imc FAMOS

Funktionsreferenz

Version 2024

imc Test & Measurement GmbH • Voltastraße 5 • 13355 Berlin • Germany

Übersicht

Sie finden hier eine alphabetisch sortierte Übersicht über die in imc FAMOS enthaltenen Funktionen, Sequenzbefehle, Konstanten und Operatoren.

-

Punktweise Subtraktion

Deklaration:

```
Minuend - Subtrahend -> Differenz
```

Parameter:

Minuend	Erster Parameter, Minuend
Subtrahend	Zweiter Parameter, Subtrahend
Differenz	Differenz, Ergebnis der punktweisen Subtraktion.

Beschreibung:

Es wird die Differenz von zwei Variablen gebildet. Datensätze werden punktweise subtrahiert unabhängig von den Maßstäben ihrer x-Achsen. Bei der Subtraktion von einem Einzelwert und einem Datensatz wird der Einzelwert auf jeden Punkt des Datensatzes angewendet.

Für eine zeit- bzw. x-richtige Subtraktion können Sie die Funktion Sub() verwenden.

Anmerkungen

- Unter bestimmten Voraussetzungen können auch strukturierte Datensätze (Events/Segmente) verrechnet werden. Dazu muss entweder einer der beiden Parameter ein Einzelwert sein, oder beide Parameter müssen exakt die gleiche Struktur besitzen (d.h. Gesamtlänge, Segmentlänge, Eventzahl und -Längen müssen gleich sein).
- Für eine sinnvolle Berechnung müssen die x-Skalierungen von beiden beteiligten Datensätzen gleich sein. Sind sie nicht gleich, wird eine Warnung generiert. Es werden dann die Angaben zur ersten Variable benutzt.
- Bei der Subtraktion sind nicht alle Kombinationen mit komplexen Datentypen möglich. Wenn Sie einen Datensatz vom Typ DP subtrahieren möchten, müssen Sie ihn zuerst in einen anderen Typ wandeln, zweckmäßigerweise mit der Funktion idB(). Wenn Sie einen komplexen und einen reellen Datensatz subtrahieren möchten, müssen Sie vorerst die reellen Daten zu komplexen machen, zweckmäßigerweise mit der Funktion Compl().
- Komplexe Datensätze werden nach den Regeln der komplexen Rechnung punktweise subtrahiert. Beachten Sie, dass bei der Subtraktion von komplexen Zahlen in Polarkoordinaten-Darstellung nicht die Beträge und Phasen einzeln subtrahiert werden.
- Haben die übergebenen Datensätze unterschiedliche Längen, entspricht die Länge des Ergebnisses der des kürzeren Datensatzes.

Beispiele:

Offset-Korrektur, Subtraktion einer festen Zahl:

```
NDcorr = NDdata - offset
XYdata.Y = XYData.Y- offset
```

Unterschied zwischen zwei Spektren:

```
MPdiff = MPspec1 - MPspec2
```

Unterschied zwischen zwei Spektren, wobei das eine in dB vorliegt.:

```
MPdiff = MPspec1 - idB(DPspec2)
```

Zwei Möglichkeiten einer Subtraktion eines reellen Datensatzes von einem komplexen Datensatz RIspec mit den Komponenten RIspec.R und RIspec.I:

```
RIdiff = Compl(RIspec.R - NDreal, RIspec.I)
RIdiff = RIspec - NDreal
```

Zwei Möglichkeiten einer Subtraktion eines Einzelwertes von einem komplexen Datensatz RIspec mit den Komponenten RIspec.R und RIspec.I:

```
RIdiff = Compl(RIspec.R - offset, RIspec.I)
RIdiff = RIspec - offset
```

Siehe auch:

+(Addition), Sub, Append

>

Vergleichsoperator, "größer als"

Deklaration:

```
Operand1 > Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Operand2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

Vergleich zweier Zahlen. Das Ergebnis ist 1, wenn der erste Operand größer als der zweite Operand ist. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte und auf Datensätze angewendet werden. Bei Datensätzen erfolgt ein punktweiser Vergleich.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Das Maximum eines Datensatzes wird bestimmt und auf Überschreiten eines Grenzwertes getestet.

```
Channel1 = ...
Maximum = max(Channel1)
IF Maximum > 21 AND Maximum <= 34
   BoxMessage("Achtung", "Maximum nahe Grenzwert!", "!1")
ELSE
   IF Maximum > 34
      BoxMessage("Achtung", "Grenzwert ist überschritten", "!1")
   END
END
```

In einem Datensatz werden alle Werte oberhalb einer bestimmten Schwelle auf 0 gesetzt.

```
Channel1 = ...
Result = Channel1 * (Channel1 > 20)
```

In einem Datensatz werden alle Werte unterhalb 15 auf den festen Wert 10 und alle Werte größer als 15 auf den festen Wert 20 gesetzt.

```
Channel1 = ...
Result = (Channel1 <= 15) * 10  + (Channel1 > 15) * 20
```

Siehe auch:

<, >=, UpperValue

>=

Vergleichsoperator, "größer oder gleich"

Deklaration:

```
Operand1 >= Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Operand2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

Vergleich zweier Zahlen. Das Ergebnis ist 1, wenn der erste Operand größer oder gleich dem zweiten Operanden ist. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte und auf Datensätze angewendet werden. Bei Datensätzen erfolgt ein punktweiser Vergleich.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Das Maximum eines Datensatzes wird bestimmt und auf Überschreiten eines Grenzwertes getestet.

```
Channel1 = ...
Maximum = max(Channel1)
IF Maximum > 21 AND Maximum < 34
   BoxMessage("Achtung", "Maximum nahe Grenzwert!", "!1")
ELSE
   IF Maximum >= 34
      BoxMessage("Achtung", "Grenzwert ist überschritten", "!1")
   END
END
```

In einem Datensatz werden alle Werte oberhalb einer bestimmten Schwelle auf 0 gesetzt.

```
Channel1 = ...
Result = Channel1 * (Channel1 >= 20)
```

In einem Datensatz werden alle Werte unterhalb 15 auf den festen Wert 10 und alle Werte größer als 15 auf den festen Wert 20 gesetzt.

```
Channel1 = ...
Result = (Channel1 < 15) * 10  + (Channel1 >= 15) * 20
```

Siehe auch:

<=, >, UpperValue

<

Vergleichsoperator, "kleiner als"

Deklaration:

```
Operand1 < Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Operand2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

Vergleich zweier Zahlen. Das Ergebnis ist 1, wenn der erste Operand kleiner als der zweite Operand ist. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte und auf Datensätze angewendet werden. Bei Datensätzen erfolgt ein punktweiser Vergleich.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Das Maximum eines Datensatzes wird bestimmt und auf Überschreiten eines Grenzwertes getestet.

```
Channel1 = ...
Maximum = max(Channel1)
IF Maximum >= 21 AND Maximum < 34
   BoxMessage("Achtung", "Maximum nahe Grenzwert!", "!1")
ELSE
   IF Maximum >= 34
      BoxMessage("Achtung", "Grenzwert ist überschritten", "!1")
   END
END
```

In einem Datensatz werden alle Werte unterhalb einer bestimmten Schwelle auf 0 gesetzt.

```
Channel1 = ...
Result = Channel1 * (Channel1 < 20)
```

In einem Datensatz werden alle Werte unterhalb 15 auf den festen Wert 10 und alle Werte größer als 15 auf den festen Wert 20 gesetzt.

```
Channel1 = ...
Result = (Channel1 < 15) * 10  + (Channel1 >= 15) * 20
```

Siehe auch:

<=, >, LowerValue

<>

Vergleichsoperator, "ungleich"

Deklaration:

```
Operand1 <> Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert/Datensatz/Text/Textfeld.
Operand2	Zweiter zu vergleichender Einzelwert/Datensatz/Text/Textfeld.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

Test auf Ungleichheit. Das Ergebnis ist 1, wenn beide Operanden nicht gleich sind. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte, Datensätze, Texte und Textfelder angewendet werden.

Bei Datensätzen erfolgt ein punktweiser Vergleich.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Zwei Textfelder gelten als identisch, wenn sie die selbe Dimension besitzen und die Texte mit selben Index jeweils gleich sind.

Beim Vergleich von Texten erfolgt keine Unterscheidung zwischen Groß- und Kleinschreibung.

Beispiele:

Es wird gepüft, ob ein Datensatz eine gerade Anzahl von Samples enthält.

```
Channel1 = ...
len = Leng?(Channel1)
IF mod(len, 2) <> 0
   BoxMessage("Fehler", "Ungültige Datensatzlänge", "!1")
END
```

Zwei digitale Datensätze werden verglichen. Der Ergebnisdatensatz ist überall dort 1, wo die Operandendatensätze verschieden sind.

```
Result = (DigChannel1 <> DigChannel2)
```

Die Einheit eines Datensatzes wird auf Gültigkeit geprüft.

```
unit = UNIT?(Pressure_1, 1)
IF (unit <> "Bar" AND unit <> "Pa")
   BoxMessage("Achtung", "Ungültige Einheit (Druck)", "!1")
END
```

Siehe auch:

=, <=, >, TComp

<=

Vergleichsoperator, "kleiner oder gleich"

Deklaration:

```
Operand1 <= Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Operand2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

Vergleich zweier Zahlen. Das Ergebnis ist 1, wenn der erste Operand kleiner oder gleich dem zweiten Operanden ist. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte und auf Datensätze angewendet werden. Bei Datensätzen erfolgt ein punktweiser Vergleich.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Das Maximum eines Datensatzes wird bestimmt und auf Überschreiten eines Grenzwertes getestet.

```
Channel1 = ...
Maximum = max(Channel1)
IF Maximum > 21 AND Maximum <= 34
   BoxMessage("Achtung", "Maximum nahe Grenzwert!", "!1")
ELSE
   IF Maximum > 34
      BoxMessage("Achtung", "Grenzwert ist überschritten", "!1")
   END
END
```

In einem Datensatz werden alle Werte unterhalb einer bestimmten Schwelle auf 0 gesetzt.

```
Channel1 = ...
Result = Channel1 * (Channel1 <= 20)
```

In einem Datensatz werden alle Werte unterhalb 15 auf den festen Wert 10 und alle Werte größer als 15 auf den festen Wert 20 gesetzt.

