausdruck sein.	
	'Reset' > 0	Zähler wird zurückgesetzt
	'Reset' = 0	Zählerwert bleibt erhalten/zählt weiter (Voreinstellung)

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

**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 'Hysterese' kann ein Toleranzband angegeben werden, was zu gleichen Teilen ober- und unterhalb von 'Level' liegt. Es werden nur komplette Durchgänge durch das Toleranzband gezählt.

Der Parameter 'EdgeType' bestimmt, welche Flanken gezählt werden.

Der Parameter 'Reset' dient zum Rücksetzen des Zählerwertes auf 0. 'Reset' kann auch als Ausdruck formuliert werden.

Beispiele:

Count([0:0],10,1,1)	Es erfolgt kein Reset ('Reset' weggelassen)
Count([0:0],10,1,1,If(Mod(T(),20)=0,TRUE(),FALSE()))	Der Zähler wird in einem Zeitintervall von 20 s zurückgesetzt.
Count([0:0], [3.1])	z. B. mit [3.1] = If([0:0]<1, 1, 0)  Der Counter wird zurückgesetzt, sobald der Ausdruck [3.1] TRUE zurückgibt, also wenn der Ausdruck [0:0] den Grenzwert 1 unterschreitet

**Tipp**

Die COUNT-Funktion kann auch für binäre Signale verwendet werden. Dazu als Pegel 0.5 und als Hysterese z. B. 0.1 eingeben. Damit werden alle Wechsel von FALSE nach TRUE und umgekehrt erfasst und gezählt.

**Beispiel****Tipp**

Diese Funktion kann in einem virtuellen Remanenzmodul verwendet werden. Ihre Ergebniswerte können damit über ein Stoppen und Neustarten der Messung erhalten werden.

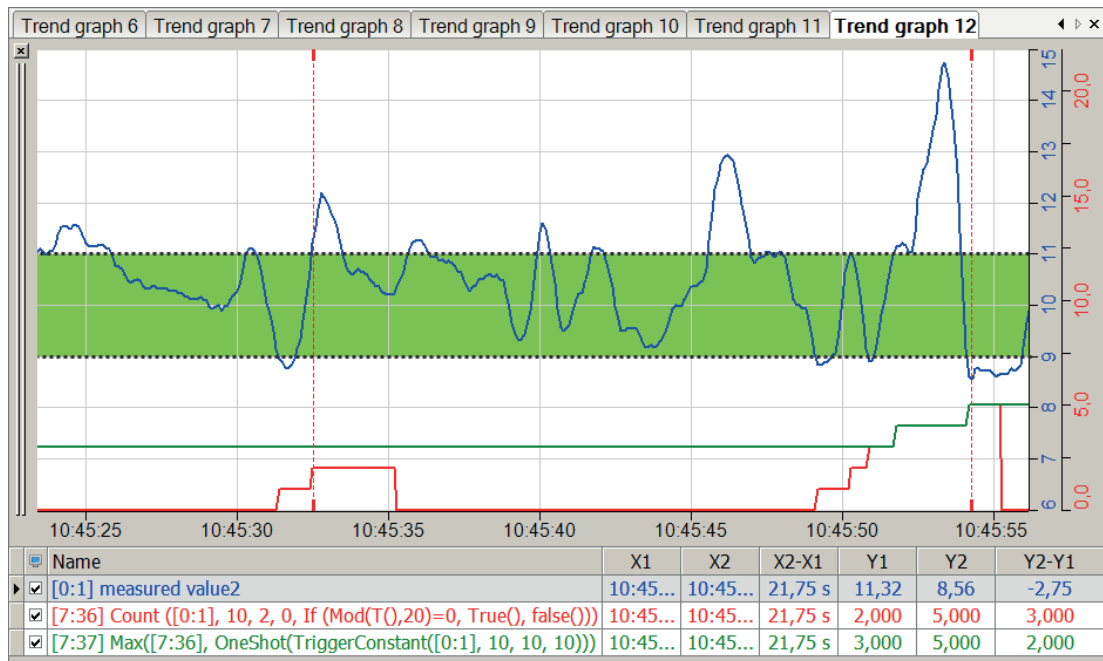
Es soll die Anzahl der Abweichungen des Messsignals vom Wert 10 in einem 20-Sekunden Intervall in beide Richtungen gezählt werden, allerdings nur nach kompletten Durchgängen durch das Hystereseband der Breite 2. Dabei soll der Maximalwert der einzelnen Intervalle ohne die Startphase gespeichert werden.



## Aufgabenstellung

Der 'Reset'-Parameter der Count-Funktion besteht aus einer If-Abfrage der Modulo-Funktion der Zeit. Der Maximalwert wird einmalig über die OneShot-Funktion zurückgesetzt, sobald sich der Messwert zu Beginn der Messung eingependelt hat.

## Lösung



Blau	Messwert	Grüner Bereich	Hystereseband
Rot	Count-Funktion	Grünes Signal	Maximalwert der Intervalle der Count-Funktion

## Tipp



Bei Angabe eines Hysteresebandes von 2 um das 'Level' 10 werden Durchgänge in steigender Richtung erst bei 'Expression' > 11 und in fallender Richtung bei 'Expression' < 9 gezählt.

## 2.7.4 CountUpDown

```
CountUpDown('Up', 'Down', 'Reset', 'UpperLimit=none', 'LowerLimit=none', 'ResetOnLimit=0')
```

### Argumente

'Up'	Digitales Signal, mit dessen steigender Flanke (FALSE --> TRUE) der Zählerwert um 1 erhöht wird	
'Down'	Digitales Signal, mit dessen steigender Flanke (FALSE --> TRUE) der Zählerwert um 1 erniedrigt wird	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen, Rücksetzen und Neustarten der Berechnung	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten und Ergebnis auf 0 setzen
'UpperLimit'	Optionaler Parameter (Voreinstellung =0); oberer Grenzwert für den Zähler	
'LowerLimit'	Optionaler Parameter (Voreinstellung =0) ; untere Grenzwert für den Zähler	
'ResetOnLimit'	Optionaler Parameter (Voreinstellung =0) zum Rücksetzen des Zählerwertes auf 0 bei Erreichen eines Grenzwertes	
	'ResetOnLimit' = 0	Zählerwert bleibt bei Erreichen eines Grenzwertes auf dem Grenzwert stehen.
	'ResetOnLimit' = 1	Zählerwert wird bei Erreichen eines Grenzwertes auf 0 gesetzt

### Beschreibung

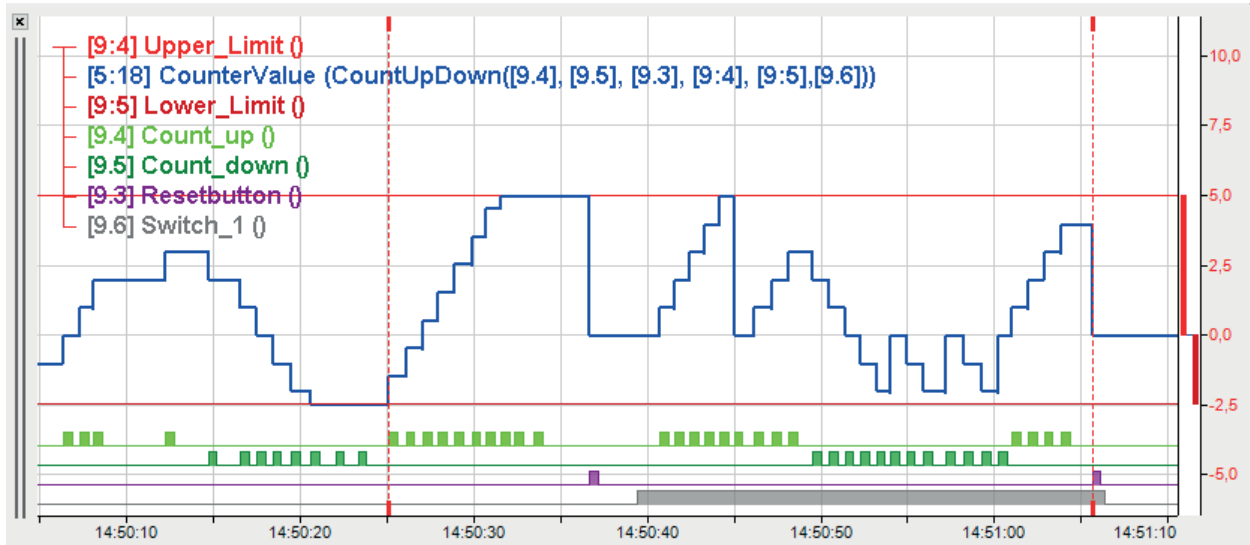
Diese Funktion erzeugt einen Zählerwert, der mit jeder steigenden Flanke von 'up' um 1 erhöht und mit jeder steigenden Flanke von 'Down' um 1 erniedrigt wird. Die Zählung kann unbegrenzt erfolgen und mit 'Reset' = TRUE bzw. 1 auf null zurückgesetzt werden.

'UpperLimit' (oberer Grenzwert) und 'LowerLimit' (untere Grenzwert) sind optional und können als fester Wert oder durch einen Ausdruck vorgegeben werden. Wenn die Grenzwertargumente vorhanden sind und der Zählerstand einen Grenzwert erreicht, zählt er nicht weiter sondern bleibt auf dem Grenzwert stehen, bis entweder ein Zählimpuls in die Gegenrichtung erfolgt oder 'Reset' = TRUE gesetzt wird.

Wenn der optionale Parameter 'ResetOnLimit' = TRUE bzw. 1 ist, dann wird der Zählerstand automatisch auf null gesetzt, wenn er einen Grenzwert erreicht oder überschreiten würde.

Das Rücksetzen auf null funktioniert nur, wenn der untere Grenzwert <0 bzw. der obere Grenzwert >0 ist.

## Beispiel



## 2.7.5 Delay

`Delay('Expression', 'NumberSamples*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt eine verzögerte Kopie des Signals 'Expression' wieder. Die Verzögerung wird angegeben in Anzahl Messungen ('NumberSamples'). Das Ergebnis ist eine Signalkurve mit den Werten des Originalsignals bei 'NumberSamples' vor der aktuellen Zeit.

Um eine Speicherüberlastung zu vermeiden, ist 'NumberSamples' auf 10.000 begrenzt.

### Tipp



Bei verschiedenen Zeitbasen von Messwert und Funktion ist die Zeitbasis der Funktion relevant für die Anzahl der Messungen.

### Beispiel

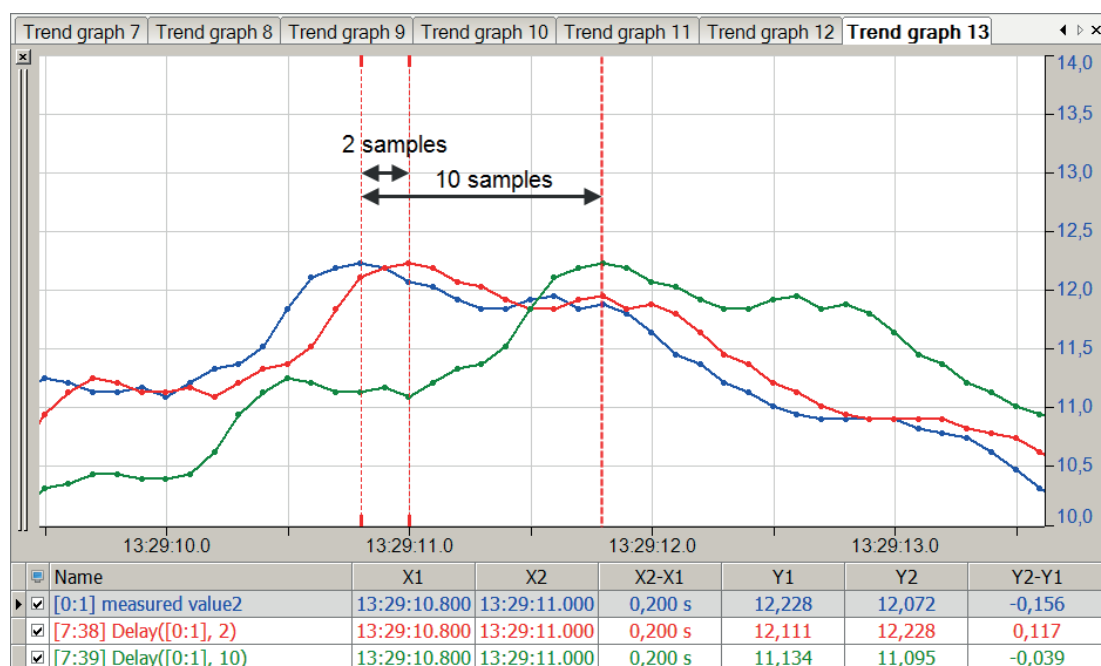
Ein Messwert soll verzögert um 2 Messwerte und um 10 Messwerte ausgegeben werden.

### Aufgabenstellung

Bei der gewählten Zeitbasis von 100 ms entspricht eine Verzögerung um zwei Messwerte einer Zeit von 0,2 Sekunden und eine Verzögerung um 10 Messwerte 1 Sekunde.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve das ursprüngliche Signal und die rote Kurve zeigt das um 2 Messwerte (=0,2 Sekunden) verzögerte Signal. Die grüne Kurve zeigt das um 10 Messwerte (=1 Sekunde) verzögerte Signal.



## 2.7.6 DelayLengthL

```
DelayLengthL('Expression', 'Length', 'MaxLengthDelta', 'DelayInMeter', 'Resolution*',  
'Filter=0*')
```

### Argumente

'Expression'	Eingangssignal	
'Length'	Längensignal (in m)	
'MaxLengthDelta'	Obergrenze zur Berücksichtigung der Änderungen des Längensignals	
'DelayInMeter'	Verzögerung (in m)	
'Resolution*'	Auflösung: Längenbasis des Ergebnisses (in m)	
'Filter*'	Optional Parameter (Voreinstellung =0) zur Einstellung des Filters bei Zeit-Längen-Umwandlung	
	'Filter' = 1	Minimaler Filter
	'Filter' = 2	Maximaler Filter
	'Filter' = 0 und andere	Kein Filter

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion erzeugt über das Längensignal 'Length' (in m) eine längenbasierte Version von 'Expression' mit einer Verzögerung von 'DelayInMeter' Meter. Änderungen im Längensignal, die 'MaxLengthDelta' überschreiten, werden ignoriert. Die Auflösung ist die Längenbasis des Ergebnisses (in m).

'Filter' legt den Filter bei der Zeit-Längen-Umwandlung fest.

Vor allem zur Überwachung von Qualitätsdaten in Verbindung mit ibaQDR können Messsignale von entfernten Quellen hinsichtlich der Produktlänge abgeglichen werden. Die Entfernung zwischen den Signalquellen bestimmt den Wert 'DelayInMeter'.