```
Channel1 = ...
Result = (Channel1 <= 15) * 10  + (Channel1 > 15) * 20
```

Siehe auch:

<, >=, LowerValue

*

Punktweise Multiplikation

Deklaration:

```
Faktor1 * Faktor2 -> Produkt
```

Parameter:

Faktor1	Erster Parameter, Faktor
Faktor2	Zweiter Parameter, Faktor
Produkt	Produkt, Ergebnis der punktweisen Multiplikation.

Beschreibung:

Es wird das Produkt von zwei Variablen gebildet. Datensätze werden punktweise multipliziert unabhängig von den Maßstäben ihrer x-Achsen.

Für eine zeit- bzw. x-richtige Multiplikation können Sie die Funktion Mult() verwenden.

Bei der Multiplikation von Einzelwerten und Datensätzen wird der Einzelwert bei jedem Wert des Datensatzes gleichermaßen benutzt.

Alle komplexen Datentypen sind bei der Multiplikation austauschbar. Das Ergebnis ist also vom Inhalt her immer gleich, egal in welchem Typ die Parameter vorliegen. Sobald an der Multiplikation mindestens ein komplexer Datensatz beteiligt ist, werden alle Einheiten auf dB überprüft und die Daten entsprechend behandelt. Die Multiplikation von Daten in dB wird ausgeführt, indem dB-Zahlen addiert werden.

Anmerkungen

- Unter bestimmten Voraussetzungen können auch strukturierte Datensätze (Events/Segmente) verrechnet werden. Dazu muss entweder einer der beiden Parameter ein Einzelwert sein, oder beide Parameter müssen exakt die gleiche Struktur besitzen (d.h. Gesamtlänge, Segmentlänge, Eventzahl und -Längen müssen gleich sein).
- Für eine sinnvolle Berechnung müssen die x-Skalierungen von beiden beteiligten Datensätzen gleich sein. Sind sie nicht gleich, wird eine Warnung generiert. Es werden dann die Angaben zur ersten Variable benutzt.
- Komplexe Datensätze werden nach den Regeln der komplexen Rechnung punktweise multipliziert. Beachten Sie, dass bei der Multiplikation von komplexen Zahlen in kartesischer Darstellung nicht Real- und Imaginärteil einzeln multipliziert werden. Beachten Sie weiter, dass bei Darstellungen in Polarkoordinaten Phasen und dB-Zahlen addiert werden.
- Haben die übergebenen Datensätze unterschiedliche Längen, entspricht die Länge des Ergebnisses der des kürzeren Datensatzes.

Beispiele:

Ein Datensatz wird mit einem Bezugswert normiert:

```
NDnorm = NDdaten * Ewbezug
XYdaten.Y = XYddaten.Y * EwBezug
```

Das Spektrum der Ausgangsgröße eines Systems wird berechnet, indem die Übertragungsfunktion mit dem Spektrum der Eingangsgröße multipliziert wird:

```
BPspek2 = BPspek1 * RIübertr
```

Das Spektrum der Ausgangsgröße eines Systems wird berechnet, indem die Übertragungsfunktion (in dB) mit dem Spektrum der Eingangsgröße multipliziert wird.

```
BPspek2 = BPspek1 * DPübertr
```

Normierung eines Spektrums mit einem Bezugswert. Ist die Einheit des Bezugswertes dB, wird er als ein Wert in Dezibel interpretiert und entsprechend verrechnet:

```
BPübertr = BPspek * EWbezug
```

Multiplikation von zwei reellen Datensätzen, wobei einer in dB vorliegt. Da kein komplexer Datensatz beteiligt ist, werden die dB nicht automatisch erkannt. Deshalb ist auf den Datensatz, der in dB vorliegt, die Funktion zur Berechnung der inversen dB anzuwenden.

```
NDdat = idB(NDdb) * NDkeindb
; oder alternativ:
NDdatdb = NDdb + dB(NDkeindb)
```

Siehe auch:

/(Division), Mult, MatrixMult

/

Punktweise Division

Deklaration:

```
Dividend / Divisor -> Quotient
```

Parameter:

Dividend	Erster Parameter, Dividend
Divisor	Zweiter Parameter, Divisor
Quotient	Quotient, Ergebnis der punktweisen Division.

Beschreibung:

Es wird der Quotient von zwei Variablen gebildet. Datensätze werden punktweise dividiert unabhängig von den Maßstäben ihrer x-Achsen.

Für eine zeit- bzw. x-richtige Division können Sie die Funktion Div() verwenden.

Bei der Division von Einzelwerten und Datensätzen wird der Einzelwert bei jedem Wert des Datensatzes gleichermaßen benutzt.

Alle komplexen Datentypen sind bei der Division austauschbar. Das Ergebnis ist also vom Inhalt her immer gleich, egal in welchem Typ die Parameter vorliegen. Sobald an der Division mindestens ein komplexer Datensatz beteiligt ist, werden alle Einheiten auf dB überprüft und die Daten entsprechend behandelt. Die Division von Daten in dB wird ausgeführt, indem dB-Zahlen subtrahiert werden.

Anmerkungen

- Unter bestimmten Voraussetzungen können auch strukturierte Datensätze (Events/Segmente) verrechnet werden. Dazu muss entweder einer der beiden Parameter ein Einzelwert sein, oder beide Parameter müssen exakt die gleiche Struktur besitzen (d.h. Gesamtlänge, Segmentlänge, Eventzahl und -Längen müssen gleich sein).
- Für eine sinnvolle Berechnung müssen die x-Skalierungen von beiden beteiligten Datensätzen gleich sein. Sind sie nicht gleich, wird eine Warnung generiert. Es werden dann die Angaben zur ersten Variable benutzt.
- Komplexe Datensätze werden nach den Regeln der komplexen Rechnung punktweise dividiert. Beachten Sie, dass bei der Division von komplexen Zahlen in kartesischer Darstellung nicht Real- und Imaginärteil einzeln dividiert werden. Beachten Sie weiter, dass bei Darstellungen in Polarkoordinaten Phasen und dB-Zahlen subtrahiert werden.
- Haben die übergebenen Datensätze unterschiedliche Längen, entspricht die Länge des Ergebnisses der des kürzeren Datensatzes.

Beispiele:

Ein Datensatz wird mit einem Bezugswert normiert:

```
NDnorm = NDdaten / Ewbezug
XYdaten.Y = XYdaten.Y / Ewbezug
```

Übertragungsfunktion als Quotient von zwei Spektren:

```
BPübertr = BPspek1 / RIspek2
```

Übertragungsfunktion (in dB) als Quotient von zwei Spektren, die in dB vorliegen:

```
DPübertr = DPspek1 / DPspek2
```

Normierung eines Spektrums mit einem Bezugswert. Ist die Einheit des Bezugswertes dB, wird er als ein Wert in Dezibel interpretiert und entsprechend verrechnet:

```
BPübertr = BPspek / EWbezug
```

Division von zwei reellen Datensätzen, wobei einer in dB vorliegt. Da kein komplexer Datensatz beteiligt ist, werden die dB nicht automatisch erkannt. Deshalb ist auf den Datensatz, der in dB vorliegt, die Funktion zur Berechnung der inversen dB anzuwenden.

```
NDdat = idB(NDdb) / NDkeindb
; oder alternativ:
NDdatdb = NDdb - dB(NDkeindb)
```

Siehe auch:

*(Multiplikation), Div

^

Potenz-Operator (hoch)

Deklaration:

```
Basis ^ Exponent -> Ergebnis
```

Parameter:

Basis	Erster Parameter, Basis
Exponent	Zweiter Parameter, Exponent
Ergebnis	Potenz (Basis hoch Exponent).

Beschreibung:

Die Potenzfunktion berechnet Basis hoch Exponent. Die Potenzfunktion bearbeitet Datensätze Punkt für Punkt unabhängig von deren x-Skalierungen.

Wird ein Einzelwert mit einem Datensatz verknüpft, wird der Einzelwert auf jeden Wert des Datensatzes angewendet.

Anmerkungen

- Ist die Basis positiv, darf der Exponent jeden Wert annehmen.
- Ist die Basis kleiner Null, darf der Exponent nur ganzzahlig sein.
- Ist die Basis Null, muss der Exponent positiv sein.
- Nach Möglichkeit wird die Einheit des Ergebnisses aus der Einheit der Basis und dem Wert des Exponenten bestimmt. Das ist nur möglich, wenn der Exponent ein Einzelwert ist. Ansonsten wird die Einheit der Basis beibehalten.
- Das Zeichen ^ bindet in einer Formel stärker als Punktrechnung. Zum Beispiel wird 4 * 3 ^ 2 als 4 * (3 ^ 2) verstanden und zu 36 bewertet.
- Unter bestimmten Voraussetzungen können auch strukturierte Datensätze (Events/Segmente) verrechnet werden. Dazu muss entweder einer der beiden Parameter ein Einzelwert sein, oder beide Parameter müssen exakt die gleiche Struktur besitzen (d.h. Gesamtlänge, Segmentlänge, Eventzahl und -Längen müssen gleich sein).
- Haben die übergebenen Datensätze unterschiedliche Längen, entspricht die Länge des Ergebnisses der des kürzeren Datensatzes.

Beispiele:

SVpower wird der Wert 8 V^3 zugewiesen:

```
SVpower = 2 'V' ^ 3
```

Es wird der Kehrwert der 3. Wurzel des Datensatzes berechnet. Ist die Einheit des Datensatzes V^3, wird die Einheit des Ergebnisses 1/V.