### 2.7.7 DelayLengthV

```
DelayLengthV('Expression','Speed','DelayInMeter','Resolution*', 'Filter=0*')
```

#### Argumente

'Expression'	Eingangssignal	
'Speed'	Geschwindigkeitssignal (in m/s)	
'DelayInMeter'	Verzögerung (in m)	
'Resolution*'	Auflösung: Längenbasis des Ergebnisses (in m)	
'Filter*'	Optionaler Parameter (Voreinstellung =0) zur Einstellung des Filters bei Zeit-Längen-Umwandlung	
	'Filter' = 1	Minimaler Filter
	'Filter' = 2	Maximaler Filter
	'Filter' = 0 und andere	Kein Filter

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Diese Funktion erzeugt über das Geschwindigkeitssignal 'Speed' (in m/s) eine längenbasierte Version von 'Expression' mit einer Verzögerung von 'DelayInMeter' Meter. Die Auflösung ist die Längenbasis des Ergebnisses (in m).

'Filter' legt den Filter bei der Zeit-Längen-Umwandlung fest.

## 2.7.8 DWORD

DWORD('Low', 'High')

### Argumente

'Low'	16 bit Integer: Word	
'High'	16 bit Integer: Word	

### Beschreibung

Diese Funktion gibt als Ergebnis den 32 bit Integer DWORD wieder, bestehend aus den int16 WORDS 'Low' und 'High'.

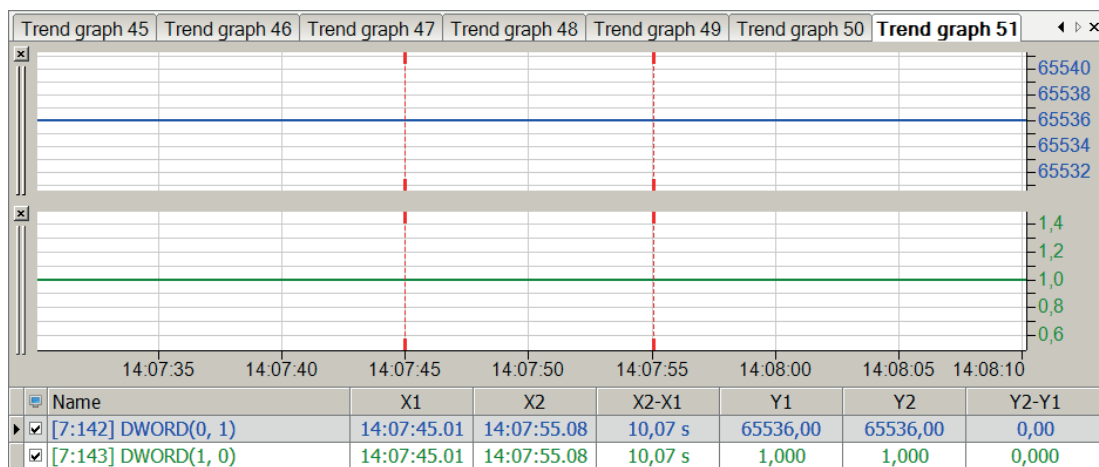
### Beispiel

Berechnung zweier DWORDS aus einer einfachen 0 und 1 Kombination für 'Low' und 'High'.

	aus int16 WORDS zusammengesetztes DWORD			
DWORD('Low', 'High')	int16 'High'	int16 'Low'	Umrech.	Dez.
DWORD(0, 1)	0000 0000 0000 0001	0000 0000 0000 0000	216	65536
DWORD(1, 0)	0000 0000 0000 0000	0000 0000 0000 0001	20	1

### Lösung

Die nachfolgende Abbildung zeigt die Berechnung zweier einfacher DWORDS.



## 2.7.9 ElapsedTime

ElapsedTime('Start','Stop')

### Beschreibung

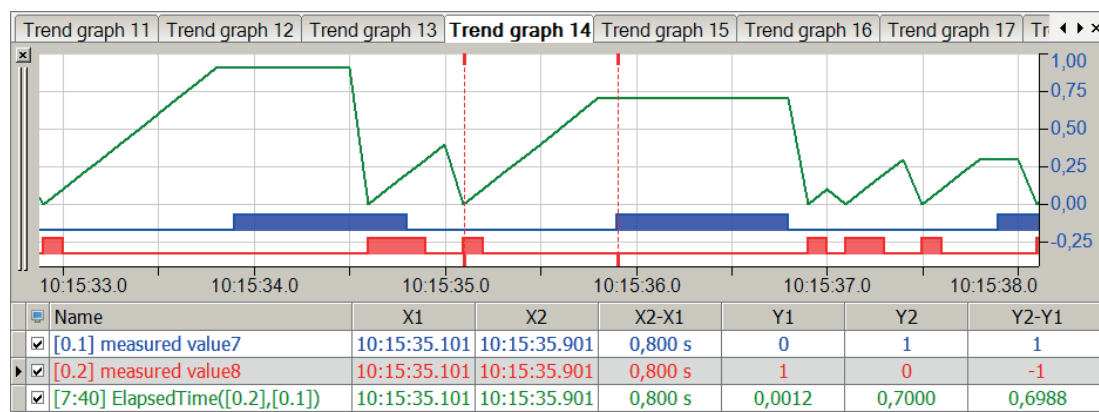
Diese Funktion gibt die Zeit wieder, die seit der letzten steigenden Flanke bei 'Start' vergangen ist. Die Zeit wird an der steigenden Flanke bei 'Stop' angehalten.

### Beispiel

Messung des Zeitabstandes zwischen der letzten steigenden Flanke eines Messwertes bis zur nächsten steigenden Flanke eines anderen Messwertes.

### Lösung

In der nachfolgenden Abbildung zeigt der blaue Balken den Messwert 'Stop' und der rote Balken zeigt den Messwert 'Start'. Die grüne Kurve zeigt die abgelaufene Zeit zwischen den steigenden Flanken von Messwert 'Start' bis Messwert 'Stop'.



### Hinweis



Durch das Schalten der Flanken im Messpunkttraster kann die grafische Darstellung und die Berechnung der abgelaufenen Zeit um einen Messpunkt abweichen.



## 2.7.10 ExecuteCommand

```
ExecuteCommand('Trigger', '"Command"', '"Arguments"', '"UserName"', '"Password"', 'Timeout=0')
```

### Argumente

'Trigger'	Binäres Signal oder Ausdruck als Auslöser für die Ausführung des Befehls
""Command""	Vollständiger und absoluter Pfad zu dem auszuführenden Befehl (.exe, .bat usw.)
""Arguments""	Für die Ausführung des Befehls erforderliche Argumente/Parameter
""UserName""	Angabe eines Benutzernamens, unter dem der Befehl ""Command"" ausgeführt werden darf
""Password""	Angabe des passenden Kennworts zum angegebenen Benutzer
'Timeout'	Optional Parameter; wenn angegeben und >0, wird die Ausführung abgebrochen, sobald die Zeit 'Timeout' [s] abgelaufen ist.

### Beschreibung

Diese Funktion führt die Kommandozeile ""Command"" ""Arguments"" bei einer steigenden Flanke von 'Trigger' aus. Der Rückgabewert ist 1 (TRUE), solange der aufgerufene Prozess läuft. Die Funktion ist daher als Ausdruck für Digitalsignale zu verwenden. Weitere steigende Flanken werden ignoriert, solange der Prozess läuft.

Bei ""Command"" ist der vollständige, absolute Pfad zur ausführbaren Datei aus Sicht des *ibaPDA*-Servers anzugeben.

Wenn 'Timeout' nicht angegeben wird oder kleiner/gleich null ist, wird die Kommandoausführung nicht unterbrochen und kann beliebig lange laufen. Wird für 'Timeout' ein Wert größer null angegeben, dann wird die Ausführung des Kommandos abgebrochen, wenn sie innerhalb von 'Timeout' Sekunden nicht abgeschlossen wurde.

Für die Nutzung der Funktion ist die Angabe von Benutzername (""UserName"" ) und Kennwort (""Password"" ) erforderlich. Aus Sicherheitsgründen müssen die von der Funktion aufgerufenen Kommandos oder Programme mit einem dedizierten Benutzerkonto ausgeführt werden. Das mit 'UserName' und 'Password' spezifizierte Benutzerkonto muss die Berechtigung zur Ausführung des eingetragenen Kommandos besitzen.

Bei der Eingabe von Argument, Benutzername und Kennwort können Sie die Verschlüsselungsfunktion des Ausdruckseditors nutzen, um diese Angaben unkenntlich zu machen. Die Verschlüsselungsfunktion wird Ihnen angeboten, sobald Sie Text zwischen zwei Anführungszeichen ("...") eingeben.

### Hinweis



Wenn Sie mit diesem Kommando Programme aufrufen, die normalerweise über eine Benutzeroberfläche (GUI) verfügen, dann ist die GUI dieser Programme nicht sichtbar. Da das gestartete Programm in der Dienstsession läuft, ohne Verbindung zum Desktop, ist es im Taskmanager zwar zu sehen, aber die GUI dazu fehlt.

## Beispiel 1

Rechner herunterfahren

### Aufgabenstellung

Abhängig von einem Signal aus der Anlagensteuerung wegen Umschaltung auf unterbrechungsfreie Stromversorgung bei einem Stromausfall soll der *ibaPDA*-Rechner gesteuert heruntergefahren werden (Shutdown).

### Lösung

Für das Herunterfahren des Rechners wird das Kommando "shutdown" aus der Windows Command-Shell genutzt. Da mehrere Befehle, wie das Stoppen des *ibaPDA*-Dienstes erforderlich sind, wird eine Batch-Datei verwendet.

```
ExecuteCommand([4.14], "D:\Schulung\ibaPDA_ExecuteCommand\Shutdown_PC.bat", "", "MyUsername", "MyPassword")
```

- [4.14]: Digitales Triggersignal zum Ausführen des Kommandos
- D:\Schulung\ibaPDA\_ExecuteCommand\Shutdown\_PC.bat: Programmpfad für die Batch-Datei
- MyUsername und MyPassword: Angaben zum Benutzerkonto

Inhalt der Batch-Datei `Shutdown_PC.bat`:

1	SETLOCAL
2	sc stop ibaPDAService
3	shutdown /f /s /t 10
4	ENDLOCAL

## Beispiel 2

Batchdatei starten, die eine Textdatei mit aktuellem Inhalt erzeugt.

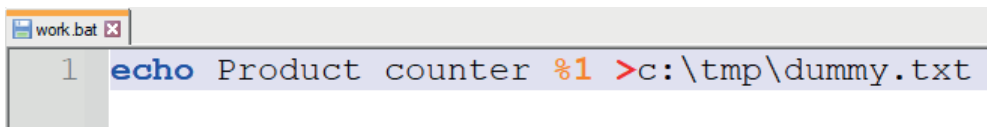
### Aufgabenstellung

Mit steigender Flanke eines binären Signals soll ein aktueller Zählerstand in eine Textdatei auf der Festplatte geschrieben werden.

### Lösung

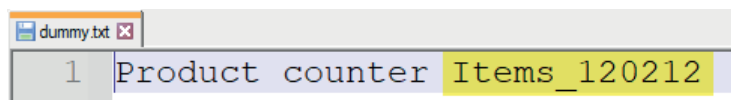
```
ExecuteCommand([3.17], "c:\tmp\work.bat", GenerateText("Items_%1"), "MyUserName", "MyPassword", 0)
```

- [3.17]: Digitales Triggersignal zum Ausführen des Kommandos
- `c:\tmp\work.bat`: Programmpfad für die auszuführende Batchdatei `work.bat`; die Angabe einer Batch-Datei ist gleichbedeutend mit einem Kommando.  
Inhalt der Batchdatei (Beispiel):



```
1 echo Product counter %1 >c:\tmp\dummy.txt
```

- `GenerateText("Items_%1")`: Argument für die Batchdatei; das Ergebnis dieses Ausdrucks (erzeugen eines Strings mit Zählerwert) wird als Parameter %1 der Batchdatei in die Textdatei `dummy.txt` eingefügt.  
Inhalt der Textdatei `dummy.txt` (Beispiel, dynamischer Inhalt gelb markiert):



```
1 Product counter Items_120212
```

- `MyUserName, MyPassword`: Anmeldeinformationen für das Benutzerkonto
- `0`: Wert für Timeout, d. h., die Ausführung wird nicht abgebrochen.

## 2.7.11 GenerateSignal

```
GenerateSignal('Type', 'Amplitude=10', 'T1=1', 'T2=1')
```

### Argumente

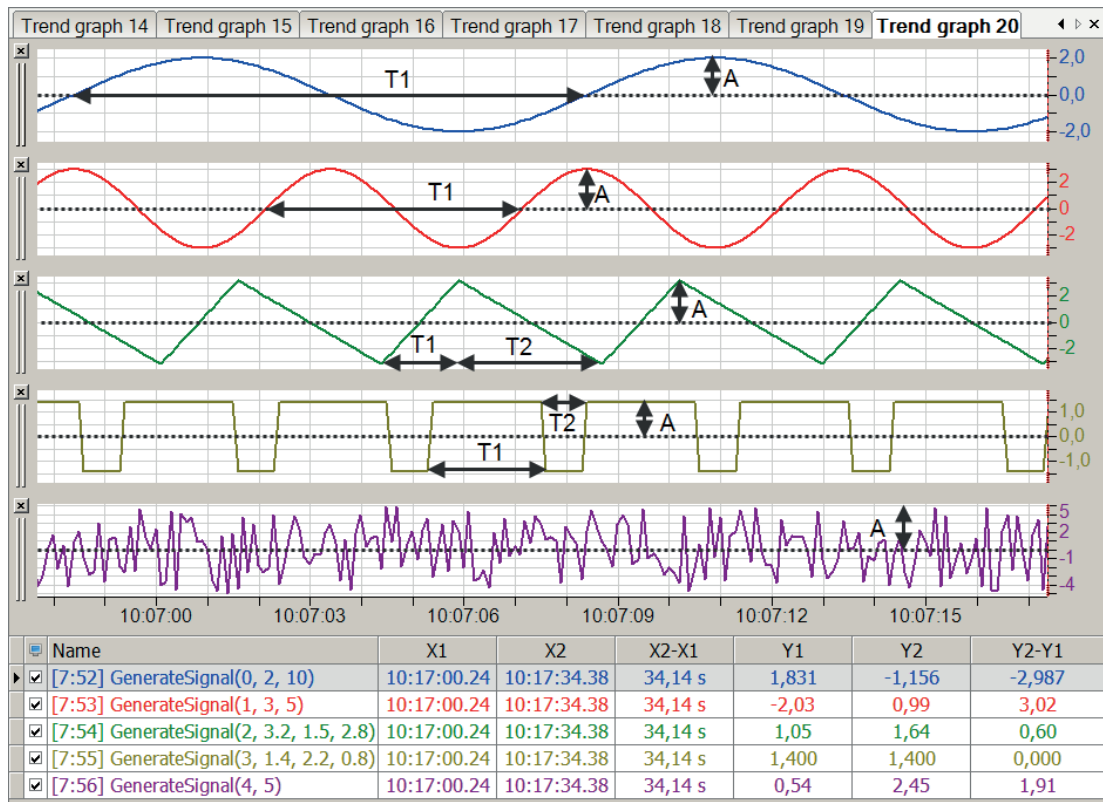
'Type'	Angabe des Signaltyps	
	'Type' = 0	Sinusfunktion, mit A=Amplitude und T1=Periodendauer
	'Type' = 1	Cosinusfunktion, mit A=Amplitude und T1=Periodendauer
	'Type' = 2	Dreiecksfunktion, mit A=Amplitude, T1 = Zeitangabe ansteigende Flanke, T2 = Zeitangabe absteigende Flanke
	'Type' = 3	Rechteckfunktion, mit A=Amplitude, T1 = Zeitangabe obere Flanke, T2 = Zeitangabe untere Flanke
	'Type' = 4	Zufallssignal mit A=maximaler Amplitude
'Amplitude'	Optional Parameter, Angabe der Amplitude; Voreinstellung = 10	
'T1'	Optional Parameter, Zeitangabe 1 (außer bei Typ 4); Voreinstellung = 1	
'T2'	Optional Parameter, Zeitangabe 2 (nur bei Typ 2 und Typ 3); Voreinstellung = 1	

### Beschreibung

Diese Funktion generiert ein Testsignal mit der Amplitude 'Amplitude' und den Zeiträumen 'T1' und 'T2'. Folgende Signaltypen können erzeugt werden:

**Beispiel**

verschiedene Beispiele für generierte Signale

**Lösung**

Blau	Sinusfunktion Typ 0	Rot	Cosinusfunktion Typ 1
Grün	Dreiecksfunktion Typ 2	Gelb	Rechteckfunktion Typ 3
Lila	Zufallsfunktion Typ 4		

## 2.7.12 GenerateText

`GenerateText ('Text*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Argumente

'Text'	Text, der als Inhalt des Textsignals erzeugt werden soll. Der Text muss zwischen Anführungszeichen ("...") stehen. Der Text wird einmalig zum Start der Erfassung übernommen.	
	%n	Parameter, der in den Text eingefügt werden kann und im erzeugten Text durch einen automatischen Zählerwert ersetzt wird. Der Zähler zählt mit jedem n-ten Zyklus, basierend auf der Zeitbasis des <i>ibaPDA</i> -Systems, um 1 aufwärts, z. B.  %1 - mit jedem Zyklus %2 - mit jedem zweiten Zyklus %10 - mit jedem zehnten Zyklus %100 - mit jedem 100. Zyklus

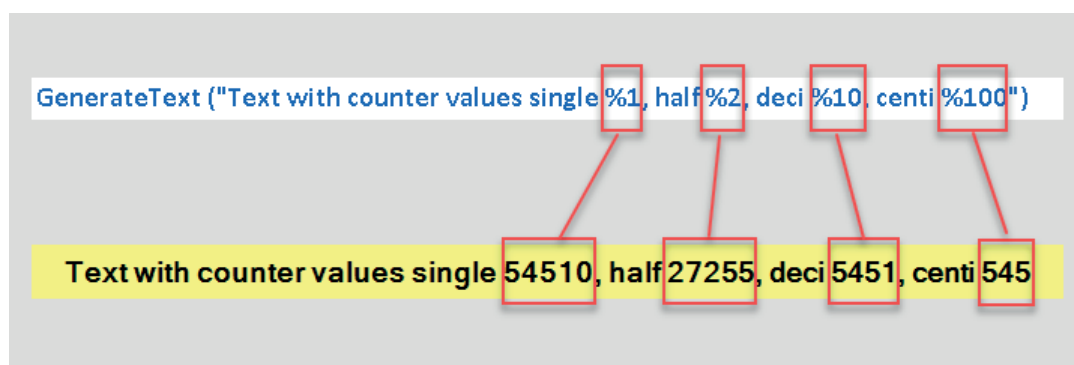
### Beschreibung

Diese Funktion erzeugt ein Textsignal. Sie kann nur in der Signaltabelle *Analog* eines geeigneten Moduls (z. B. virtuelles Modul oder Ausgangsmodul) genutzt werden.

Das erzeugte Textsignal kann wie andere Textsignale verwendet werden, u. a. in einem Texterzeuger-Modul, einer digitalen Textanzeige oder eine QPanel-Textfeld.

Sie können Platzhalter %1, %2, ... im Text verwenden. Der Platzhalter %n wird durch den Wert eines Zählers ersetzt, der alle n Samples erhöht wird.

### Beispiel



Die obere Zeile zeigt den vollständigen Funktionsaufruf. Die untere Zeile zeigt das Ergebnis in einem Textfeld in ibaQPanel.

### 2.7.13 GetFloatBit

```
GetFloatBit('Expression', 'BitNo')
```

#### Argumente

'Expression'	Beliebige Zahl
'BitNo'	Diese Zahl, 0...15, bzw. 0...31 gibt die Position des gewünschten Digitalsignals in einem 16-Bit- bzw. 32-Bit-Block im Datenstrom an, bezogen auf die zugehörige Offset-Adresse. Erhöhung der Bit-Nr. um 1 bis 15 (31), dann Erhöhung der Adresse um 2 (4).

#### Beschreibung

Diese Funktion interpretiert 'Expression' als Bitmaske eines Floatwertes und liefert als Ergebnis den booleschen Wert des Bits 'BitNumber' von 'Expression'. Gültige Bitnummern-Reihenfolge: 0 (LSB) bis 31 (MSB).

Diese Funktion wurde speziell für den Fall entwickelt, wenn 32 Bits für die Übertragung/Aufzeichnung als Float-Daten gepackt werden. Die Funktion GetBitFloat wertet lediglich die Valenz des spezifizierten Bits 'BitNo' aus, ungeachtet dessen, ob es Teil der Mantisse oder des Exponenten ist. Im Gegensatz zur Funktion GetBit werden die Werte nicht gerundet.

#### Beispiel

Ein in 'Expression' eingegebener Wert wird in das Gleitkommaformat nach IEEE 754 umgewandelt und die Stellen, die dafür auf TRUE gesetzt sind, abgerufen.

#### Lösung

Gleitkommazahlen nach dem IEEE 754 Format können allgemein wie folgt geschrieben werden: (Vorzeichen)\*Mantisse\*10^Exponent

Die Zahl 0,15625 lautet als Gleitkommazahl nach IEEE 754 wie folgt:

Vorzeichen (1 bit)	Exponent (8 bit)	Mantisse (23 bit)
0	0111 1100	0100 0000 0000 0000 0000 000

Dabei ist das LSB (Least Significant Bit) an äußerster rechter Stelle.

**Tipp**

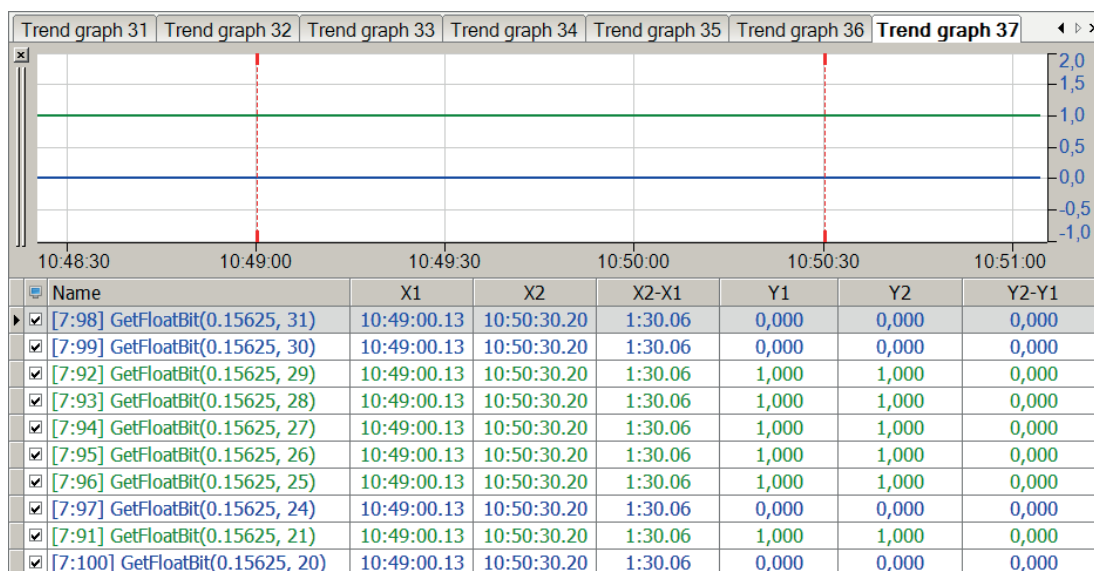
Gleitkommazahlen nach dem IEEE 754 Format werden in der dargestellten Weise geschrieben. Dabei sind bei den einzelnen Komponenten folgende Punkte zu beachten:

Vorzeichen: 0 steht für positiv; 1 für negativ

Exponent: Da bei der Umrechnung in eine Gleitkommazahl der Exponentenwert um 127 verschoben wird, müssen von dem hier dargestellten Exponenten bei der Umrechnung in eine Dezimalzahl wieder 127 subtrahiert werden.

Mantisse: Da die Mantisse standardmäßig eine 1 vor dem Komma hat, wird diese Stelle nicht mehr dargestellt. Alle Mantissenstellen sind deshalb Nachkommastellen und müssen bei der Umrechnung mit negativen Exponenten zur Basis 2 berücksichtigt werden.

Die nachfolgende Abbildung zeigt den Abruf der Valenz ausgewählter Bits des Floatwertes von 0,15625.





## 2.7.14 GetIntBit

```
GetIntBit('Expression', 'BitNo')
```

### Beschreibung

Diese Funktion liefert als Ergebnis den booleschen Wert des Bits 'BitNo' von 'Expression' nach Rundung von 'Expression' auf den nächsten Integerwert. Die Rundungsgrenze liegt jeweils bei 0,5-Schritten. (2,49 → 2; 2,50 → 3).

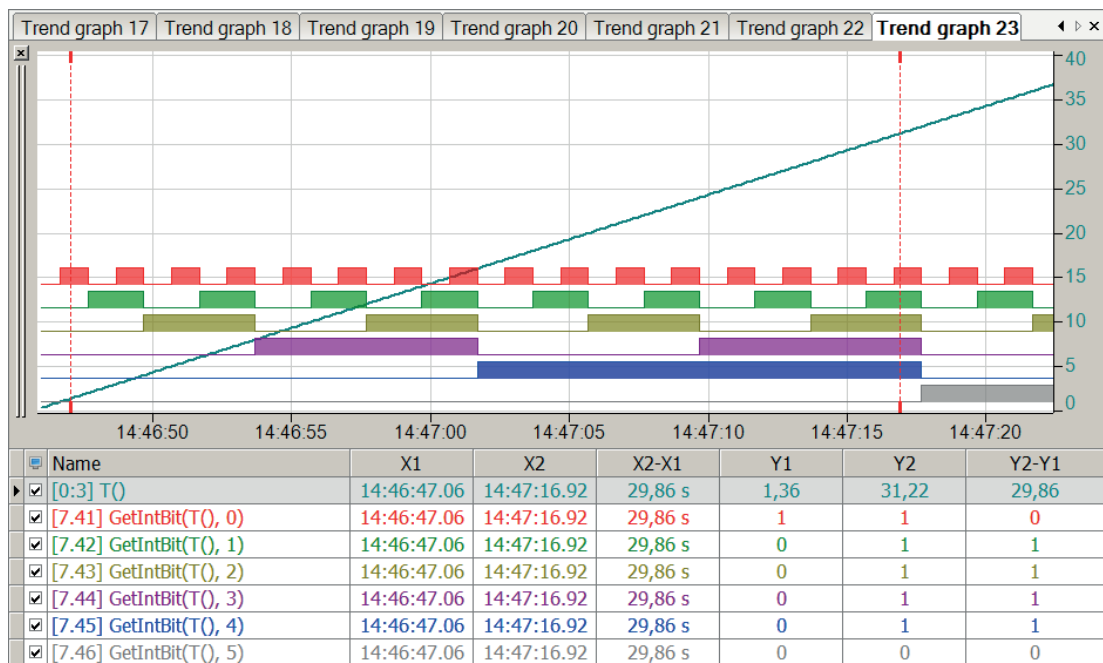
Gültige Bitnummernreihenfolge: 0 (LSB) bis 31 (MSB)

### Beispiel

Darstellung der Zeit in booleschen Werten

### Lösung

Die nachfolgende Abbildung zeigt die Funktion GetIntBit der ersten sechs Bits der Zeitfunktion T().



### 2.7.15 GetSignalMetaData

```
GetSignalMetaData('Signal*', 'InfoType*')
```

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Diese Funktion gibt verschiedene Metadaten-Informationen eines Signals zurück. Der Parameter 'Signal' kann als Signalnummer ([m:n] bzw. [m.n]) oder als Signalname ([SignalName]) angegeben werden.

Geben Sie den Informationstyp ('InfoType') an, den Sie erhalten wollen.

Folgende Informationstypen werden unterstützt:

Informationstypen	Mögliche Ergebnisse
0	Signalname
1	Signaleinheit
2	Kommentar 1
3	Kommentar 2

Tab. 4: Informationstypen und mögliche Ergebnisse der Funktion GetSignalMetaData

## 2.7.16 GetSystemTime

```
GetSystemTime('Part*')
```

### Argumente

'Part*'	Auszugebener Teil der Systemzeitangabe	
	'Part' = 0	Millisekunden
	'Part' = 1	Sekunden
	'Part' = 2	Minuten
	'Part' = 3	Stunden
	'Part' = 4	Tag des Monats
	'Part' = 5	Monat
	'Part' = 6	Jahr
	'Part' = 7	Tag des Jahres
	'Part' = 8	Wochentag (Montag=1, Sonntag=7)

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den unter 'Part' angegebenen Teil der Systemzeit zurück.

### Tipp



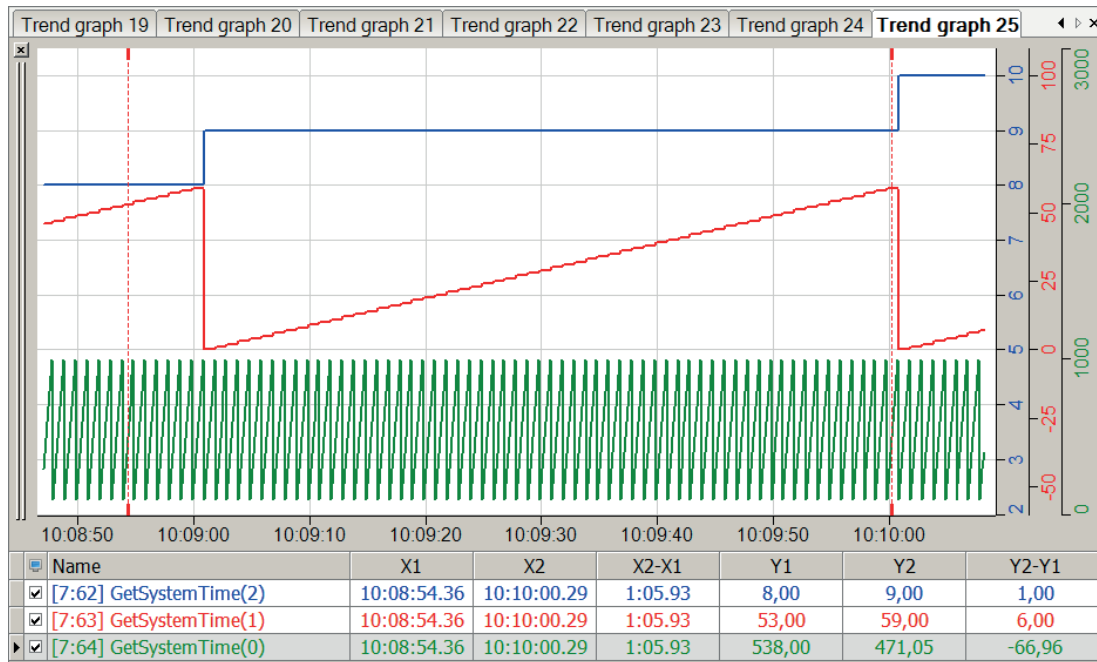
In *ibaQPanel* kann der Wochentag (Teil 8) leicht mit seinem Namen über die Mehrfachstatusanzeige dargestellt werden. Für die anderen Teile ist die normale Bezeichnung völlig ausreichend. Verwenden Sie eine Messuhr mit Rundanzeige um in *ibaQPanel* die Zeit wie auf einer analogen Uhr anzeigen zu lassen.