```
NDpower = NDdata ^ (-1/3)
```

Siehe auch:

*(Multiplikation), sqr, sqrt, exp

+

Punktweise Addition

Deklaration:

```
Summand1 + Summand2 -> Summe
```

Parameter:

Summand1	Erster Summmand
Summand2	Zweiter Summand.
Summe	Summe, Ergebnis der punktweisen Addition.

Beschreibung:

Es wird die Summe von zwei Variablen gebildet. Datensätze werden punktweise addiert unabhängig von den Maßstäben ihrer x-Achsen.

Bei der Addition von einem Einzelwert und einem Datensatz wird der Einzelwert zu jedem Punkt des Datensatzes addiert.

Für eine zeit- bzw. x-richtige Addition können Sie die Funktion Add verwenden.

Dieser Operator kann auch verwendet werden, um Texte aneinanderzuhängen (analog zur Funktion TAdd()).

Anmerkungen

- Unter bestimmten Voraussetzungen können auch strukturierte Datensätze (Events/Segmente) verrechnet werden. Dazu muss entweder einer der beiden Parameter ein Einzelwert sein, oder beide Parameter müssen exakt die gleiche Struktur besitzen (d.h. Gesamtlänge, Segmentlänge, Eventzahl und -Längen müssen gleich sein).
- Beim Arbeiten mit XY-Datensätzen müssen Sie die gewünschte Komponente angeben. Einen Einzelwert können Sie auch direkt auf einen XY-Datensatz addieren, es wird dann zu jedem Y-Wert der Einzelwert addiert.
- Für eine sinnvolle Berechnung müssen die x-Skalierungen von beiden beteiligten Datensätzen gleich sein. Sind sie nicht gleich, wird eine Warnung generiert. Es werden dann die Angaben zur ersten Variable benutzt.
- Bei der Addition sind nicht alle Kombinationen mit komplexen Datentypen möglich. Wenn Sie einen Datensatz vom Typ DP (Komplexer Datensatz in Polarkoordinaten, Betrag in dB) addieren möchten, müssen Sie ihn zunächst in einen anderen Typ wandeln, zweckmäßigerweise mit der Funktion idB().
- Komplexe Datensätze werden nach den Regeln der komplexen Rechnung punktweise addiert. Beachten Sie, dass bei der Addition von komplexen Zahlen in Polarkoordinaten-Darstellung nicht die Beträge und Phasen einzeln addiert werden.
- Haben die übergebenen Datensätze unterschiedliche Längen, entspricht die Länge des Ergebnisses der des kürzeren Datensatzes.

Beispiele:

Offset-Korrektur, Addition einer festen Zahl:

```
NDkorrek = NDdaten + EWoffset
```

Die Amplitude eines XY-Datensatzes wird korrigiert:

```
XYkorr.Y= XYkorr.Y+2
```

Überlagerung von zwei Spektren:

```
BPüberla = BPspek1 + BPspek2
```

Überlagerung von zwei Spektren, wobei das eine in dB vorliegt:

```
BPüberla = BPspek1 + idB (DPspek2)
```

Zwei Möglichkeiten einer Addition eines reellen Datensatzes zu einem komplexen Datensatz RIspek mit den Komponenten RIspek.R und RIspek.I:

```
RIsumme = Compl(RIspek.R + NDreal, RIspek.I)
RIsumme = RIspek + NDreal
```

Zwei Möglichkeiten einer Addition eines Einzelwertes zu einem komplexen Datensatz RIspek mit den Komponenten RIspek.R und RIspek.I:

```
RIsumme = Compl(RIspek.R + EWoffset, RIspek.I)
RIsumme = RIspek + EWoffset
```

Siehe auch:

-(Subtraktion), Add, MatrixAdd, Append

=

Vergleichsoperator, "gleich"

Deklaration:

```
Operand1 = Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert/Datensatz/Text/Textfeld.
Operand2	Zweiter zu vergleichender Einzelwert/Datensatz/Text/Textfeld.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

Test auf Gleichheit. Das Ergebnis ist 1, wenn beide Operanden gleich sind. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte, Datensätze, Texte und Textfelder angewendet werden.

Bei Datensätzen erfolgt ein punktweiser Vergleich.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Zwei Textfelder gelten als identisch, wenn sie die selbe Dimension besitzen und die Texte mit selben Index jeweils gleich sind.

Beim Vergleich von Texten erfolgt keine Unterscheidung zwischen Groß- und Kleinschreibung.

Beispiele:

Es wird gepüft, ob ein Datensatz eine gerade Anzahl von Samples enthält.

```
Channel1 = ...
Len = leng?(Channel1)
IF mod(len, 2) = 1
   BoxMessage("Fehler", "Ungültige Datensatzlänge", "!1")
END
```

Zwei digitale Datensätze werden verglichen. Der Ergebnisdatensatz ist überall dort 1, wo die Operandendatensätze den gleichen Wert aufweisen.

```
Result = (DigChannel1 = DigChannel2)
```

Die Einheit eines Datensatzes wird auf "Bar" geprüft und gegebenenfalls in "Pascal" umgerechnet.

```
IF Unit?(Pressure_1, 1) = "Bar"
   Pressure_1 = Pressure_1 * 1e5
   SetUnit(Pressure_1, "Pa", 1)
END
```

Siehe auch:

<>, <=, >, TComp

ABCRating

A-, B- oder C-Frequenzbewertung nach DIN EN 61672-1 (DIN IEC 651)
Alternativer Name: **ABCBewertung**

Deklaration:

```
ABCRating ( Signal, EwTyp, EwZeitbewertung, EwReduktion, Null ) -> Bewertet
```

Parameter:

Signal	Zu bewertendes Signal [ND].
EwTyp	Bewertungs-Typ
	1 : A-Bewertung
	2 : B-Bewertung
	3 : C-Bewertung
EwZeitbewertung	Nachträgliche Zeitbewertung
	0 : Keine Zeitbewertung
	>=0 : Zeitkonstante für Mittelung
	-1 : FAST-Bewertung
	-2 : SLOW-Bewertung
	-3 : IMPULSE-Bewertung
	-4 : PEAK-Bewertung
EwReduktion	Nachabtastung
	0 : Gleichgewichteter Effektivwert über das gesamte frequenzbewertete Signal. Die angegebene Zeitbewertung wird ignoriert.
	1 : Keine Nachabtastung
	>1 : Faktor für Nachabtastung
Null	Reservierter Parameter, immer 0.
Bewertet	Frequenzbewertetes Signal

Beschreibung:

Die Funktion führt eine A-, B- oder C-Frequenzbewertung eines Signals nach DIN IEC 651 (Schallpegelmesser) durch.

Zusätzlich kann eine nachträgliche Zeitbewertung (gleitender Effektivwert mit exponentieller Mittelung) und eine Nachabtastung ausgeführt werden.

Die A-Bewertung entspricht IEC 61672-1, 1st edition, 2002-05, Class1 und DIN IEC 651, 1981, Klasse 0.

Die aus den Anforderungen der Norm resultierenden Bandpässe haben in etwa die folgenden Grenzfrequenzen:

Bewertung	Untere Grenzfrequenz	Obere Grenzfrequenz
A	500 Hz	11 kHz
B	160 Hz	8 kHz
C	31.5 Hz	8 kHz

Für A-Bewertung muß die Abtastfrequenz größer als 3.34 kHz sein, für B- und C-Bewertung muss sie muss oberhalb der unteren Grenzfrequenzen der Filter liegen. Idealerweise sollte die Abtastfrequenz deutlich über der oberen Grenzfrequenz liegen.

Die vordefinierten Zeitbewertungen haben folgende Bedeutung:

FAST	Zeitkonstante = 0.125 s.
SLOW	Zeitkonstante = 1 s.
Impulse	Bei ansteigender Amplitude beträgt die Zeitkonstante 35ms, bei abfallender Amplitude 1.5s. Damit werden impulsförmige Signale schnell erfasst, die Anzeige klingt langsam ab.
Peak	Extreme Anzeige für ganz kurze Impulse, wobei garantiert der Spitzenwert gezeigt wird. Bei ansteigender Amplitude Zeitkonstante Null (ist im Computer exakt machbar, in Analogschaltung nur näherungsweise). Bei abfallender Amplitude 3s.

Eine Tabelle mit den Frequenzbewertungskurven für die einzelnen Bewertungen finden Sie bei der Beschreibung der Terzanalyse - OctA().

Beispiele:

```
SignalA = ABCRating(Signal, 1, 0.2, 2, 0)
```

Das Signal wird einer A-Bewertung unterzogen. Das frequenzbewertete Signal wird dann mit einer Zeitkonstante von 0.2s zeitbewertet und mit dem Faktor 2 nachabgetastet.

```
SignalCRms = ABCRating(Signal, 3, 0, 0)
```

Der gleichgewichtete Effektivwert des C-bewerteten Signals wird bestimmt.

Siehe auch:

OctA, ExpoRMS, RMS

Abs

Absoluter Betrag.

Deklaration:

```
Abs ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Parameter. Erlaubte Typen: [ND],[XY].
Ergebnis	Absoluter Betrag des Parameters

Beschreibung:

Es wird der Absolutbetrag von reellen Zahlen gebildet. Positive Zahlen bleiben unverändert, bei negativen wird das Vorzeichen umgekehrt. Diese Funktion simuliert einen idealen Gleichrichter.

Anmerkungen

- Die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Die Einheit wird nicht verändert.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Gleichrichtung eines Sinus:

```
NDhalfWave = Abs(NDsinus)
```

Siehe auch:

Round, Mod

ACF

Autokorrelationsfunktion
Alternativer Name: **AKF**

Deklaration:

```
ACF ( EingangsDaten ) -> Ergebnis
```

Parameter:

EingangsDaten	Datensatz, der mit sich selbst korreliert werden soll
Ergebnis	Ergebnis der Autokorrelation

Beschreibung:

Die Autokorrelationsfunktion ist der Spezialfall einer Kreuzkorrelationsfunktion von zwei identischen Datensätzen. Autokorrelation heißt, dass ein Datensatz mit sich selbst korreliert wird.

In der vorliegenden Implementierung wird der Datensatz, der als Parameter übergeben wird, periodisch nach beiden Seiten fortgesetzt gedacht. Stellt der Datensatz (das Signal) einen einzigen Impuls dar, so wird das Signal interpretiert, als wäre es eine Aneinanderreihung von vielen Impulsen, alle gleich geformt und gleich lang. Dieses Signal wird mit sich selbst verglichen, wobei es einmal etwas in x-Richtung verschoben wird.

Die Autokorrelationsfunktion gibt nun für jede Verschiebung an, wie ähnlich sich die Signale bei dieser Verschiebung sind. Dabei bedeutet ein Wert von 1, dass beide identisch sind. Das ist der Fall bei der Verschiebung von 0, aber auch bei einer Verschiebung um eine Periodendauer. Ein Wert von -1 bedeutet, dass beide Signale entgegengesetzt gleich sind. Ist ein Signal positiv, ist das andere genauso groß, aber negativ.

Ein Wert von 0 bedeutet, dass die beiden Signale bei dieser Verschiebung nichts miteinander zu tun haben (nicht korreliert sind). Es können alle Werte zwischen -1 und +1 auftreten.

Der erzeugte Datensatz wird über eine Periode berechnet. Die Autokorrelationsfunktion ist selbst wieder periodisch, so dass es nicht nötig ist, sie über einen größeren Bereich zu berechnen. Der erzeugte Datensatz ist achsensymmetrisch zu seiner Mitte (in x-Richtung). Das liegt daran, dass beim Vergleich von zwei identischen periodischen Funktionen es gleich ist, ob eine Funktion um 0.9 Perioden in eine Richtung geschoben wird oder um 0.1 Perioden in die andere oder auch um 0.1 Perioden in dieselbe Richtung. Die eine Hälfte des erzeugten Datensatzes enthält also redundante Daten, die Sie abschneiden können.

Um eine akzeptable Rechengeschwindigkeit zu erzielen, wird die Autokorrelationsfunktion unter Zuhilfenahme der Funktionen FFT(), iFFT() berechnet. Dies hat neben der gewaltigen Rechenzeitersparnis vor allem bei längeren Datensätzen zwei wichtige Konsequenzen:

Sie können zur Vorverarbeitung die Fensterfunktion benutzen, die Sie für die FFT eingestellt haben. Möchten Sie keine Fensterung benutzen, wählen Sie das Rechteckfenster, siehe FFT. Zum anderen ist die Länge der verarbeitbaren Datensätze auf Zweierpotenzen bis 2^27 beschränkt. Wenn ein Datensatz eine andere Länge aufweist, wird er (noch vor einer eventuellen Fensterung) auf die nächstkleinere Zweierpotenz von Zahlenwerten abgeschnitten. Wenn aber keine Werte abgeschnitten werden dürfen, benutzen Sie vorher die Funktion Red2(), siehe Beispiele.

- Die hier implementierte AKF liefert stets normierte Werte, das Ergebnis hat also keine Einheit. Es wird auf die Summe der Quadrate der Werte des Datensatzes normiert. D. h. es wird auf den Wert der unnormierten Autokorrelationsfunktion an der Stelle Null normiert. Dieser Wert ist das Quadrat des Effektivwertes.
- Wenn die Länge des als Parameter übergebenen Datensatzes 2^27 überschreitet, wird eine entsprechende Fehlermeldung ausgegeben. Es ist dann keine Berechnung möglich. Verkürzen Sie dann den Datensatz, mit den Funktionen Leng() oder Red2(). Die maximal bearbeitbare Länge eines Datensatzes ist 134.217.728.
- Wenn die Länge des als Parameter übergebenen Datensatzes keine Zweierpotenz ist, wird eine entsprechende Warnung erzeugt. Der Datensatz wird dann automatisch verkürzt (abgeschnitten).
- Da die Funktion ACF() intern die Funktion FFT() benutzt, wird temporärer Speicherplatz im Arbeitsspeicher benötigt. Falls kein ausreichender Speicherplatz vorhanden ist, wird eine entsprechende Fehlermeldung erzeugt. Die Berechnung wird abgebrochen.
- Ist der zu bearbeitende Datensatz mit einem hohen Mittelwert (y- Offset) versehen, lohnt es sich, den Datensatz vorerst mittelwertfrei zu machen. Ansonsten beeinflusst der Mittelwert das Korrelationsergebnis wesentlich stärker als das eigentliche Signal. In der Tat wird damit nicht mehr die Autokorrelationsfunktion, sondern die Autokovarianzfunktion berechnet.

Beispiele:

```
acf_result = ACF(data)
```

Einfachste Anwendung, periodische AKF, Datensatz wird ggf. auf eine Zweierpotenz von Punkten verkürzt.

```
acf_result = ACF(Red2(data - Mean(data)))
```

Periodische AKF, Berücksichtigung des gesamten Datensatzes möglich durch geeignete Nachabtastung. Der Datensatz wird mittelwertfrei gemacht, da er mit einem recht großen y-Offset versehen ist.

```
help = Red2(data)
help = Leng(help, 2 * Leng?(help))
acf_result = ACF(Red2(help))
acf_result = Leng(acf_result, 0.5 * Leng?(acf_result))
```

Dies ist ein Beispiel für eine nichtperiodische AKF. Das Signal wird dazu mit Nullen erweitert (Appending Zero in der Literatur), so dass trotz periodischer Berechnung der AKF selbst Nichtperiodizität vorgetäuscht wird. Am Ende ist die zweite Hälfte des Ergebnisses zu verwerfen. Der zweimalige Aufruf der Red2()-Funktion stellt sicher, dass an eine Zweierpotenz von gültigen Daten die gleiche Zahl von Nullen gehängt ist. Sie sollten hierbei die Fensterfunktion stets auf Rechteckfenster gestellt haben, siehe FFT().

Siehe auch:

CCF, CorrCoeff, FFT, Red2

acos

Arkuskosinus, Umkehrfunktion Kosinus

Deklaration:

```
acos ( Daten ) -> Winkel
```

Parameter:

Daten	Daten. Erlaubte Typen: [ND],[XY].
Winkel	Arkuskosinus des Parameters (Winkel in Bogenmaß)

Beschreibung:

Es wird die arccos-Funktion gebildet.

Die Funktion liefert einen Winkel im Bogenmaß ohne Einheit zurück.

Die x-Koordinate(n) von Ergebnis und Parameter sind gleich.

Anmerkungen

- Das Argument sollte keine Einheit haben, wenn doch, wird eine Warnung generiert, die Einheit wird unverändert übernommen
- Der zulässige Wertebereich des Parameters liegt zwischen -1 und +1.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

acos(0) = PI/2, acos(1) = 0, acos(-1) = PI

```
pihalf = acos(0.0)
```

Aus einem sinusförmigen Datensatz wird ein zackiger dreieckiger Datensatz:

```
NDspikes = acos(NDcos)
```

Mit der vordefinierten Konstante PI oder INDEGR wird der zurückgegebene Winkel vom Bogenmaß in das Gradmaß konvertiert:

```
value_90 = acos(0.0) * 180 '°' / PI
value_90 = acos(0.0) * INDEGR
```

Siehe auch:

cos, asin, atan2

Add

Zeit- bzw. x-richtige Addition

Deklaration:

```
Add ( Summand1, Summand2, EwOption ) -> Summe
```

Parameter:

Summand1	Erster Summand. Erlaubte Typen: [ND],[XY].
Summand2	Zweiter Summand. Erlaubte Typen: [ND],[XY].
EwOption	Option
	0 : Die Triggerzeit der beiden Summanden wird ignoriert
	1 : Zeitrichtige Überlagerung mit Berücksichtigung der Triggerzeit
Summe	Summe, Ergebnis der Addition. [XY]

Beschreibung:

Zwei Datensätze werden zeit- bzw. x-richtig addiert, d. h. es werden immer 2 y-Werte addiert, die die gleiche Zeit bzw. den gleichen x-Wert besitzen.

Das Ergebnis ist nur für den x-Bereich definiert, den die beiden Parameter-Datensätze gemeinsam haben. In diesem Bereich wird überall dort ein Ergebnis-Wert berechnet, wo mindestens einer der beiden Datensätze eine Stützstelle besitzt. Wenn an dieser Stelle der andere Datensatz keine Stützstelle besitzt, wird dieser dort linear interpoliert.

Die x-Spur beider Parameter-Datensätze muss monoton steigend sein, d.h. die x-Koordinaten müssen kontinuierlich wachsen.

Der Operator + (Addition) führt dagegen eine punktweise Addition von Datensätzen aus.

Beispiele:

Zwei Kanäle werden zwischen 11 und 13 Uhr bzw. 12 und 14 Uhr aufgenommen.

```
superpos = Add(voltage11_13h, voltage12_14h, 1)
```

Eine zeitrichtige Addition der beiden Datensätze mit Berücksichtigung der Triggerzeit wird ausgeführt. Das Ergebnis ist zwischen 12 und 13 Uhr definiert.

Siehe auch:

+(Addition), Sub, Mult, Div, Append

All0

Liefert die x-Position aller Nullstellen eines Datensatzes
Alternativer Name: **Alle0**

Deklaration:

```
All0 ( Daten ) -> XNullStellen
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND],[XY].
XNullStellen	Die ermittelten X-Koordinaten der Nullstellen.

Beschreibung:

Die x-Koordinaten aller Nullstellen bilden das Ergebnis. Eine Nullstelle liegt dann vor, wenn Datenpunkte das Vorzeichen wechseln oder gleich Null sind. Die Funktion All0() interpoliert nicht. Liegt eine Nullstelle zwischen zwei Datenpunkten, wird deshalb die größere x-Koordinate ausgegeben.

Falls der Parameter ein XY-Datensatz ist, muss dieser eine monotone Zeit- bzw. x-Spur aufweisen.

- Eine führende Null im Datensatz wird nicht als Nullstelle interpretiert.
- Wenn keine Nullstellen gefunden werden, wird ein leerer Datensatz zurückgegeben.
- Für eine interpolierende Nullstellensuche auch für XY-Datensätze können Sie die Funktion SearchLevel() benutzen.

Beispiele:

```
xnull = All0(data)
```

Das Ergebnis sind die x-Koordinaten aller Nullstellen des Datensatzes.

Siehe auch:

SearchLevel, Top, xMax, PolynomRoots, PosiEx

AmpSpectrumPeak

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Amplitudenspektrum (Harmonische als Spitzenwert bzw. Amplitude bestimmt) mit gleitendem Fenster und linearer Mittelung. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Deklaration:

```
AmpSpectrumPeak ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	0 : Keine Mittelung
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Betragsspektren). Der Mittelwert wird über so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	2 : Peak Hold Max ab Beginn, Maximalwerte, berechnet von allen bislang berechneten Spektren
	3 : Peak Hold Max pro Intervall, Maximalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	4 : Peak Hold Min ab Beginn, Minimalwerte, berechnet von allen bislang berechneten Spektren
	5 : Peak Hold Min pro Intervall, Minimalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	6 : Quadratischer Mittelwert über alle berechneten Betragsspektren
	7 : Quadratischer Mittelwert über alle berechneten Betragsspektren. Das Ergebnis wird durch sqrt(ENBW=Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch sqrt(1.5) im Fall des Hanning-Fensters.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz.

Beschreibung:

Beispiele:

```
Spektren = AmpSpectrumPeak ( Kanal, 1000, 0, 50, 1, 0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen.

```
Spektren = AmpSpectrumPeak ( Kanal, 2048, 1, 0, 10, 6, 0 )
```

Das ist die Berechnung einer Folge von 2048 Punkte-Spektren mit Hamming-Fenster. Über je 10 Spektren wird der Mittelwert bestimmt und im Ergebnis vermerkt.