## Beispiel

Ausgabe des Minuten-Teils, Sekunden-Teils und Millisekunden-Teils der Systemzeit.

## Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve die Minuten der Systemzeit ('part'=2) und die rote Kurve zeigt die Sekunden der Systemzeit ('part'=1). Die grüne Kurve zeigt die Millisekunden der Systemzeit ('part'=0').



## 2.7.17 GetSystemTimeAsText

```
GetSystemTimeAsText('DateTimeFormat*="yyyy-MM-dd HH:mm:ss"', 'UTC*=0', 'Enable=1')
```

### Argumente

'Argument'	Beschreibung	
'DateTimeFormat'	yy, yyyy	Jahr, 2- oder 4-stellig
	M, MM	Monat M: 1, 2, 3,...12; MM: 01, 02, 03,...12
	MMM, MMMM	Monatsname
	d, dd	Tag d: 1, 2, 3, ...31; dd: 01, 02, 03, ... 31
	ddd, dddd	Wochentag
	h, hh,	Stunde 12h-Format
	H, HH	Stunde 24h-Format
	m, mm	Minute
	s,ss	Sekunde
	f, ff, fff	Sekundenbruchteile bis 0.1, 0.01, 0.001 (Millisekunde), ... Genauigkeit
	tt	AM/PM-Angabe; nur nutzbar, wenn ibaPDA-Dienst in einer Sprache läuft, die AM/PM unterstützt (z. B. US-Englisch)
'UTC'	0 (default)	Datum/Uhrzeit lokal
	1	UTC-Datum und -Uhrzeit
'Enable'	1	Optionales Argument; wenn 'Enable' genutzt wird, dann werden Datum/Uhrzeit nur ausgegeben, wenn 'Enable' = 1 bzw. True ist.  Wenn das Argument weggelassen wird, dann werden Datum/Uhrzeit immer ausgegeben.

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt das aktuelle Datum und die aktuelle Uhrzeit zurück. Abhängig vom Parameter 'UTC' werden Datum und Uhrzeit als lokale Angabe oder als UTC-Angabe zurückgegeben.

Mit dem optionalen Parameter 'Enable' kann die Ausgabe gesteuert werden. Möchten Sie auch die genauere Zeitangabe mit Sekundenbruchteilen nutzen, muss der Parameter 'Enable' vorhanden sein und den Wert 1 bzw. True haben.

**Beispiele**

Beispiele für Dienstag, den 07. August 2023, 17:05 Uhr (MESZ)

Formel	Ergebnis
<code>GetSystemAsText ("yyyy-MM-dd HH:mm:ss", 0)</code>	2023-08-07 17:05:42
<code>GetSystemAsText ("yyyy-MM-dd HH:mm:ss", 1)</code>	2023-08-07 15:05:42
<code>GetSystemAsText ("yy-MMM-dd hh:mm:ss", 1)</code>	23-Aug-07 03:05:42
<code>GetSystemAsText ("yyyy-MM-dd HH:mm:ss fff", 0)</code>	2023-08-07 17:05:42 478

## 2.7.18 GetWeekOfYear

```
GetWeekOfYear('Rule*=0', 'FirstDayOfWeek*=0')
```

### Argumente

'Rule*'	Definitionsregel für erste Woche des Jahres	
	'Rule' = 0	Erste Woche des Jahres ist die Woche, die mindestens 4 Tage des neuen Jahres enthält (ISO 8601)
	'Rule' = 1	Erste Woche des Jahres beginnt am 1. Januar
	'Rule' = 2	Erste Woche des Jahres beginnt am ersten Tag der Woche am oder nach dem 1. Januar
'FirstDayOf-Week*'	Definition erster Tag der Woche	
	'FirstDayOfWeek' = 0	Montag (ISO 8601)
	'FirstDayOfWeek' = 1	Dienstag
	'FirstDayOfWeek' = 2	Mittwoch
	'FirstDayOfWeek' = 3	Donnerstag
	'FirstDayOfWeek' = 4	Freitag
	'FirstDayOfWeek' = 5	Sonnabend
	'FirstDayOfWeek' = 6	Sonntag

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt die Nummer der aktuellen Woche des aktuellen Jahres zurück. Die Wochennummerierung erfolgt entsprechend der Regel 'Rule'.

## 2.7.19 LimitAlarm

```
LimitAlarm('Expression', 'Limit', 'DeadBand', 'Time', ' Reset=0')
```

### Argumente

'Expression'	Messwert	
'Limit'	Grenzwert, ab dem Funktion TRUE zurückgibt	
'DeadBand'	Angabe einer Totzone unterhalb des Grenzwertes, innerhalb der die Funktion nicht auf FALSE zurückgesetzt wird	
'Time'	Angabe der Zeit, die der Messwert oberhalb des Grenzwertes liegen muss bis die Funktion auf TRUE gesetzt wird	
'Reset'	Optionalen Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset' = 0	Berechnung durchführen
	'Reset' = 1	Berechnung anhalten, zurücksetzen und Ergebnis auf 0 setzen

	'Reset' = 2	Berechnung anhalten, zurücksetzen und Ergebnis beibehalten
--	-------------	--

### Beschreibung

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

### Tipp



Die Funktion LimitAlarm kann auch für einen unteren Grenzwert verwendet werden. Dazu müssen lediglich der Messwert und der Grenzwert gespiegelt werden, d. h. multipliziert mit (-1).

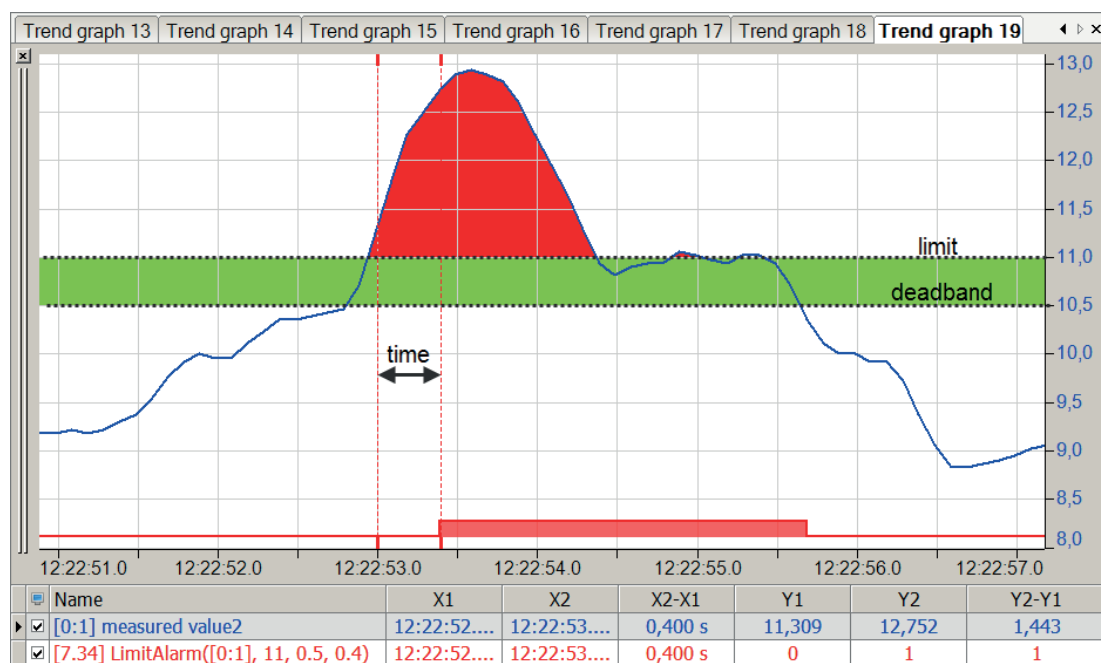
z. B.: `LimitAlarm([0:1] *(-1), 9 *(-1), 0.5, 0.4)`

### Beispiel

Die Funktion soll TRUE zurückgeben wenn der Messwert länger als 0,4 Sekunden oberhalb des Wertes 11 ist. Die Funktion soll erst dann wieder FALSE zurückgeben, wenn der Messwert unter 10,5 gefallen ist.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert und der rote Balken zeigt die Überschreitung des Grenzwertes. Das grüne Band zeigt die Totzone, die ein direktes FALSE-Setzen der Funktion verhindert.





## 2.7.20 ModuleSignalCount

```
ModuleSignalCount('ModuleNo*', 'SignalType*=0', 'Direction*=0')
```

### Argumente

'SignalType*'	Signalart	
	'SignalType' = 0	Analog- und Digitalsignale
	'SignalType' = 1	Nur Analogsignale
	'SignalType' = 2	Nur Digitalsignale
'Direction*'	Ein- oder Ausgaberrichtung oder beides	
	'Direction'= 0	Ein- und Ausgänge
	'Direction'= 1	Nur Eingänge
	'Direction'=2	Nur Ausgänge

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt die Anzahl der aktiven Signale in Modul mit der Nummer 'ModuleNo' zurück. Welche Signalarten dabei berücksichtigt werden, bestimmen Sie mit den Parametern 'SignalType' und 'Direction'. Signale, die im Modul zwar konfiguriert sind, aber kein Häkchen in der Spalte *Aktiv* haben, werden nicht mitgezählt.

## 2.7.21 PulseFreq

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

### Argumente

'Expression'	Pulszählersignal	
'Omega*'	Filterfrequenz	
'EdgeType*'	Flankenart, die gezählt werden soll	
	'EdgeType' = -1	nur fallende Flanken
	'EdgeType' = 0	steigende und fallende Flanken
	'EdgeType' = 1	nur steigende Flanken
	'EdgeType' = 2	'Expression' ist ein Impulszähler

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

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

Ein Tiefpassfilter mit einer Grenz-Winkelgeschwindigkeit '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.

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.

## 2.7.22 RestartAcquisition

```
RestartAcquisition('Trigger')
```

### Beschreibung

Die Erfassung wird neu gestartet, wenn 'Trigger' eine steigende Flanke hat. Die Funktion gibt den Wert 1 zurück, wenn die Erfassung neu gestartet wurde.

## 2.7.23 SampleAndHold

SampleAndHold('Expression', 'Sample', 'Initial=0)

### Argumente

'Expression'	Messwert
'Sample'	Parameter, der bestimmt, ob die Funktion dem Messwert folgt (1) oder den letzten Messwert hält (0). 'Sample' kann selbst eine Bedingung sein oder durch eine andere Funktion bestimmt werden.
'Initial'	Optional Parameter (Voreinstellung = 0), der den Initialwert der Funktion festlegt, wenn bei Beginn der Messung 'Sample' inaktiv ist.

### Beschreibung

Diese Funktion ist eine Abtast-Halte-Funktion. Der Ausgang folgt 'Expression', wenn 'Sample' = TRUE. Er bleibt unverändert, wenn 'Sample' = FALSE. Mit dem optionalen Parameter 'Initial' kann der Initialwert des Ausgangs angegeben werden, wenn die Funktion beim Aufruf auf "Halten" steht.

### Beispiel

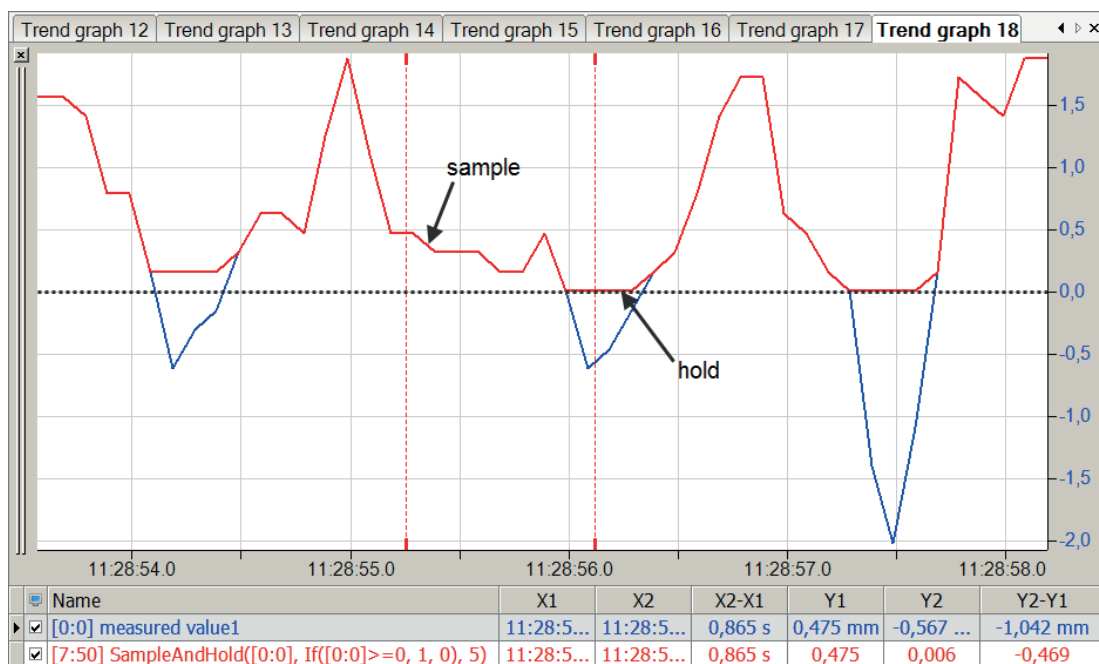
Die Funktion soll einem Messwert folgen, wenn dieser positiv ist und gehalten werden, sobald der Messwert negativ wird.

### Aufgabenstellung

Eine If-Abfrage als 'Sample'-Parameter gibt den Wert 1 für positive Messwerte und den Wert 0 für negative Messwerte aus.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert und die rote Kurve zeigt den Ausgangswert mit gehaltenen Messwerten.



## 2.7.24 SampleOnce

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

### Argumente

'Expression'	Messwert
'Sample'	Parameter, der bestimmt, wann die Funktion dem Messwert 'Expression' für ein Sample folgt. 'Sample' kann selbst eine Bedingung sein oder durch eine andere Funktion bestimmt werden.

### Beschreibung

Diese Funktion ist eine einmalige Abtastfunktion. Das Ergebnis der Funktion folgt 'Expression' für die Dauer eines Samples, wenn 'Sample' eine steigende Flanke hat. Zu allen anderen Zeitpunkten ist das Ergebnis NaN, wenn 'Expression' ein numerisches Signal ist, und ein leerer String, wenn 'Expression' ein Textsignal ist.

## 2.7.25 Sign

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

### Beschreibung

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

'Expression' > 0 --> +1

'Expression' = 0 --> 0

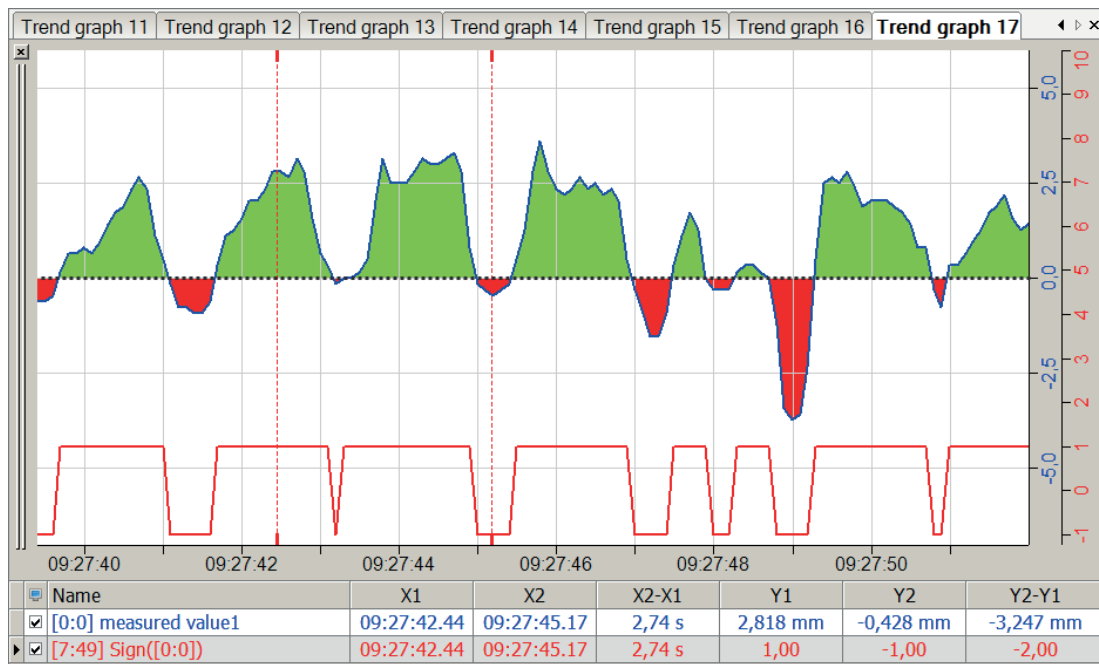
'Expression' < 0 --> -1

### Beispiel

Von einem Messwert ist nur das Vorzeichen relevant.

### Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert und die rote Kurve zeigt die Vorzeichen des Messwerts.



## 2.7.26 T

T ()

### Beschreibung

Diese Funktion gibt die seit dem Start der Messung verstrichene Zeit wieder (in Sekunden).

Mithilfe der Zeitfunktion können zeitabhängige, virtuelle Größen berechnet werden, z. B. Sinuskurven.

Beispiel einer Sinuskurve mit 0,5 Hz:  $\sin(2 \cdot \pi() \cdot 0,5 \cdot T())$

### Tipp



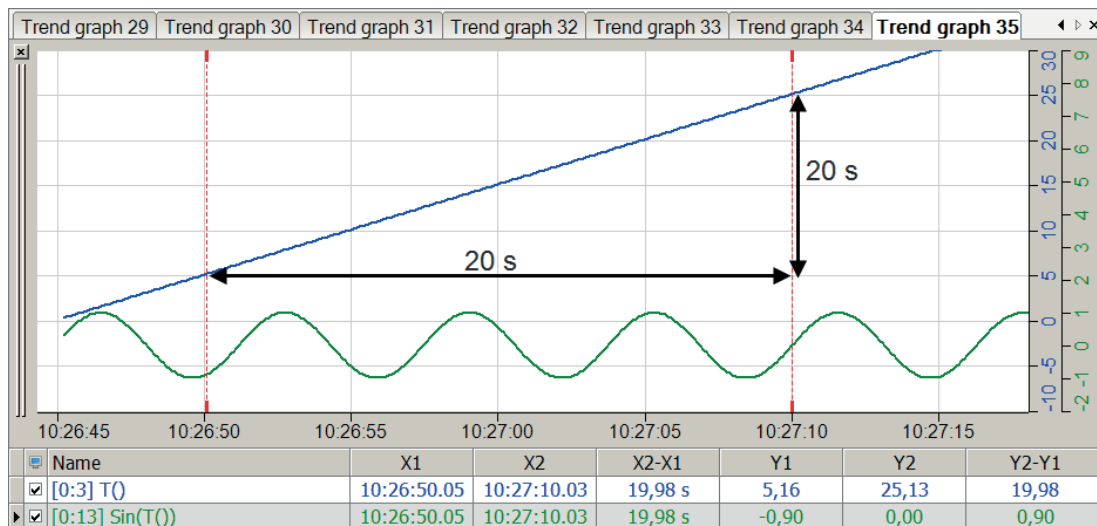
Die Zeitfunktion kann im Ausdruckseditor einfach mit Doppelklick in andere Ausdrücke eingefügt werden, wie bei einem Signal. Die beiden Klammern hinter dem T sind aus syntaktischen Gründen obligatorisch, genau wie bei der Zahl Pi.

### Beispiel

Darstellung der Zeitfunktion und einer zeitabhängigen, virtuellen Größe

### Lösung

Berechnung der Zeitfunktion und einer zeitabhängigen Sinuskurve



## 2.7.27 VarDelay

```
VarDelay('Expression','Delay', 'MaxDelay=30*')
```

### Argumente

'Expression'	Eingangssignal
'Delay'	Verzögerungszeit in Sekunden
'MaxDelay*'	Optionaler Parameter (Voreinstellung =30) zur Festlegung der maximal zulässigen Verzögerung in Sekunden

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion verzögert das Signal 'Expression' um die Verzögerungszeit 'Delay'. Im Unterschied zur Delay Funktion kann sich dabei jedoch die Verzögerungszeit mit der Zeit ändern. 'MaxDelay' gibt die maximal zulässige Verzögerung an und ist mit 30 s standardmäßig vorgelegt.

### Beispiel

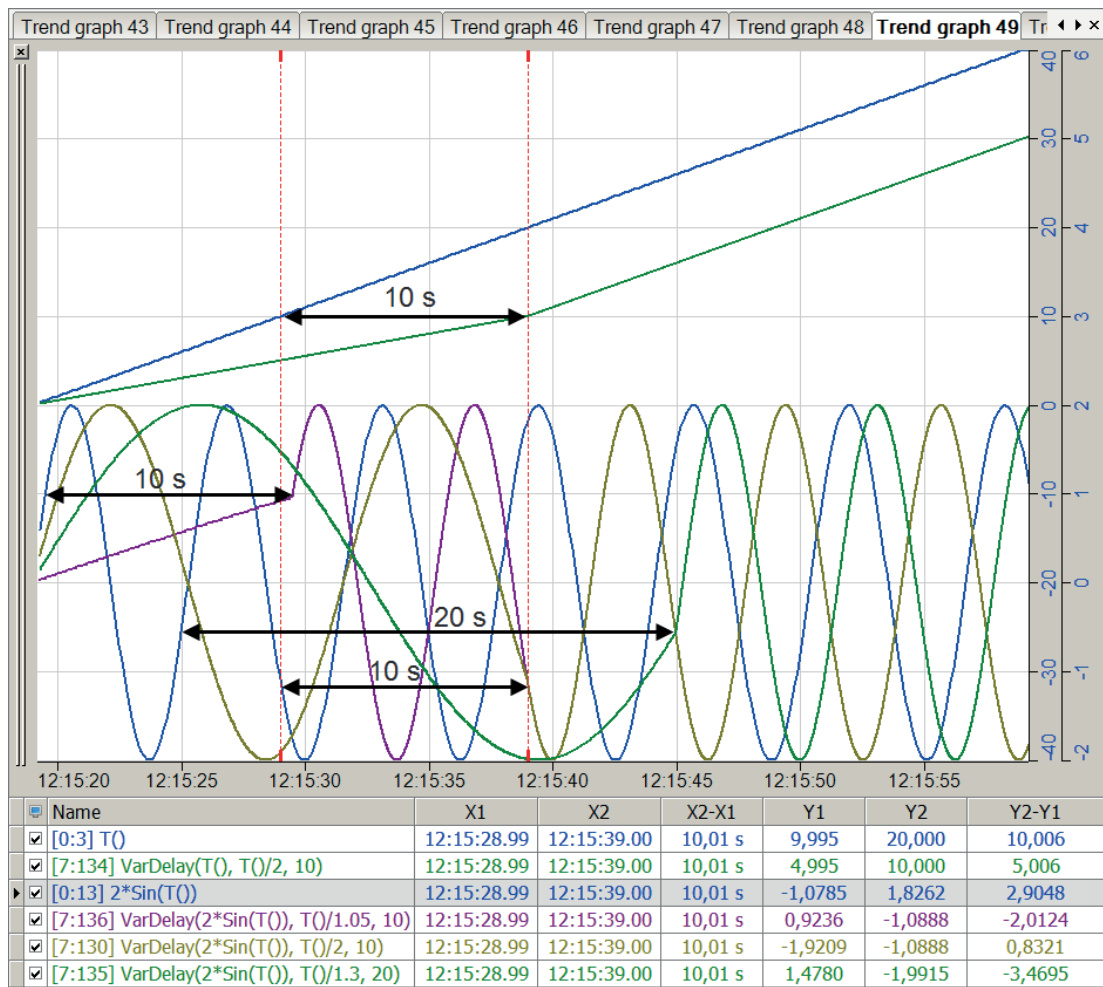
Zeitlich variable Verzögerung eines Signals mit Erreichen der maximal zulässigen Verzögerung

### Lösung

Als veränderliche Verzögerung wird die Zeitfunktion T() verwendet. Diese selbst und ein Sinus-Signal werden mit dieser variablen Verzögerung belegt und die Ergebniskurven aufgezeichnet.

Sobald die maximal zulässige Verzögerung erreicht ist, bleibt die Verzögerung konstant und die Ergebniskurven stellen nur noch die verschobenen Ausgangssignale dar.

In der nachfolgenden Abbildung zeigt der obere Teil die Verzögerung des Zeitsignals um eine variable Verschiebung mit einem Maximalwert von 10 Sekunden. Der untere Teil zeigt die Verzögerung einer Sinus-Funktion mit drei verschiedenen Verzögerungs- und Verzögerungsmaximalwerten.



### Tipp



Negative Werte für 'Delay' geben keine Fehlermeldung, sondern werden wie 0 behandelt, d. h., es gibt keine Verzögerung.

## 2.7.28 WindowAlarm

```
WindowAlarm('Expression','Limit1','DeadBand1','Limit2','DeadBand2','Time',' Reset=0')
```

### Argumente

'Expression'	Messwert	
'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 der Messwert größer als der obere Grenzwert oder kleiner als der untere Grenzwert sein muss, bis die Funktion auf TRUE gesetzt wird	
'Reset'	Optionaler Parameter (Voreinstellung =0) zum Stoppen und Neustarten der Berechnung	
	'Reset'=0	Berechnung durchführen
	'Reset'=1	Berechnung anhalten, zurücksetzen und Ergebnis auf 0 setzen
	'Reset'=2	Berechnung anhalten, zurücksetzen und Ergebnis beibehalten

### Beschreibung

Diese Funktion überwacht den Messwert ('Expression') und setzt das Ergebnis auf TRUE, wenn der Messwert 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 der Messwert den oberen Grenzwert um den unter Totzone1 ('DeadBand1') angegebenen Wert unter-, bzw. den unteren Grenzwert um den unter Totzone2 ('DeadBand2') angegebenen Wert überschreitet.

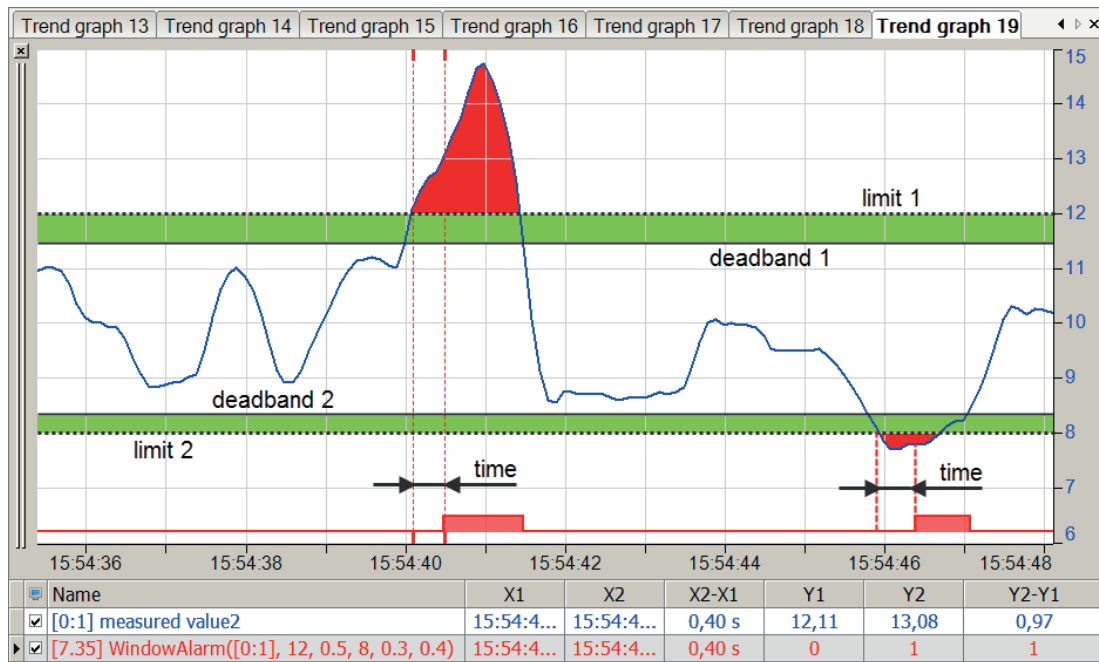


## Beispiel

Die Funktion soll ausgelöst werden, wenn der Messwert länger als 0,4 s außerhalb des Wertebereichs zwischen 8 und 12 liegt. Die obere Totzone soll dabei 0,5 betragen, die untere Totzone 0,3.

## Lösung

In der nachfolgenden Abbildung zeigt die blaue Kurve den Messwert und der rote Balken zeigt das Auslösen der Funktion WindowAlarm an. Die grünen Bereiche zeigen die Totzonen der jeweiligen Grenzwerte.



## 2.8 Diagnosefunktionen

### 2.8.1 CameraStatus

```
CameraStatus('ModuleNo*', 'SignalNo*', 'Timeout=2*')
```

#### Argumente

'ModuleNo*'	Modulnummer des ibaCapture Servers im Signalbaum	
'SignalNo*'	Signalnummer (Kamera)	
'Timeout*'	Zeit in Sekunden, in der sich das Synchronisationssignal ändern muss; Voreinstellung = 2	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Diese Funktion gibt den Status einer ibaCapture Kamera wieder.