Siehe auch:

AmpSpectrumPeak_exp, AmpSpectrumPeak_1, AmpSpectrumRMS

AmpSpectrumPeak_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein gemitteltes Amplitudenspektrum (Harmonische als Spitzenwert bzw. Amplitude bestimmt) wird bestimmt. Die Mittelung erfolgt über viele Spektren, die aus einem gleitenden Fenster ermittelt wurden.

Deklaration:

```
AmpSpectrumPeak_1 ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Betragsspektren). Der Mittelwert wird über alle berechneten Spektren gebildet.
	2 : Peak Hold Max, Maximalwerte, berechnet von allen berechneten Spektren
	4 : Peak Hold Min, Minimalwerte, berechnet von allen berechneten Spektren
	6 : Quadratischer Mittelwert über alle berechneten Betragsspektren
	7 : Quadratischer Mittelwert über alle berechneten Betragsspektren. Das Ergebnis wird durch sqrt(ENBW=Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch sqrt(1.5) im Fall des Hanning-Fensters.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ein gemitteltes Spektrum als Ergebnis.

Beschreibung:

Beispiele:

```
Spektrum = AmpSpectrumPeak_1 ( Kanal, 1000, 0, 50, 6, 0 )
```

Das ist die Berechnung eines gemittelten Spektrums. Die Mittelung geschieht über eine Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Eingangskanal hat etwa 20000 Messwerte.

Siehe auch:

AmpSpectrumPeak_exp, AmpSpectrumPeak, AmpSpectrumRMS_1

AmpSpectrumPeak_exp

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Amplitudenspektrum (Harmonische als Spitzenwert bzw. Amplitude bestimmt) mit gleitendem Fenster und exponentieller Mittelung. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Deklaration:

```
AmpSpectrumPeak_exp ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Zeitkonstante [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Zeitkonstante	Die Zeitkonstante für die exponentielle Mittelung. In Sekunden angegeben.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz.

Beschreibung:

Beispiele:

```
Spektren = AmpSpectrumPeak_exp ( Kanal, 1000, 0, 50, 2, 40.0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Kanal hat eine Abtastzeit von 10ms. Also wird alle 5s ein Spektrum gebildet. Diese werden mit einer Zeitkonstante von 40.0s geglättet. Jedes 2. Spektrum wird ausgegeben.

Siehe auch:

AmpSpectrumPeak_1, AmpSpectrumPeak, AmpSpectrumRMS_exp

AmpSpectrumRMS

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Amplitudenspektrum (Harmonische als Effektivwerte bestimmt) mit gleitendem Fenster und linearer Mittelung. Berechnung mittels FFT. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Deklaration:

```
AmpSpectrumRMS ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	0 : Keine Mittelung
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Betragsspektren). Der Mittelwert wird über so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	2 : Peak Hold Max ab Beginn, Maximalwerte, berechnet von allen bislang berechneten Spektren
	3 : Peak Hold Max pro Intervall, Maximalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	4 : Peak Hold Min ab Beginn, Minimalwerte, berechnet von allen bislang berechneten Spektren
	5 : Peak Hold Min pro Intervall, Minimalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	6 : Quadratischer Mittelwert über alle berechneten Betragsspektren
	7 : Quadratischer Mittelwert über alle berechneten Betragsspektren. Das Ergebnis wird durch sqrt(ENBW=Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch sqrt(1.5) im Fall des Hanning-Fensters.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz.

Beschreibung:

Beispiele:

```
Spektren = AmpSpectrumRMS ( Kanal, 1000, 0, 50, 1, 0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen.

```
Spektren = AmpSpectrumRMS ( Kanal, 2048, 1, 0, 10, 6, 2 )
```

Das ist die Berechnung einer Folge von 2048 Punkte-Spektren mit Hamming-Fenster. Über je 10 Spektren wird der Mittelwert bestimmt und im Ergebnis vermerkt.

Siehe auch:

AmpSpectrumRMS_exp, AmpSpectrumRMS_1, AmpSpectrumPeak

AmpSpectrumRMS_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein gemitteltes Amplitudenspektrum (Harmonische als Effektivwerte bestimmt) wird bestimmt. Die Mittelung erfolgt über viele Spektren, die aus einem gleitenden Fenster ermittelt wurden. Berechnung mittels FFT.

Deklaration:

```
AmpSpectrumRMS_1 ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Betragsspektren). Der Mittelwert wird über alle berechneten Spektren gebildet.
	2 : Peak Hold Max, Maximalwerte, berechnet von allen berechneten Spektren
	4 : Peak Hold Min, Minimalwerte, berechnet von allen berechneten Spektren
	6 : Quadratischer Mittelwert über alle berechneten Betragsspektren
	7 : Quadratischer Mittelwert über alle berechneten Betragsspektren. Das Ergebnis wird durch sqrt(ENBW=Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch sqrt(1.5) im Fall des Hanning-Fensters.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ein gemitteltes Spektrum als Ergebnis.

Beschreibung:

Koeffizienten der Fensterfunktionen: Hamming: 1, -0.46/0.54; Hanning: 1, -1; Blackman: 1, -0.50/0.42, 0.08/0.42; Blackman-Harris: 1, -0.48829/0.35875, 0.14128/0.35875, -0.01168/0.35875; Flat top: 1, -1.93, 1.29, -0.388, 0.0322

Beispiele:

```
Spektrum = AmpSpectrumRMS_1 ( Kanal, 1000, 0, 50, 6, 0 )
```

Das ist die Berechnung eines gemittelten Spektrums. Die Mittelung geschieht über eine Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Eingangskanal hat etwa 20000 Messwerte.

Siehe auch:

AmpSpectrumRMS, AmpSpectrumRMS_exp, AmpSpectrumPeak_1

AmpSpectrumRMS_exp

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Amplitudenspektrum (Harmonische als Effektivwerte bestimmt) mit gleitendem Fenster und exponentieller Mittelung. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Deklaration:

```
AmpSpectrumRMS_exp ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Zeitkonstante [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Zeitkonstante	Die Zeitkonstante für die exponentielle Mittelung. In Sekunden angegeben.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz.

Beschreibung:

Beispiele:

```
Spektren = AmpSpectrumRMS_exp ( Kanal, 1000, 0, 50, 2, 40.0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Kanal hat eine Abtastzeit von 10ms. Also wird alle 5s ein Spektrum gebildet. Diese werden mit einer Zeitkonstante von 40.0s geglättet. Jedes 2. Spektrum wird ausgegeben.

Siehe auch:

AmpSpectrumRMS_1, AmpSpectrumRMS, AmpSpectrumPeak_exp

AND

Logischer "UND"-Operator

Deklaration:

```
Operand1 AND Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Operand2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

"UND"-Verknüpfung zweier Zahlen. Das Ergebnis ist 1, wenn keiner der Operanden 0 ist. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte und auf Datensätze angewendet werden. Bei Datensätzen erfolgt eine punktweise Verknüpfung.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Das Maximum eines Datensatzes wird bestimmt und auf Überschreiten eines Grenzwertes getestet.

```
Channel1 = ...
Maximum = max(Channel1)
IF Maximum > 21 AND Maximum < 34
   BoxMessage("Achtung", "Maximum nahe Grenzwert!", "!1")
ELSE
   IF Maximum >= 34
      BoxMessage("Achtung", "Grenzwert ist überschritten", "!1")
   END
END
```

Zwei Datensätze mit gleicher Zeitbasis werden untersucht. Es werden alle Zeitpunkte ermittelt, an denen beide Datensätze oberhalb einer gegebenen Schwelle liegen:

```
t = Top((Channel1 > 3) OR (Channel2 > 3), 0.9)
```

Zwei digitale Datensätze werden verknüpft. Der Ergebnisdatensatz ist überall dort 1, wo beide Operandendatensätze den Wert 1 aufweisen.

```
Result = (DigChannel1 AND DigChannel2)
```

Siehe auch:

NOT, OR, XOR

Append

Zeit- bzw. x-richtiges Verbinden von Datensätzen

Deklaration:

```
Append ( Daten1, Daten2, EwOption ) -> Verbunden
```

Parameter:

Daten1	Erster Datensatz. Erlaubte Typen: [ND],[XY].
Daten2	Zweiter Datensatz. Erlaubte Typen: [ND],[XY].
EwOption	Option
	0 : Die Triggerzeit der beiden Summanden wird ignoriert
	1 : Zeitrichtige Überlagerung mit Berücksichtigung der Triggerzeit
Verbunden	Resultierender Datensatz. [XY]

Beschreibung:

Die Funktion gestattet das zeit-/x-richtige Aneinanderhängen bzw. Verschmelzen von Datensätzen.

Das Ergebnis besitzt Punkte an allen Stützstellen, die in [Daten1] **oder** [Daten2] definiert sind. Falls eine Stützstelle in beiden Datensätzen definiert ist, wird der Mittelwert aus den zugehörigen y-Werten verwendet.

Beide Datensätze müssen eine monotone Zeit- bzw. x-Achse besitzen.

Die Funktion Join() führt dagegen ein punktweises Aneinanderhängen von Datensätzen aus.

Beispiele:

Zwei Messkanäle, die die gleiche physikalische Größe messen, werden zwischen 11 und 13 Uhr bzw. 12 und 14 Uhr aufgenommen.