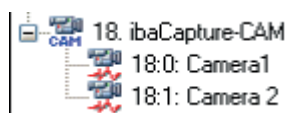
#### Ergebnisse

0	Kamera nicht OK
1	Kamera OK

Eine Kamera ist "nicht OK", wenn sich der Wert des Synchronisationssignals der Kamera nicht innerhalb von 'Timeout' Sekunden ändert. Damit sich der Wert des Synchronisationssignals ändern kann, müssen sowohl die Aufzeichnung dieser Kamera auf dem *ibaCapture*-Server als auch die Synchronisationsverbindung zwischen *ibaCapture*-Server und *ibaPDA*-Server arbeiten

'ModuleNumber' ist die Modul-Nummer des *ibaCapture*-Servers im Signalbaum.

Beispiel:



Status von "Camera1" mit Timeout = 2 s: CameraStatus(18, 0, 2)

## 2.8.2 DataStoreInfo

`DataStoreInfo('DatastoreIndex*', 'InfoType*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt Informationen über die gewählte Datenaufzeichnung. Diese Informationen können zur Steuerung anderer Funktionen oder zu Anzeige- und Diagnosezwecken genutzt werden.

Für die normale (*ibaPDA*) Datenaufzeichnung verwenden Sie 'DatastoreIndex'  $\geq 0$ .

Für die *ibaQDR*- Datenaufzeichnung verwenden Sie 'DatastoreIndex'  $< 0$ .

Den Index erfährt man leicht über die Baumstruktur im Konfigurationsdialog der Datenaufzeichnung. Index steigt von oben nach unten an.

Geben Sie den Informationstyp ('InfoType') an, den Sie erhalten wollen.

Folgende Informationstypen werden unterstützt:

Informationstypen	Mögliche Ergebnisse
0: Status Aufzeichnung	0 = Gestoppt 1 = Warten auf Trigger 2 = Aufzeichnung 3 = Posttrigger-Aufzeichnung
1: Speichern im Backup-Verzeichnis	0 = Basisverzeichnis wird verwendet 1 = Backupverzeichnis wird verwendet
2: Aufgezeichnete Zeit in der aktuellen Datei, ausgedrückt in Sekunden	Wert wird jede Sekunde aktualisiert.
3: Freier Platz auf der aktuellen Festplatte, ausgedrückt in MB	Wert wird jede Minute aktualisiert.
4: ibaQDR synchronisiert?	0 = ibaQDR ist NICHT synchronisiert 1 = ibaQDR ist synchronisiert
5: Bildtrigger speichern ins Backup-Verzeichnis	-1=Kein aktiver oder konfigurierter Bildtrigger vorhanden 0=Alle Bildtrigger speichern ins Basisverzeichnis 1=Alle Bildtrigger speichern ins Backup-Verzeichnis 2=Einige Bildtrigger speichern ins Basis- andere ins Backup-Verzeichnis
6: Anzahl der überlappenden Dateien	Aktueller Istwert

Tab. 5: Informationstypen und mögliche Ergebnisse der Funktion DataStoreInfo

### 2.8.3 DataStoreInfoDB , ...Influx, ...Kafka, ...MindSphere, ...MQTT

```
DataStoreInfoDB('DatastoreIndex*', 'InfoType*')
DataStoreInfoInflux('DatastoreIndex*', 'InfoType*')
DataStoreInfoKafka('DatastoreIndex*', 'InfoType*')
DataStoreInfoMindSphere('DatastoreIndex*', 'InfoType*')
DataStoreInfoMQTT('DatastoreIndex*', 'InfoType*')
```

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Diese Funktion gibt Informationen über die ausgewählten Datenbank- oder Cloud-Aufzeichnungen. Diese Informationen können genutzt werden zur Steuerung anderer Funktionen oder zu Anzeige- und Diagnosezwecken.

Die gewünschte DB-/Cloud-Datenaufzeichnung identifizieren Sie mit dem 'DatastoreIndex\*' >= 0.

Den Index erfährt man leicht über die Baumstruktur im Konfigurationsdialog der Datenaufzeichnung. Index steigt von oben nach unten an.

Geben Sie den Informationstyp ('InfoType') an, den Sie erhalten wollen.

Folgende Informationstypen werden unterstützt:

Informationstypen	Mögliche Ergebnisse
0: Aufzeichnungsstatus	0 = Angehalten 1 = Warten auf Trigger 2 = Aufzeichnung 3 = Aufzeichnung Nachlaufzeit
1: Datendurchsatz	Wert in KB/s
2: Ist der Server verbunden?	0/False = nein 1/True = ja
3: Aufzeichnungsdauer seit dem letzten Starttrigger	Wert in Sekunden; bei kontinuierlicher Aufzeichnung ist der Wert konstant gleich 0.
5: Aktuelle Puffernutzung	Wert in %
6: Aktuelle Dateipuffernutzung	Wert in %
7: Unverarbeitete Bytes im Dateipuffer	Wert in %

Tab. 6: Informationstypen und mögliche Ergebnisse der Funktion DataStoreInfoDB

## 2.8.4 DataStoreInfoHD

`DataStoreInfoHD('DatastoreIndex*', 'InfoType*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

#### Hinweis



Mit ibaPDA v8.4 wurden die Argumente dieser Funktion grundlegend geändert. Um alle Zustände der getriggerten HD-Aufzeichnung zu unterstützen, verwendet die Funktion jetzt die gleichen Werte für den Status (Informationstyp 0) wie andere DataStoreInfo-Funktionen auch. Wenn Sie die Funktion bereits verwenden, überprüfen Sie bitte Ihre Auswertung der Statuswerte.

Diese Funktion gibt Informationen über die ausgewählten HD-Aufzeichnungen. Diese Informationen können genutzt werden zur Steuerung anderer Funktionen oder zu Anzeige- und Diagnosezwecken.

Die gewünschte HD-Datenaufzeichnung identifizieren Sie mit dem 'DatastoreIndex\*'  $\geq 0$ .

Den Index erfährt man leicht über die Baumstruktur im Konfigurationsdialog der Daten-Aufzeichnung. Index steigt von oben nach unten an.

Geben Sie den Informationstyp ('InfoType') an, den Sie erhalten wollen.

Folgende Informationstypen werden unterstützt:

Informationstypen	Mögliche Ergebnisse
0: Aufzeichnungsstatus	0 = Angehalten 1 = Warten auf Trigger 2 = Aufzeichnung 3 = Aufzeichnung Nachlaufzeit
1: Datendurchsatz	Wert in kB/s
2: Ist der Server verbunden?	0/False = nein 1/True = ja
5: Aktuelle Puffernutzung	Wert in %
6: Aktuelle Dateipuffernutzung	Wert in %
7: Unverarbeitete Bytes im Dateipuffer	Wert in %

Tab. 7: Informationstypen und mögliche Ergebnisse der Funktion DataStoreInfoHD

## 2.8.5 DongleInfo

`DongleInfo('InfoType*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Hinweis



Diese Funktion wird nur noch aus Gründen der Abwärtskompatibilität unterstützt. Bitte verwenden Sie stattdessen die Funktion *LicenseInfo*, die dem gleichen Zweck dient, dabei aber auch Soft-Lizenzen unterstützt.

### Beschreibung

Diese Funktion gibt Informationen über verschiedene Eigenschaften, die den Dongle betreffen. Diese Informationen können zu Anzeige- und Diagnosezwecken verwendet werden.

Geben Sie den Informationstyp ('InfoType') an, den Sie erhalten wollen. Dieser Informationstyp wird zum Start der Messung ermittelt und ausgegeben. Wenn Sie mehrere Informationen aus dem Dongle erhalten wollen, dann müssen Sie die Funktion mehrmals projektieren, jeweils mit einem anderen Informationstyp.

Folgende Informationstypen werden unterstützt:

Informationstypen	Mögliche Ergebnisse
0: Dongle verfügbar	TRUE = Dongle vorhanden FALSE = Kein Dongle oder Dongle defekt
1: Dongle Zeitlimit in Tagen	Wert der verbleibende Gültigkeitsdauer des Dongles
2: Demo Zeitlimit in Tagen	Wert der verbleibende Gültigkeitsdauer des Dongles bei Teststellungen/Demoverversionen
3: ibaQDR Erfassungszeitlimit in Sekunden	Wert der verbleibenden Zeit, die das iba-QDR-System noch weiterläuft nachdem der Dongle entfernt wurde.
10: Dongle gesteckt-Zähler	Anzahl, wie oft der Dongle gesteckt wurde
11: Dongle entfernt-Zähler	Anzahl, wie oft der Dongle entfernt
12: Dongle gewechselt-Zähler	Anzahl, wie oft Dongles gewechselt wurden

Tab. 8: Informationstypen und mögliche Ergebnisse der Funktion DongleInfo

### Anwendungsbeispiel

Durch Auswertung des Informationstyps 0 (Verfügbarkeit des Dongles) können Sie überwachen, ob ein Dongle vorhanden ist oder nicht bzw. ob der Dongle defekt ist.

Diese Information können Sie dazu verwenden eine E-Mail zu triggern, die den Empfänger darauf hinweist, dass der Dongle nicht mehr verfügbar ist. Ein System, dem im Betrieb der Dongle abhanden gekommen ist, wird nach einer Wartezeit angehalten, was insbesondere bei produktionsrelevanten *ibaQDR*-Systemen unbedingt verhindert werden sollte.

## 2.8.6 FobDLinkStatus

```
FobDLinkStatus('BoardNo*', 'LinkNo*')
```

### Argumente

'BoardNo*'	Kartennummer (0 bis 7)	
'LinkNo*'	Linknummer (dem Signalbaum im I/O-Manager zu entnehmen)	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den Status eines Links einer ibaFOB-D-Karte wieder.

Die Kartennummer (0 bis 7) kann der 7-stelligen Anzeige an der Karte oder der grafischen Darstellung im I/O-Manager entnommen werden. Die Linknummer erhalten Sie über den Signalbaum im I/O-Manager.

### Ergebnisse

0	Link nicht aktiv
1	Link OK
2	Link unterbrochen
3	Link RX ok, aber Flex Ring unterbrochen

### Voraussetzungen

Diese Funktion gibt nur einen Wert zurück, wenn im I/O-Manager auch aktive Eingangsmodule angelegt und konfiguriert sind.

Bei Verwendung eines Busmonitors (z. B. ibaBM-DPM-S) muss im I/O-Manager mindestens ein Aktiver Slave projektiert sein.

## 2.8.7 FobFastLinkStatus

```
FobFastLinkStatus('BoardNo*', 'LinkNo*', 'Filtered*')
```

### Argumente

'BoardNo*'	Kartennummer (0 bis 3)	
'LinkNo*'	Linknummer (dem Signalbaum im I/O-Manager zu entnehmen)	
'Filtered*'	Filter-Einstellung (TRUE oder FALSE)	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den Status eines Links einer ibaFOB-X-Karte im 32-Bit-Modus (32 Mbit/s) wieder.

Die Kartennummer (0 bis 3) kann der 7-stelligen Anzeige an der Karte oder der grafischen Darstellung im I/O-Manager entnommen werden. Die Linknummer erhalten Sie über den Signalbaum im I/O-Manager. 'Filtered' kann entweder TRUE oder FALSE sein. Der gefilterte Verbindungsstatus ignoriert jede Änderung des Link-Status, die schneller ist als 40 ms.

### Ergebnisse

0	Link nicht aktiv
1	Link OK
2	Link unterbrochen

### Voraussetzungen

Diese Funktion gibt nur einen Wert zurück, wenn im I/O-Manager auch aktive Eingangsmodule angelegt und konfiguriert sind.

Bei Verwendung eines Busmonitors (z. B. ibaBM-DPM-S) muss im I/O-Manager mindestens ein Aktiver Slave projiziert sein.

## 2.8.8 FobFlexDeviceStatus

```
FobFlexDeviceStatus('BoardNo*', 'LinkNo*', 'Address*')
```

### Argumente

'BoardNo*'	Kartennummer (0 bis 7)	
'LinkNo*'	Linknummer (dem Signalbaum im I/O-Manager zu entnehmen)	
'Address*'	Adresse des Geräts im Flexring (1 bis 15)	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den Status des Flex-Gerätes mit der Adresse 'Adresse' am Link 'LinkNo' einer ibaFOB-D-Karte 'BoardNo' wieder.

Die Kartennummer (0 bis 7) kann der 7-stelligen Anzeige an der Karte oder der grafischen Darstellung im I/O-Manager entnommen werden. Die Linknummer erhalten Sie über den Signalbaum im I/O-Manager.

Die Adresse des Gerätes wird am Drehschalter des Gerätes eingestellt. Adresswerte 1 bis 15 sind möglich.

### Ergebnisse

0	Gerät nicht konfiguriert
1	Gerät OK
2	Gerät nicht verbunden



## 2.8.9 FobFLinkStatus

```
FobFLinkStatus('BoardNo*', 'LinkNo*')
```

### Argumente

'BoardNo*'	Kartennummer (0 bis 3)	
'LinkNo*'	Linknummer (dem Signalbaum im I/O-Manager zu entnehmen)	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den Status eines Links einer FOB-S oder einer FOB-X-Karte im 3 Mbit-Modus (3,3 Mbit/s) wieder.

Die Kartennummer (0 bis 3) kann der 7-stelligen Anzeige an der Karte oder der grafischen Darstellung im I/O-Manager entnommen werden. Die Linknummer erhalten Sie über den Signalbaum im I/O-Manager.

### Ergebnisse

0	Link nicht aktiv
1	Link OK
2	Link unterbrochen

### Voraussetzungen

Diese Funktion gibt nur einen Wert zurück, wenn im I/O-Manager auch aktive Eingangsmodule angelegt und konfiguriert sind.

Bei Verwendung eines Busmonitors (z. B. ibaBM-DPM-S) muss im I/O-Manager mindestens ein Aktiver Slave projiziert sein.

## 2.8.10 FobMLinkStatus

```
FobMLinkStatus('BoardNo*', 'LinkNo*')
```

### Argumente

'BoardNo*'	Kartennummer (0 bis 3)	
'LinkNo*'	Linknummer (dem Signalbaum im I/O-Manager zu entnehmen)	

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den Status eines Links einer FOB-S-Karte im 5 Mbit-Modus (5 Mbit/s) wieder.

Die Kartennummer (0 bis 3) kann der 7-stelligen Anzeige an der Karte oder der grafischen Darstellung im I/O-Manager entnommen werden. Die Linknummer erhalten Sie über den Signalbaum im I/O-Manager.

### Ergebnisse

0	Link nicht aktiv
1	Link OK
2	Link unterbrochen

### Voraussetzungen

Diese Funktion gibt nur einen Wert zurück, wenn im I/O-Manager auch aktive Eingangsmodule angelegt und konfiguriert sind.

## 2.8.11 FobPlusControlLinkStatus

`FobPlusControlLinkStatus('BoardNo*')`

### Argumente

'BoardNo*'	Kartennummer (0 bis 3)	
------------	------------------------	--

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den Status des Links einer ibaFOB-PlusControl-Karte wieder.

Die Kartennummer (0 bis 3) kann der 7-stelligen Anzeige an der Karte oder der grafischen Darstellung im I/O-Manager entnommen werden. Die Karte besitzt nur einen Link.

### Ergebnisse

0	Link nicht aktiv
1	Link OK
2	Link unterbrochen

### Voraussetzungen

Diese Funktion gibt nur einen Wert zurück, wenn im I/O-Manager auch aktive Eingangsmodule angelegt und konfiguriert sind.