```
voltage11_14h = Append(voltage11_13h, voltage12_14h, 1)
```

Die beiden Datensätze werden verschmolzen. Das Ergebnis ist von 11- 14 Uhr definiert. Zwischen 11-12 Uhr bzw. 13-14 Uhr werden die Werte des ersten bzw. zweiten Datensatzes verwendet. Zwischen 12 und 13 Uhr werden die Messwerte beider Kanäle zeitrichtig einsortiert. Wenn beide Kanäle zur gleichen Zeit einen Messwert liefern, wird der Mittelwert verwendet.

Siehe auch:

Join, JoinEx, AppendLoop, Add, Sub, Mult, Div

AppendLoop

Verfügbar ab: Professional Edition

Hängt an einen Datensatz weitere Daten bzw. Werte an. Die Funktion ist optimiert für den Aufruf innerhalb einer Schleife, in der wiederholt kleine Datenmengen angehängt werden.

Deklaration:

```
AppendLoop ( Datensatz, Append )
```

Parameter:

Datensatz	An diesen Datensatz werden Daten angehängt. Die Funktion ändert die übergebene Variable.
Append	Anzuhängende Daten

Beschreibung:

Alle Werte des Parameters Append werden an den Datensatz angehängt.

Die Skalierung (Einheiten, x0, dx etc.) des Datensatzes bleibt erhalten.

Das Zahlenformat (z.B. digital oder ganze Zahlen mit Wertebereich oder reelle Zahlen) des Datensatzes bleibt erhalten; nur nicht, wenn der Datensatz leer ist und nichtleere Daten angehängt werden. Dann wird das Zahlenformat der angehängten Daten übernommen.

Der Datensatz kann ein normaler, XY- oder komplexer Datensatz sein. Er darf keine Events haben. Die anzuhängenden Daten sind vom selben Typ.

Für segmentierte Daten gilt: Hat der Datensatz Segmente, darf der Parameter Append auch Segmente gleicher Länge haben. Alternativ darf der Parameter Append keine Segmente haben, wenn seine Länge der Segmentlänge des Datensatzes entspricht.

Als Datensatz kann nur eine komplette Variable angegeben werden: Teile (z.B. A.y, A[2]) oder Formeln (z.B. 2*A) sind nicht erlaubt.

Die Funktion arbeitet zur Steigerung der Ausführungsgeschwindigkeit mit vergrößertem Speicher am Datensatz. Die höhere Ausführungsgeschwindigkeit macht sich typisch in einer Schleife bemerkbar, wenn die Funktion zum selben Datensatz viele Male aufgerufen wird. Ist die anzuhängende Datenmenge klein (z.B. weniger als 1000 Samples), ist die Ausführungsgeschwindigkeit deutlich höher.

Wenn dann das Anhängen an einen Datensatz abgeschlossen ist, typisch nach einer Schleife, bleibt ggf. der erhöhte Speicher am Datensatz bestehen. Mit einem Aufruf von AppendLoopEnd() wird dieser zusätzliche Speicher entfernt. Der Aufruf von AppendLoopEnd() bleibt ohne Wirkung, wenn kein erhöhter Speicher (mehr) besteht. Der Aufruf von AppendLoopEnd() entfällt, wenn bei der Arbeit der erhöhte Speicherbedarf ohne Belang ist.

Werden Kopien des Datensatzes oder seiner Komponenten angefertigt, haben auch die Kopien den erhöhten Speicherbedarf.

Die Funktion liefert vor allem dann einen zeitlichen Vorteil, wenn dem Programm ausreichend RAM zur Verfügung steht.

Für TSA-Daten wird TsaAppend() benutzt.

Beispiele:

A = leer

xdelta A 0.1

xeinheit A V

```
for i = 1 to 2000
   y=3*i
   AppendLoop(A, y)
end
AppendLoopEnd(A)
```

A = XYof (empty, empty)

```
for i = 1 to 2000
   x=i
   y=2*i
   AppendLoop(A, XYof(x, y))
end
AppendLoopEnd(A)
```

Siehe auch:

AppendLoopEnd, Binde, Append, TsaAppend, EventAppend

AppendLoopEnd

Verfügbar ab: Professional Edition

Nachdem (in einer Schleife) die Funktion AppendLoop (mehrfach) aufgerufen wurde, wird als Abschluss dieser Folge der zusätzlich benötigte Speicher wieder freigegeben.

Deklaration:

```
AppendLoopEnd ( Datensatz )
```

Parameter:

Datensatz	Der Datensatz, an den mit AppendLoop angehängt wurde.

Beschreibung:

Alle Einschränkungen wie bei AppendLoop.

Mehrfacher Aufruf oder ein Aufruf ohne vorheriges AppendLoop ist ohne Schaden.

Beispiele:

A = leer

xdelta A 0.1

xeinheit A V

```
for i = 1 to 2000
   y=3*i
   AppendLoop(A, y)
end
AppendLoopEnd(A)
```

Siehe auch:

AppendLoop

APPLICATION

Windows-Applikation oder DOS-Programm starten
Alternativer Name: **APPLIKATION**

Der Befehl ist veraltet, statt dessen sollte die leistungsfähigere Funktion Execute() verwendet werden.

Deklaration:

```
APPLICATION Modus Name Parameter
```

Parameter:

Modus	Fenster-Modus, mit dem das Programm aufstarten soll
	NORMAL : Programm startet in normaler Größe
	VOLLBILD : Programm starten als Vollbild (maximiert)
	SYMBOL : Programm startet minimiert (Icon)
Name	Name (Programmpfad) des zu startenden Programms
Parameter	Kommandozeilen-Parameter für das zu startende Programm

Beschreibung:

Dieser Befehl startet eine Windows-Applikation oder ein DOS-Programm. Sie können dabei auch Kommandozeilen-Parameter (Befehlszeilen-Parameter) übergeben und den Modus, unter dem die Applikation unter Windows starten soll, festlegen

Beispiele:

```
APPLICATION NORMAL Notepad.exe
APPLICATION MAXIMIZED c:\windows\Notepad.exe
```

Zwei Möglichkeiten, die Windows-Applikation 'Notepad' zu starten.

Siehe auch:

Execute

Appro

Approximation der Werte eines Datensatzes durch beliebige Funktionen

Deklaration:

```
Appro ( Daten, Fkt1, Fkt2, Fkt3, Fkt4, Fkt5, Fkt6, Fkt7, Fkt8 ) -> Koeffizienten
```

Parameter:

Daten	Datensatz, der approximiert werden soll. Erlaubte Datentypen: [ND]
Fkt1	Datensatz für den 1. Koeffizienten. [ND]
Fkt2	Datensatz für den 2. Koeffizienten. [ND]
Fkt3	Datensatz für den 3. Koeffizienten. [ND]
Fkt4	Datensatz für den 4. Koeffizienten. [ND]
Fkt5	Datensatz für den 5. Koeffizienten. [ND]
Fkt6	Datensatz für den 6. Koeffizienten. [ND]
Fkt7	Datensatz für den 7. Koeffizienten. [ND]
Fkt8	Datensatz für den 8. Koeffizienten. [ND]
Koeffizienten	Koeffizienten der approximierenden Funktion

Beschreibung:

Es wird eine Approximation des eingegebenen Datensatzes [Daten] durch die eingegebene - bis auf die Koeffizienten bekannte - Funktion durchgeführt. Die eingegebene Funktion kann sich aus bis zu acht Teilfunktionen zusammensetzen. Zu jeder Teilfunktion gehört ein unbekannter Koeffizient. Die m Koeffizienten der einzelnen Teilfunktionen werden bestimmt. Dazu wird die Methode der kleinsten Quadrate angewendet.

Die Vorgehensweise bei der Funktion Appro() wird folgend erläutert: Sei ein Messdatensatz mit den Werten g(x[i]) (i = 0, ..., Datensatzlänge - 1) gegeben. Der Messdatensatz soll durch eine - bis auf die Koeffizienten bekannte - Funktion approximiert werden. Diese Funktion kann sich aus mehreren Teilfunktionen zusammensetzen. Zu jeder Teilfunktion gehört ein Koeffizient. Die Koeffizienten werden durch die Funktion Appro bestimmt. Zuerst müssen Sie dazu die Datensätze der Funktionswerte für die bekannten Teilfunktionen f0, ..., fm-1 für alle x[i] bestimmen. Mit der Funktion Appro werden die m Koeffizienten k[0], ..., k[m-1] nach der Methode der kleinsten Quadrate so bestimmt, dass die Bedingung

$$g(x[i]) \approx k[0] \cdot f_0(x[i]) + k[1] \cdot f_1(x[i]) + \ldots + k[m-1] \cdot f_{m-1}(x[i])$$

für alle i möglichst gut erfüllt wird.

In der Reihenfolge, in der die Datensätze [Fkt] der Teilfunktionen eingegeben werden, werden auch die den Teilfunktionen zugehörigen Koeffizienten ausgegeben.

Setzt sich die zu approximierende Funktion aus weniger als acht Teilfunktionen zusammen, sind die Datensätze der Teilfunktionen [Fkt], deren Koeffizienten bestimmt werden sollen, einzugeben und die restlichen Parameter der Funktion Appro müssen dann auf beliebige Einzelwerte gesetzt werden (0 bietet sich an).

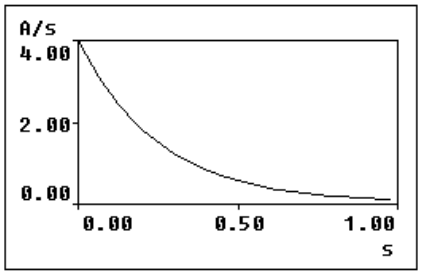
- Passen die eingegeben Datensätze nicht zueinander (verschiedene Länge, Einheiten, x0 oder DeltaX), wird eine Warnung ausgegeben. Sind die Datensätze verschieden lang, werden alle Datensätze für die Approximation auf die kleinste auftretende Datensatzlänge gesetzt. Liegen verschiedene Einheiten, x0 oder DeltaX vor, werden die Berechnungen mit den Größen des zu approximierenden Datensatzes NDdaten durchgeführt.
- Es können auch mehrere einzugebende Datensätze als Messdatensätze vorliegen. Mit der Funktion Appro() werden die zugehörigen Koeffizienten berechnet (siehe Beispiel).
- Die Funktion Appro ist eine Verallgemeinerung der Funktion Poly().
- Zur Genauigkeit siehe auch Anmerkungen bei der Funktion Poly().

Beispiele:

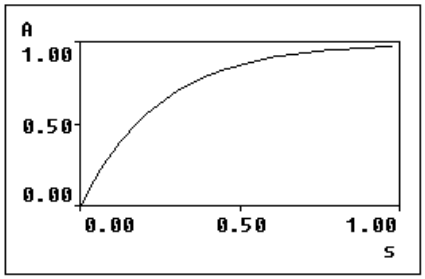
Der Messdatensatz der Ableitung der Funktion y und der Messdatensatz der Funktion y liegen vor. Es sollen die beiden Parameter k[0] und k[1] nach der Methode der kleinsten Quadrate so bestimmt werden, dass die Bedingung

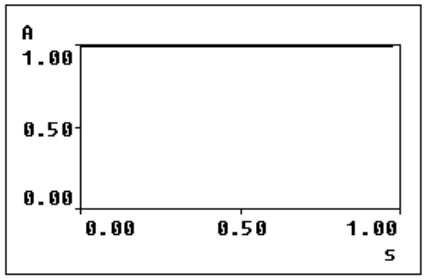
$$y'(x[i]) \approx k[0] \cdot y(x[i]) + k[1]$$

für alle i (i = 0, ..., Datensatzlänge - 1) möglichst gut erfüllt wird. Der zur folgenden Abbildung gehörige Datensatz NDdaten ist der Messdatensatz für y'. Der Datensatz enthält 50 Werte:



Der zum ersten Koeffizienten gehörige Datensatz NDdaten1 ist der Messdatensatz für y. Der Datensatz NDdaten1 wird in der linken Abbildung dargestellt. Der zum zweiten Koeffizienten gehörige Datensatz NDdaten2 besteht nur aus Werten von 1. Der Datensatz NDdaten2 wird in der rechten Abbildung dargestellt. Beide Datensätze enthalten 50 Werte:






```
NDkoeff = Appro(NDdaten, NDdaten1, NDdaten2, 0, 0, 0, 0, 0, 0)
```

In FAMOS wird als Ergebnis der Datensatz NDkoeff (Koeffizienten) ausgegeben. Im Datensatz NDkoeff befinden sich die beiden Werte koeff0 = -4.0000 und koeff1 = 4.0000.

So erhalten Sie die einzelnen Koeffizienten:

```
EWkoeff0 = NDkoeff[1]
EWkoeff1 = NDkoeff[2]
```

Siehe auch:

ApproNonLin, Poly, eFit, LFit, Value

ApproNonLin

Verfügbar ab: Professional Edition

Approximation durch eine Funktion, die in ihren Koeffizienten nichtlinear sein kann. Eine durch eine Formel beschriebene Funktion wird im Sinn kleinster Quadrate minimiert. Die Koeffizienten der Formel werden dabei ermittelt.

Deklaration:

```
ApproNonLin ( Input, Formel [, TimeOut] [, Fehlerbehandlung] [, Limits] [, Gewicht] [, Variante] ) -> Koeffizienten
```

Parameter:

Input	Input
Formel	Frei definierbare Formel
	"A1+A2*exp(A3*x)-y" : Abklingfunktion
	"A1+A2*exp(A3*x)*sin(A4*x+A5)-y" : Gedämpfte Schwingung
	"sqrt((X-A1)^2+(Y-A2)^2)-A3" : Kreis
	"A1*exp(-((x-A2)^2/(2*A3^2)))-y" : Dichte Normalverteilung
	"A1*exp(A2*x)-y" : Abklingfunktion ohne Offset
	"A1+x*(A2+x*A3)-y" : Polynom 2. Grades
	"A1+x*(A2+x*(A3+x*A4))-y" : Polynom 3. Grades
TimeOut	TimeOut in Sekunden. Wenn der Algorithmus innerhalb dieser Zeit nicht fertig ist, wird mit einem Timeout-Fehler abgebrochen. 0 für keine Überprüfung. (optional , Standardwert: 0)
Fehlerbehandlung	Wie soll im Fehlerfall reagiert werden? (optional , Standardwert: 0)
	0 : Abbruch mit Fehlermeldung
	1 : Leeren Datensatz zurückgeben
Limits	Limits für die Koeffizienten. Durch Komma getrennte Bedingungen, z.B. "A1 >= 0, A2 > 0, A2 < 1e3", die den Wertebereich der Koeffizienten einschränken. Mit init eingeleitete Initialisierungen sind Startwerte für die Iteration, z.B. "init A1 = 3". Der Parameter darf auch leer sein. Initialisierungen und Beschränkungen dürfen kombiniert werden, dürfen sich aber nicht widersprechen. (optional , Standardwert: "")
Gewicht	Gewicht zur unterschiedlichen Gewichtung der einzelnen Messwerte (optional)
Variante	Nach welcher Variante des Algorithmus wird berechnet? (optional , Standardwert: 0)
	0 : Automatisch. Die beste in der aktuellen Version verfügbare Variante des Algorithmus wird benutzt. Der Algorithmus kann in späteren Versionen von FAMOS weiterentwickelt werden.
	1 : So wie in FAMOS 7.2. Auch bei späteren Versionen von FAMOS verhält sich der Algorithmus unverändert.
	2 : So wie in FAMOS 7.3. Auch bei späteren Versionen von FAMOS verhält sich der Algorithmus unverändert.
Koeffizienten	Koeffizienten A1, A2, ...

Beschreibung:

Die angegebenen Formel im Sinn der kleinsten Quadrate minimiert: Über alle gegebenen Messwerte wird das Quadrat der Formel minimiert, also möglichst nahe null.

Ist z.B. die Formel "exp(A1*x)-y", so wird "Summe (exp(A1*x)-y)^2" minimiert.

I. Allg. soll die Differenz von Original y-Werten zur approximierenden Funktion gefunden werden. Ein Term -y ist dann in der Formel erforderlich.

Ist z.B. eine Messung durch eine Funktion "y=exp(A1*x)" zu approximieren, darf diese Funktion nicht selbst als Formel übergeben werden. Vielmehr sind die zu minimierenden Abweichungen gegenüber den Messwerten y anzugeben, also "exp(A1*x)-y".

Minimierung

Das gefundene lokale Minimum muss nicht mit dem globalen Minimum übereinstimmen.

Das wurde von Dr. Knopp anschaulich beschrieben: Angenommen, man befindet sich in einer Hügellandschaft. Es ist nebelig, man kann nicht weit schauen. Es geht abwärts und man folgt der Richtung, bis es nicht mehr weiter nach unten geht. Man ist angekommen. Aber ob hinter einem nächsten Hügel ein Tal liegt, dass tiefer ist, bleibt unbekannt.

Prinzipiell folgt die Suche dem Gauß-Newton Algorithmus. Das Verfahren arbeitet iterativ und verbessert ausgehend von Startwerten die Lösung schrittweise. Die Schrittweite wird allerdings ähnlich wie bei Levenberg-Marquardt gesteuert.

Ein Minimum wird dann gut gefunden, wenn die Koeffizienten vor Termen stehen, die sich in ihrem Wesen bzw. Form stark voneinander unterscheiden.

Koeffizienten

In der Formel werden die Koeffizienten A1, A2, ... A16 fortlaufend angegeben. Bei A1 beginnend. Nur so viele wie benötigt werden.

Variablen

In der Formel werden die Variablen, die die Messwerte sind, i. Allg. mit x und y angegeben.

Input	Anwendung
äquidistant	Die Funktion kann auf äquidistante Eingangsdaten angewendet werden. Dann werden x und y in der Formel benutzt.
XY	Die Funktion kann auf XY-Eingangsdaten angewendet werden. Dann werden x und y in der Formel benutzt.
Matrix	Die Funktion kann auf segmentierte Daten angewendet werden. Dann werden x, y und z in der Formel benutzt. Wie sonst bei FAMOS üblich ist y die Amplitude, x und z spannen die Matrix auf und sind jeweils durch Startwert und Inkrement implizit gegeben. y=y(x,z).
Liste	Die Funktion kann auf segmentierte Daten angewendet werden, wobei jedes Segment als eigene Variable gedeutet wird. In der Formel werden dann x, y1, y2, .. y16 benutzt. Dabei ist x die allen Segmenten gemeinsame x-Koordinate. y1 sind die Werte des 1. Segmentes, y2 die des 2. etc.

Startwerte

Startwerte werden über den Parameter Limits übergeben.

Ausgehend von den Startwerten nähert sich die Funktion einem lokalen Minimum.

Der Anwender ermittelt vor Aufruf geeignete Startwerte.

Bei ungünstigen Startwerten konvergiert die Funktion nicht und findet kein Ergebnis.

In einigen Fällen brauchen keine Startwerte angegeben werden. Passende Startwerte werden dann automatisch gesucht. Die automatisch gefundenen Startwerte und damit das Ergebnis sind nicht immer gut. Das Resultat ist in jedem Fall zu überprüfen.

Bei Verwendung von sin() oder cos() bei äquidistanten Daten und Argumenten der Form "A1*x+A2" findet die Funktion automatisch Startwerte über eine intern ausgeführte FFT ohne Beachtung von Gewichtsfaktoren. Das führt i. Allg. aber nur bei konstanter Frequenz und dominanter Amplitude im Signal zum Erfolg.

Bei Formeln vom Typ "A1+A2*exp(A3*x)-y" kann sehr gut auf die Vorgabe von Startwerte verzichtet werden.

Werden Startwerte vorgegeben, müssen sie mindestens in einem sinnvollen Bereich liegen. Über- und Unterschreiten des Zahlenbereichs muss vermieden werden. Bei "exp(A1*x)-y" und x im Bereich 10 führt mit Startwert A1=100 zum Overflow der Formel und ihrer Ableitung nach A1. A1=-100 ergibt einen ständigen Underflow.

Für Koeffizienten, die linear in der Formel stehen, braucht niemals ein Startwert vorgegeben zu werden. Vor Beginn er Iteration werden für diese Koeffizienten ohnehin von allein passende Startwerte ermittelt.

Bei trigonometrischen Funktionen gibt es typisch eine richtig große Anzahl von lokalen Minima, die auch noch alle sehr dicht beieinander liegen. Auch bei guter Wahl der Startwerte kann das Verfahren leicht auf ein Minimum in der Nähe springen.

Formel

Die Formel kann neben den Rechenzeichen auch die Funktionen sin, cos, tan, exp, asin, acos, atan, ln, log, sqrt, abs benutzen.

Alle trigonometrischen Funktionen arbeiten im Bogenmaß. Falls eine Phase im Gradmaß gewünscht ist, z.B. "sin(A1*x+A2*PI/180)". PI und PI2 sind definiert.

Koeffizienten können passend skaliert werden, z.B. die Frequenz in einer sin() Funktion, "sin(2*PI*A1*x+A2)".

Das Verfahren beinhaltet, dass Ableitungen der Formel nach den Koeffizienten berechnet werden. Die Ableitung muss stetig sein. Damit können die Koeffizienten durchaus in einer Funktion wie exp() stehen, aber nicht in abs(). Funktionen wie abs() können an anderen Stellen der Formel benutzt werden wie z.B. abs(x).

Lineare Abhängigkeiten sind zu vermeiden wie z.B. ..+(A1-A2). Das führt i. Allg. zu einem Fehler während der Iteration oder auch zu einem Divergieren.

Linear eingesetzte Koeffizienten

Die Formel sei dann linear in einem Koeffizient, wenn die Ableitung nach dem Koeffizient von keinem linearen Koeffizienten abhängt.

Koeffizienten, die linear eingesetzt werden, werden vorteilhaft behandelt. So ist "A1*x+..." zu bevorzugen gegenüber "x/A1+..."

Ähnlich ist auch "A1*x+A2+..." gegenüber "A1*(x+A2)+..." zu bevorzugen. Beim ersten Ausdruck sind A1 und A2 linear eingesetzt. Beim zweiten Ausdruck sind es nicht beide.

So ist auch "A2*(A1+x)^2..." gegenüber "((A1+x)/A2)^2..." zu bevorzugen. Beim ersten Ausdruck ist A2 linear eingesetzt.

Auch in Argumenten von Funktionen ist der linear eingesetzte Koeffizient unbedingt zu bevorzugen, z.B. "exp(A1*x)" und "sin(A1*x+A2)" anstelle von "exp(x/A2)" und "sin(x/A1+A2)".

Eingangsdaten

Ausreichend viele unabhängige Eingangsdaten müssen zur Verfügung gestellt werden. Enthalten die Eingangsdaten nicht ausreichend viele unabhängige Informationen, führt es zu denselben Schwierigkeiten wie bei linear abhängigen Koeffizienten.

Limits

Für jeden in der Formel benutzten Koeffizient darf eine Unter- und eine Obergrenze angegeben werden.

Ist kein Startwert für einen Koeffizienten vorgegeben, versucht die Funktion, innnerhalb des Bereichs von Unter- bis Obergrenze einen passenden Startwert zu ermitteln.

Je nach Formel ist es sinnvoll, einen möglichst engen Bereich zu bilden, um die Suche nach Startwerten zu verkürzen.

Ist ein Startwert für einen Koeffizienten gegeben, wird dieser auch benutzt.

Unter- und Obergrenze werden während der Iteration nicht überprüft. Es sind damit keine wirklichen Nebenbedingungen der Minimierungsaufgabe.

Unter- und Obergrenze werden aber benutzt, um das Ergebnis zu prüfen. Liegt es außerhalb, wird das als Fehler gewertet.

Die Grenzen und Initialisierungswerte können auch durch eine Berechnung gegeben sein, solange sie auf eine Konstante führt, z.B. "A1 > 10-3; A1 < 10+3"

Deutung der ermittelten Koeffizienten

Das Resultat ist in jedem Fall zu überprüfen.

Können in einer Formel die Koeffizienten vertauscht werden, ohne dass sich der Inhalt ändert, können auch im Ergebnis die Koeffizienten vertauscht erscheinen. Z.B. in "exp(A1*x)+exp(A2*x)-y" kann das Ergebnis mal [-3,-7] sein, mal auch [-7,-3]. Solche Anordnungen können aber ohnehin problematisch in Bezug auf Konvergenz sein.

Bei Koeffizienten in trigonometrischen Funktionen können auch unerwartete Resultate auftreten: Z.B. bei der Formel "A1*tan(A2*x+A3)-y" können z.B. A1 und A2 beide negativ statt positiv werden, A3 kann um Vielfache von PI größer oder kleiner werden.

Bei Koeffizienten innerhalb von sin() und cos() wie z.B. in "A1*sin(A2*x+A3)-y" werden A1 und A2 bevorzugt positiv ermittelt, A3 bevorzugt im Bereich -PI..+PI.

Ergebnisse mit Wert 1e35 oder auch -1e35 sind möglicherweise auf den Wertebereich von FAMOS beschränkt. Eigentlich wäre ihr Betrag dann noch größer.

Geschwindigkeit

Zur Verbesserung der Ausführungsgeschwindigkeit bei großen Datenmengen kann mit einer vorher passend ausgewählten kleinen, aber repräsentativen Datenmenge ein Zwischenergebnis erzielt werden. Dieses wird als Startwert für die eigentliche große Approximationsaufgabe benutzt.

Bei großer Anzahl von Koeffizienten und ohne Vorgabe von Startwerten ist die interne Suche nach geeigneten Startwerten besonders rechenintensiv.

Die interne Suche nach Startwerten kostet Rechenzeit vor allem für Koeffizienten, die nichtlinear eingesetzt sind. Also in A1+A2*exp(A3*x)-y sind A1 und A2 linear verwendet, A3 ist nichtlinear eingesetzt und kostet Suchzeit.

Ggf. werden mehrere Prozessorkerne ausgenutzt.

Eine Datenmenge von 100000 Punkten ist für die Funktion schon groß.

Gewicht

Falls kein Gewicht angegeben ist, werden alle Messwerte gleich gewichtet.

Der Datensatz Gewicht enthält für jeden Messwert einen Gewichtsfaktor. 1.0 ist dabei neutral. Aber auch >1 und <1 sind möglich bis hin zu 0.

Der Datensatz Gewicht ist genauso lang wie die Eingangsdaten. Er ist nie segmentiert. Falls die Eingangsdaten segmentiert sind und eine Liste enthalten, ist der Datensatz Gewicht genauso lang wie ein Segment.

Ist der Datensatz Gewicht leer, ist kein Gewicht vorgegeben.

Allgemein

Bei Aufruf mit falschen Koeffizienten, Syntaxfehlern in der Formel oder nicht ausreichend Speicher wird stets mit der üblichen Fehlermeldung abgebrochen.

Gibt es Probleme bei der Konvergenz, wird die Fehlerbehandlung wirksam.

Bei zu vielen Wertebereichsüberschreitungen, z.B. exp(1000) oder Division durch null, entstehen Problem bei der Konvergenz.

Ein (vom Betrag her) großer x-Offset x0 kann bei äquidistanten Eingangsdaten bei entsprechend kleinem Abtastschritt dx zu numerischen Problemen führen. Soll z.B. die Abklingkonstante in einer Abklingfunktion ermittelt werden, muss in der Formel mit x gerechnet werden. Die gewünschte Abklingkonstante hängt aber nicht vom x-Offset ab. Hier sollte vor Ausführung der x-Offset der Eingangsdaten auf null gesetzt werden.

Die Funktion arbeitet mit der Genauigkeit von 64 bit reellen Zahlen, was die Anwendbarkeit bei extremer Numerik einschränkt.

Wird mit Timeout gearbeitet, so darf er nicht zu knapp bemessen sein. Die Rechenzeit hängt von vielen Faktoren ab wie z.B. den aktuellen Zahlenwerten, dem Prozessor und der Prozessorauslastung.

Beispiele:

Äquidistante Eingangsdaten ohne Vorgabe von Startwerten, Abklingfunktion