## 2.8.12 FobSDLinkStatus, FobSDexpLinkStatus

```
FobSDLinkStatus('BoardNo*')
```

```
FobSDexpLinkStatus('BoardNo*')
```

### Argumente

'BoardNo*'	Kartennummer (0 bis 3)	
------------	------------------------	--

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den Status des Links einer ibaFOB-SD-Karte (PCI) bzw. ibaFOB-SDexp-Karte (PCIe) wieder.

Die Kartennummer (0 bis 3) kann der 7-stelligen Anzeige an der Karte oder der grafischen Darstellung im I/O-Manager entnommen werden. Die Karte besitzt nur einen Link.

### Ergebnisse

0	Link nicht aktiv
1	Link OK
2	Link unterbrochen

### Voraussetzungen

Diese Funktion gibt nur einen Wert zurück, wenn im I/O-Manager auch aktive Eingangsmodule angelegt und konfiguriert sind.

## 2.8.13 FobTDCLinkStatus, FobTDCexpLinkStatus

```
FobTDCLinkStatus('BoardNo*')
```

```
FobTDCexpLinkStatus('BoardNo*')
```

### Argumente

'BoardNo*'	Kartennummer (0 bis 3)	
------------	------------------------	--

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt den Status des Links einer ibaFOB-TDC-Karte (PCI) bzw. ibaFOB-TDCexp-Karte (PCIe) wieder.

Die Kartennummer (0 bis 3) kann der 7-stelligen Anzeige an der Karte oder der grafischen Darstellung im I/O-Manager entnommen werden. Die Karte besitzt nur einen Link.

### Ergebnisse

0	Link nicht aktiv
1	Link OK
2	Link unterbrochen

## Voraussetzungen

Diese Funktion gibt nur einen Wert zurück, wenn im I/O-Manager auch aktive Eingangsmodule angelegt und konfiguriert sind.

### 2.8.14 ICPSensorStatus

```
ICPSensorStatus('ModuleNo*', 'SensorNo*')
```

#### Argumente

'Module-No*'	Modulnummer eines Padu-8-ICP-Moduls
'SensorNo*'	Sensor-Nummer (zwischen 0 und 7)

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Mit dieser Funktion wird der Status eines ICP-Sensors überwacht. Der erste Parameter 'Module-No' gibt die Modulnummer eines ibaPADU-8-ICP oder ibaMS8xICP-/IEPE-Moduls im Signalbaum an. Der zweite Parameter 'SensorNo' legt fest, welcher Sensor überwacht werden soll. Die Sensor-Nummer liegt zwischen 0 und 7.

#### Ergebnisse

0	Sensor OK
1	offene Schleife erkannt

### 2.8.15 InterruptCycleTime

```
InterruptCycleTime('Type=0*')
```

#### Argumente

'Type*'	Optional Parameter (Voreinstellung = 0), mit dem die Art des Ergebniswertes bestimmt werden kann.	
	'Type' = 0	Der Istwert wird ausgegeben.
	'Type' = 1	Das Minimum (kleinste gemessene Zykluszeit) wird ausgegeben.
	'Type' = 2	Das Maximum (größte gemessene Zykluszeit) wird ausgegeben.

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Diese Funktion liefert die Interrupt-Zykluszeit in Mikrosekunden zurück. Der Wert wird nicht bei jedem Interrupt aktualisiert, sondern mit einem geringeren Zyklus.

## 2.8.16 InterruptTime

```
InterruptTime('Type=0*')
```

### Argumente

'Type*'	Optionaler Parameter (Voreinstellung = 0), mit dem die Art des Ergebniswertes bestimmt werden kann.	
	'Type' = 0	Der Istwert wird ausgegeben.
	'Type' = 1	Das Minimum (kleinste gemessene Dauer) wird ausgegeben.
	'Type' = 2	Das Maximum (größte gemessene Dauer) wird ausgegeben.

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion liefert die Interrupt-Zeit (Interrupt-Dauer) in Mikrosekunden zurück. Der Wert wird nicht bei jedem Interrupt aktualisiert, sondern mit einem geringeren Zyklus.

## 2.8.17 LicenseInfo

`LicenseInfo('InfoType*')`

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion gibt Informationen über verschiedene Eigenschaften, die den Lizenzcontainer (Dongle oder Soft-Lizenz) betreffen. Diese Informationen können zu Anzeige- und Diagnosezwecken verwendet werden.

Geben Sie den Informationstyp ('InfoType') an, den Sie erhalten wollen. Dieser Informationstyp wird zum Start der Erfassung ermittelt und ausgegeben. Wenn Sie mehrere Informationen aus dem Lizenzcontainer erhalten wollen, dann müssen Sie die Funktion mehrmals projektieren, jeweils mit einem anderen Informationstyp.

Folgende Informationstypen werden unterstützt:

Informationstypen	Mögliche Ergebnisse
0: Lizenzcontainer verfügbar	TRUE = Lizenzcontainer vorhanden FALSE = Lizenzcontainer fehlt oder ist defekt
1: Lizenz-Zeitlimit in Tagen	Wert der verbleibende Gültigkeitsdauer der Lizenz
2: Demo-Zeitlimit in Tagen	Wert der verbleibende Gültigkeitsdauer der Lizenz bei Teststellungen/Demoversionen
3: Zeitlimit in Sekunden, bevor die Erfassung aufgrund einer Lizenzentfernung gestoppt wird	Wert der verbleibenden Zeit, die das System noch weiterläuft nachdem der Lizenzcontainer entfernt wurde.
10: Zähler für eingefügte Lizenzcontainer	Anzahl, wie oft der Lizenzcontainer gesteckt wurde
11: Lizenzcontainer entfernt-Zähler	Anzahl, wie oft der Lizenzcontainer entfernt wurde
12: Zähler für geänderte Lizenzcontainer	Anzahl, wie oft Lizenzcontainer gewechselt wurden

Tab. 9: Informationstypen und mögliche Ergebnisse der Funktion LicenseInfo

### Anwendungsbeispiel

Durch Auswertung des Informationstyps 0 (Verfügbarkeit des Lizenzcontainers) können Sie überwachen, ob ein Lizenzcontainer vorhanden ist oder nicht bzw. ob der Lizenzcontainer defekt ist.

Diese Information können Sie dazu verwenden eine E-Mail zu triggern, die den Empfänger darauf hinweist, dass der Lizenzcontainer nicht mehr verfügbar ist. Ein System, dem im Betrieb der Lizenzcontainer abhanden gekommen ist, wird nach einer Wartezeit angehalten, was insbesondere bei produktionsrelevanten *ibaQDR*-Systemen unbedingt verhindert werden sollte.

## 2.8.18 MultiStationStatus

```
MultiStationStatus()
```

### Beschreibung

gibt den aktuellen Multistation-Modus zurück.

### Ergebnisse

0	Standalone
1	Slave
2	Master

## 2.8.19 PerformanceCounter

```
PerformanceCounter('Category*', 'CounterName*', 'InstanceName*')
```

### Argumente

'Category*'	Texteintrag aus der Spalte "Objekt" der Windows Leistungsübersicht
'CounterName*'	Texteintrag aus der Spalte "Leistungsindikator" der Windows Leistungsübersicht
'InstanceName*'	Texteintrag aus der Spalte "Instanz" der Windows Leistungsübersicht

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Die Funktion PerformanceCounter kann bestimmte Leistungsmerkmale des Computers auslesen, anzeigen und den Verlauf aufzeichnen. Sie gibt den Wert des Leistungszählers zurück oder 0, wenn der Leistungszähler nicht existiert. Mit der PerformanceCounter-Funktion kann jeweils ein Leistungsindikator ausgelesen werden. Wenn Sie mehrere Leistungsindikatoren anzeigen wollen, dann müssen Sie die Funktion entsprechend oft konfigurieren.

### Beispiel

Überwachung der Systemleistung

### Lösung

Um die Leistungsmerkmale, die überwacht werden sollen, unter Windows 7 auszuwählen, drücken Sie die Windows-Taste und suchen Sie nach Computerverwaltung. Anschließend müssen sie auf der linken Seite unter *System – Leistung – Überwachungstools* den Punkt *Leistungsüberwachung* auswählen. Mit dem grünen +-Symbol in der oberen Leiste können Sie zu überwachende Leistungsmerkmale hinzufügen.

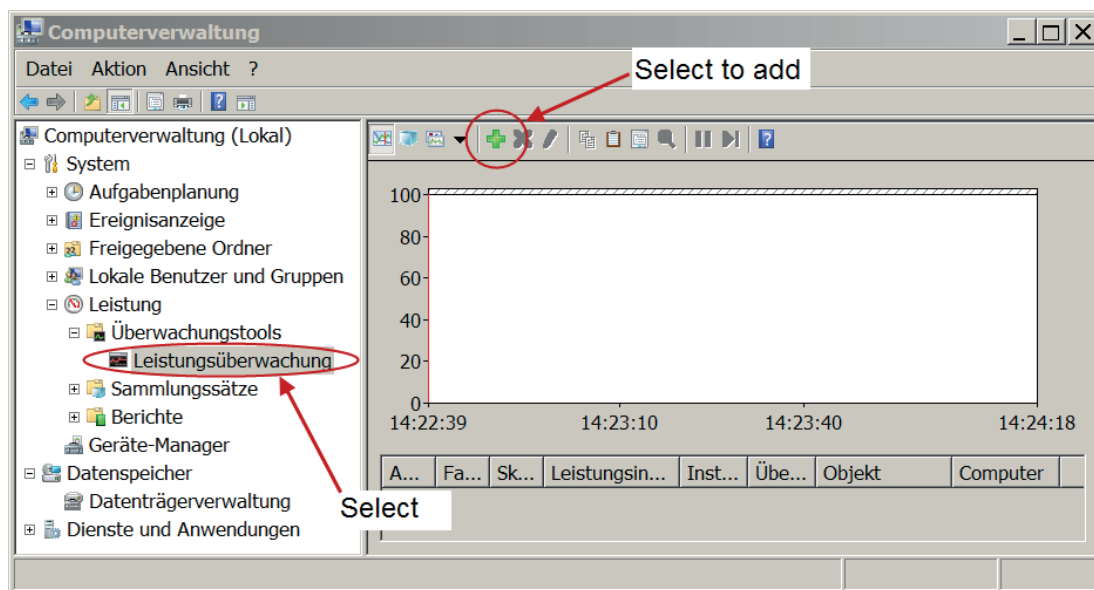


Abb. 2: Neue zu überwachende Leistungsmerkmale anzeigen

Wählen Sie die Leistungsindikatoren und anschließend die Instanzen aus. In diesem Fall wird Prozess, ibaPDA und Alle Instanzen ausgewählt, anschließend mit <OK> bestätigt. In der folgenden Abbildung sind die Schritte markiert und nummeriert.

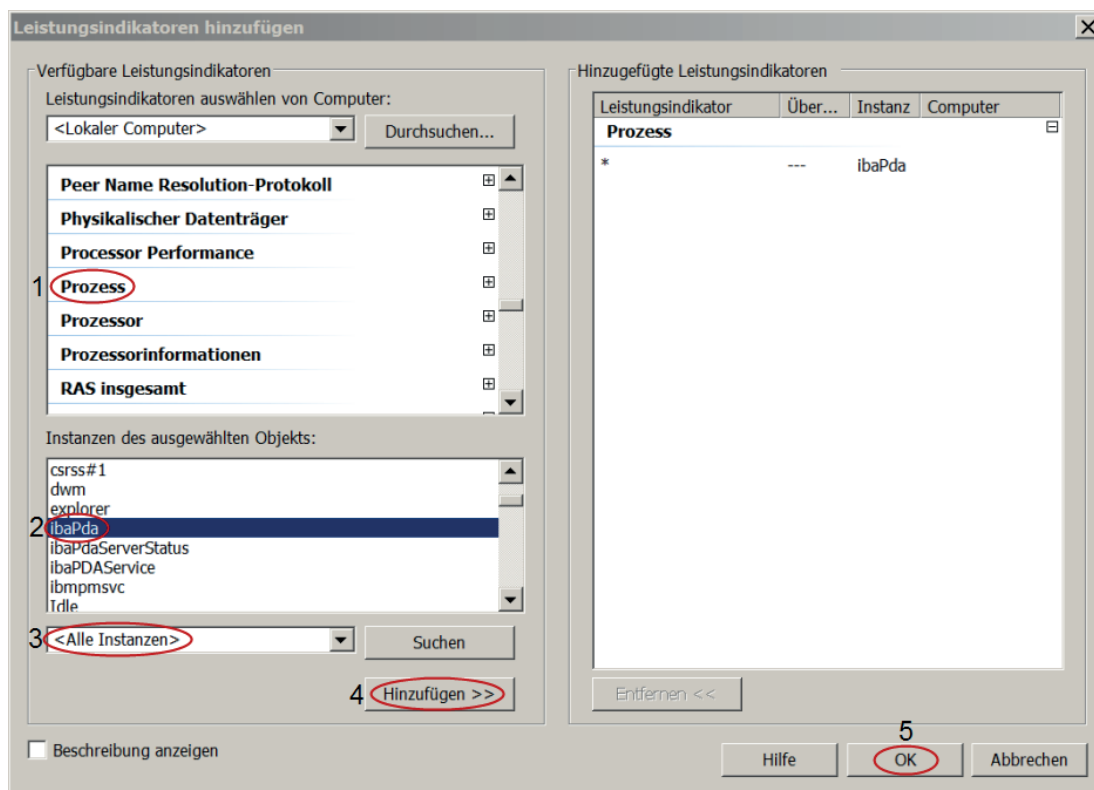


Abb. 3: Auswahl der Funktionsargumente

Die Zuordnung zu den einzelnen Funktionsargumenten von PerformanceCounter ist in der nächsten Abbildung verdeutlicht. Sie müssen für die Eingabe in ibaPDA die Bezeichnungen der einzelnen Leistungsindikatoren, Objekte und Instanzen exakt übernehmen.



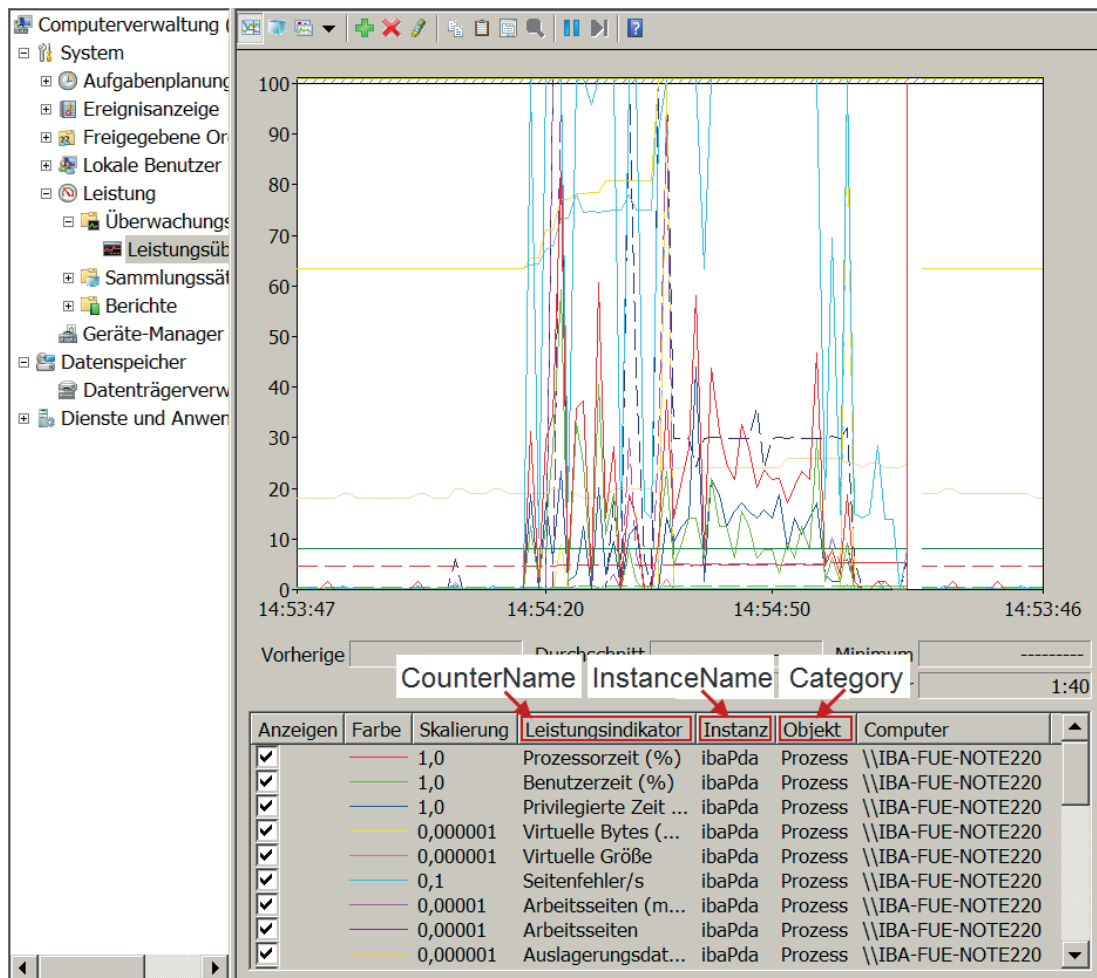
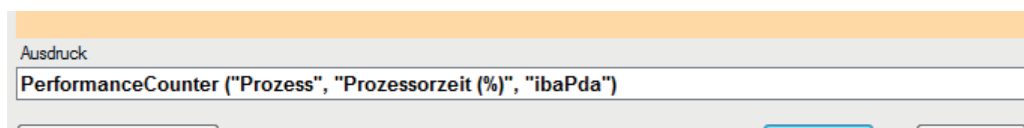


Abb. 4: Bezeichnung der Funktionsargumente

Beispiel für die Eingabe in ibaPDA zur Ermittlung der Prozessorzeit:



## 2.8.20 Ping

```
Ping('Address', 'Trigger', 'Timeout=5', 'Size=32')
```

### Argumente

'Address'	IP-Adresse des Zielrechners
'Trigger'	Triggersignal (steigende Flanke), mit dem Ping gesendet werden soll
'Timeout'	Optionaler Parameter (Voreinstellung =5) zur Vorgabe der Wartezeit auf Antwort, bevor "keine Antwort" als Ergebnis ausgegeben wird.
'Size'	Optionaler Parameter: Größe der Ping-Anforderung in Bytes (Voreinstellung=32)

### Beschreibung

Diese Funktion sendet eine Ping-Anforderung an 'Address' bei einer steigenden Flanke von 'Trigger'. Die Funktion liefert die Zeit in Millisekunden, die gebraucht wird, bis die Ping-Antwort empfangen wurde. Wenn innerhalb von 'Timeout' Sekunden keine Antwort erfolgt, dann wird -1 ausgegeben. Das Ergebnis ist 0, solange keine Ping-Anforderung gesendet wurde. 'Size' bestimmt die Größe der Ping-Anforderung in Bytes.

Als Ergebnis gibt die Funktion einen Analogwert aus.

### Ergebnisse

-1	keine Antwort auf Ping-Anforderung innerhalb von 'Timeout'
0	keine Ping-Anforderung gesendet
n	Antwortzeit in ms

## 2.8.21 TimeSinceLastSync

```
TimeSinceLastSync()
```

### Beschreibung

Diese Funktion gibt die seit der letzten Zeitsynchronisation verstrichene Zeit in Sekunden an. Ist keine Zeitsynchronisation erfolgt, gibt die Funktion -1 zurück.

### Tipp



Die Zeitsynchronisation wird in ibaPDA im Menüpunkt Konfiguration und anschließend iba I/O-Manager unter dem Punkt Allgemein eingestellt. Mögliche Quellen für die Zeitsynchronisation sind DCF77, IEC1131 und DGM oder aber es ist keine Zeitsynchronisation eingestellt.

## 2.8.22 TimeSyncStatus

TimeSyncStatus('Source\*')

### Argumente

'Source*'	Quelle für Zeitsynchronisation	
	'Source' = -1	zuletzt verwendete Quelle für die Zeitsynchronisation
	'Source' = 0	DCF77 Quelle 1
	'Source' = 1	DCF77 Quelle 2
	'Source' = 2	IEC1131
	'Source' = 3	DGM200P
	'Source' = 4	PTP Slave
	'Source' = 5	ibaClock

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Diese Funktion liefert den Status der ausgewählten Quelle für die Zeitsynchronisation.

'Source' kann folgende Werte erhalten:

### Ergebnisse

0	Quelle ist inaktiv
1	Quelle ist aktiv und gültig
2	Quelle ist aktiv aber ungültig

### Tipp



Die Zeitsynchronisation wird in *ibaPDA* im Menüpunkt Konfiguration und anschließend iba I/O-Manager unter dem Punkt Allgemein eingestellt.

## 2.9 Filter-Funktionen

### 2.9.1 BP

```
BP('Expression', 'Frequency1*', 'Frequency2*')
```

#### Argumente

'Expression'	Messwert
'Frequency1*'	Angabe der unteren Grenze des Frequenzbandes für das Filter
'Frequency2*'	Angabe der oberen Grenze des Frequenzbandes für das Filter

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Die Funktion ist ein Bandpassfilter mit einem Frequenzband zwischen den Frequenzen 'Frequency1' und 'Frequency2'. Das Filter ist ein Butterworth-Filter zweiter Ordnung.

### 2.9.2 HP

```
HP('Expression', 'Frequency*')
```

#### Argumente

'Expression'	Messwert
'Frequency*'	Angabe der Grenzfrequenz des Filters

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Die Funktion ist ein Hochpassfilter mit der Grenzfrequenz 'Frequency'. Das Filter ist ein Butterworth-Filter zweiter Ordnung.

### 2.9.3 LP

`LP('Expression', 'Omega*', 'Reset=0')`

#### Argumente

'Expression'	Messwert	
'Omega*'	Angabe der Kreisfrequenz des Filters, daraus ergibt sich die Frequenz der Funktion LP zu Frequenz = $\text{'Omega'}/(2 \cdot \pi)$	
'Reset'	Optional digitaler Parameter, der zum Deaktivieren der Funktion verwendet werden kann. 'Reset' kann selbst auch ein Ausdruck sein.	
	'Reset' > 0	Filter wird deaktiviert, Eingangssignal wird ungefiltert ausgegeben
	'Reset' = 0	Filter wird angewendet

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

#### Beschreibung

Die Funktion ist ein Lowpassfilter mit Frequenz =  $\text{'Omega'}/(2 \cdot \pi)$  auf 'Expression'. Das Filter ist ein einpoliges Filter mit einer Roll-Off-Rate von 20 dB/Dekade. Wenn der optionale Parameter 'Reset' TRUE ist, dann wird das Filter deaktiviert und das Resultat ist das ungefilterte Eingangssignal 'Expression'. 'Reset' kann auch als Ausdruck formuliert werden.

Beispiele:

<code>LP([0:0],10)</code>	Filter ist aktiv ('Reset' weggelassen).
<code>LP([0:0],10,If([0:0]&lt;0,1,0))</code>	Die Filterfunktion wird nur für positive Werte angewendet.
<code>LP([0:0],10,[3.1])</code>	z. B. mit <code>[3.1] = If([0:0]&gt;10, 1, 0)</code> Der Filter wird deaktiviert, sobald der Ausdruck <code>[3.1]</code> TRUE zurückgibt, also wenn der Ausdruck <code>[0:0]</code> den Grenzwert 10 überschreitet.

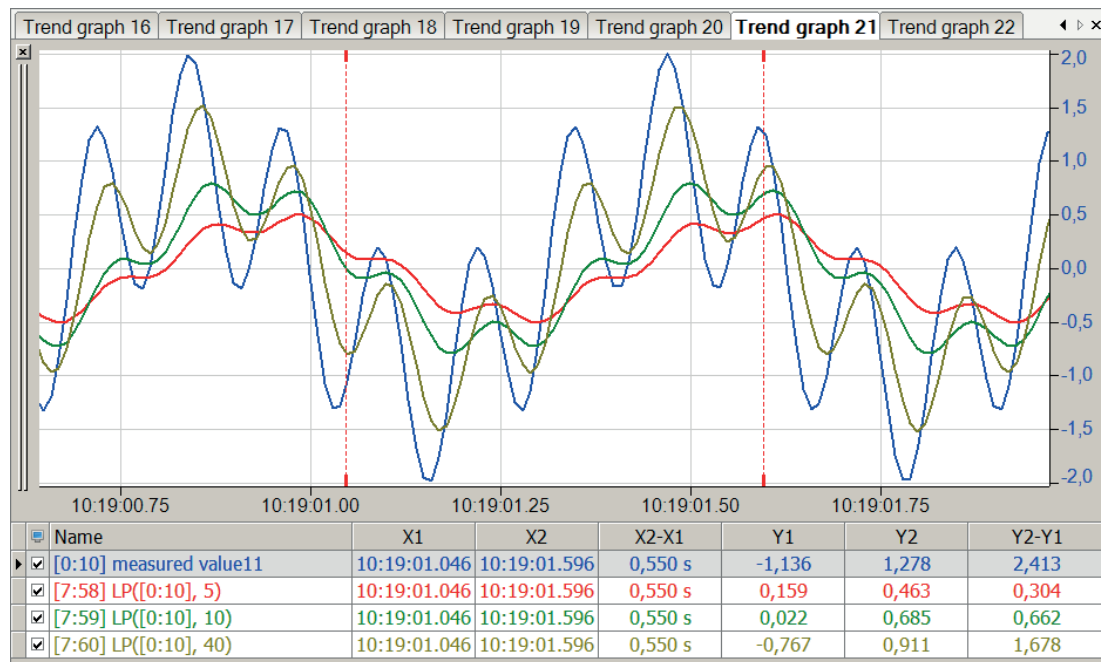
#### Beispiel

Anwendung der Filterfunktion auf eine Sinusschwingung mit 10 Hz, die von einer weiteren Schwingung mit 50 Hz überlagert wird.

#### Aufgabenstellung

Die Filterfunktion soll mit den Werten 5, 10 und 40 für 'Omega' angewendet werden.

## Lösung



Blau	Ursprüngliches Signal einer überlagerten Sinusschwingung	Rot	Gefiltertes Signal mit 'Omega'=5
Grün	Gefiltertes Signal mit 'Omega'=10	Gelb	Gefiltertes Signal mit 'Omega'=10

Je niedriger der Wert für 'Omega' gewählt wird, umso stärker wird das Signal gedämpft.

## 2.9.4 EnvelopeSpectral

`EnvelopeSpectral('Expression', 'Frequency1*', 'Frequency2*', 'Cutoff frequency')`

### Argumente

'Expression'	Messwert
'Frequency1*'	Angabe der unteren Grenze des Frequenzbandes für die Hüllkurvenberechnung Wert darf nicht kleiner sein als 10% der Nyquist-Frequenz.
'Frequency2*'	Angabe der oberen Grenze des Frequenzbandes für die Hüllkurvenberechnung Wert darf nicht größer sein als 90% der Nyquist-Frequenz
'Cutoff frequency'	Optionale Angabe einer Grenzfrequenz zur Konfiguration eines Tiefpassfilters, der auf die Hüllkurve angewendet wird.

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Die Funktion berechnet die spektrale Hüllkurve von 'Expression' mit einem konstanten Bandpass zwischen den Frequenzen 'Frequency1' und 'Frequency2'. Die kleinste Bandbreite beträgt 10% der Nyquist-Frequenz, also 10% der halben Abtastfrequenz.

Das Signal 'Expression' wird mit der Zeitbasis des virtuellen Signals neu abgetastet. Es wird kein Anti-Aliasing-Filter angewendet.

## 2.9.5 Preprocess

```
Preprocess('Expression', "'Preprocess profile name*')
```

### Argumente

'Expression'	Ausdruck, der vorverarbeitet werden soll
'Preprocess profile name**'	Name des Vorverarbeitungsprofils bzw. Vorprozessors

Parameter, die mit \* enden, werden nur einmalig zu Beginn der Erfassung übernommen.

### Beschreibung

Das Vorverarbeitungsprofil mit Namen 'Preprocess profile name' wird auf 'Expression' angewendet.

Vorverarbeitungsprofile können im Vorprozessor-Manager konfiguriert werden. Sie erreichen diesen Manager über den Ausdruckseditor eines Inspectra Expert-Moduls.

## 2.10 Remanenz-Funktionen

Die Funktionen in dieser Gruppe können in einem so genannten Remanenzmodul verwendet werden. Funktionen mit Remanenzcharakter behalten stets ihren letzten Ergebniswert bis sie gezielt zurückgesetzt werden.

Bei Verwendung dieser Funktionen in einem Remanenzmodul kann jeweils der zuletzt berechnete Ergebniswert über das Anhalten der Messung hinaus gespeichert werden. Beim Neustart der Messung kann dieser Wert wieder als Initialwert geladen werden.

Somit gehen Langzeitdaten wie z. B. Betriebsstundenzähler und Medienverbräuche durch das Stoppen und Starten der Messung nicht verloren.

Funktionen mit dieser Eigenschaft sind:

➤ *Count*, Seite 103

➤ *Int*, Seite 37

➤ *Max*, Seite 57

➤ *Min*, Seite 63



## 2.11 Plugins

Das System ibaPDA Plugin wurde entwickelt, um ibaPDA-Nutzern die Möglichkeit zu bieten, eigene Funktionen zu erstellen, um mit ibaPDA benutzerdefinierte Berechnungen von Messdaten durchzuführen. ibaPDA führt diese Funktionen in Echtzeit aus. Diese Funktionen können genau wie integrierte Funktionen in Ausdrücken für virtuelle Signale verwendet werden. Sie erscheinen ebenso im Ausdruckseditor (Plugins).

Um benutzerdefinierte Funktionen zu erzeugen, müssen Sie eine NET.dll Datei erstellen. Diese .dll kann in jeder .NET-Sprache geschrieben werden (C#, C++, VB.NET,...).

---

### Andere Dokumentation



Für eine detaillierte Beschreibung des Plugin-Programmierens und der Konfiguration lesen Sie bitte gezielt das Handbuch "ibaPDA Plugin".

---

## 3 Support und Kontakt

### Support

Tel.: +49 911 97282-14  
Fax: +49 911 97282-33  
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  
Fax: +49 911 97282-33  
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](http://www.iba-ag.com)**