```
t = ramp(0,0.01, 10)
data = 4 + -3 * exp ( t * -20 ) ; test data
A = ApproNonLin(data,"A1 + A2 * exp ( x * A3 ) - y")
A1 = A[1]
A2 = A[2]
A3 = A[3]
```

Äquidistante Eingangsdaten ohne Vorgabe von Startwerten, gedämpfte Schwingung, vorteilhafte Gestaltung der Koeffizienten

```
t = ramp(0,0.0001, 2000)
A1 = 4 ;offset [V]
A2 = 3 ; amplitude[V]
A3 = -1/0.05
A4 = 120 ; f[Hz]
A5 = 70 ; phase [degrees]
y = A1 + A2 * exp ( t * A3 ) * sin ( 2 * PI * A4 * t - A5 * PI/180 ) ; test data
```

```
A = ApproNonLin(y, "A1 + A2 * exp ( x * A3 ) * sin ( 2 * PI * A4 * x - A5 * PI/180 )  - y" )
Tau=-1/A[3] ; time constant
```

Äquidistante Eingangsdaten mit Startwerten, Polynom

```
t = ramp(0,0.001, 50000)
test = 4 * t*t + 10 ; test data
start="init A1 = 3, init A2 = 7"
A = ApproNonLin(test,"A1 * x*x + A2 - y",0,0,start)
```

Polynom 2. Grades aus äquidistanten Daten

```
x = ramp ( 0.01, 0.001, 1000 )
y = 3 + 8 * x - 5 * x^2
A = ApproNonLin ( y, "A1 + A2 * x + A3 * x^2 - y" )
```

Polynom 3. Grades aus XY Daten

```
x = 0.001 * random ( 1000, 2, 0, 0, 8 )
y = 3 + 8 * x - 5 * x^2 + 2 * x^3
input_xy = xyof ( X, Y )
A = ApproNonLin ( input_xy, "A1 + A2 * x + A3 * x^2 + A4 * x^3 - y" )
```

Fit eines 2-dimensionalen Polynoms 2. Grades z = f(x,y)

z="A1+A2*x+A3*x^2 + A4*y+A5*x*y+A6*x^2*y + A7*y^2+A8*x*y^2+A9*x^2*y^2

```
Ai=[2,3,4, -2,-3,-4,1.5,1.7,-1.3]
data=leng(0,300)
setseglen(data, 100)
x=round(ramp(-4.9,1,200),10)/200
y=mod(ramp(0,1,200),10)/10
z=(Ai[1]+x*(Ai[2]+x*Ai[3])) + y*((Ai[4]+x*(Ai[5]+x*Ai[6])) + y*(Ai[7]+x*(Ai[8]+x*Ai[9])))
data[1]=x
data[2]=y
data[3]=z
A = ApproNonLin(data,"(A1+y1*(A2+y1*A3)) + y2*((A4+y1*(A5+y1*A6)) + y2*(A7+y1*(A8+y1*A9))) - y3")
```

Dichte Normalverteilung

```
x = ramp(0,0.00001, 10000)
A1 = 10 ; amplitude
A2 = 0.03 ; center
A3 = 0.01 ; sigma
y = A1 * exp ( -( (x-A2)^2 / (2*A3^2) ) )
A = ApproNonLin(y,"A1*exp(-((x-A2)^2/(2*A3^2)))-y")
center = A[2]
sigma = A[3]
```

Äquidistante Eingangsdaten ohne Startwerte, aber mit Gewichtung, Gerade

```
t = ramp(0,0.001, 50000)
test = 4 * t + 10 ; test data
Weight=test*0+1
Weight[1] = 0.1
A = ApproNonLin(test,"A1 * x + A2 - y",0,0,"", Weight)
```

XY Eingangsdaten mit Startwerten, Kreis

```
t = ramp(0,0.001, 1000)*30
A1 = 3
A2 = 10
A3 = 30
X = A1 + A3 * sin(t) + 1*random ( 1000, 2, 0, 0, 0 )
Y = A2 + A3 * cos(t)+ 1*random ( 1000, 2, 0, 0, 0 )
input_xy = xyof ( X, Y ) ; test data
start = "init A1 = 0, init A2 = 0"
A = ApproNonLin ( input_xy, "sqrt ( (X-A1)^2+(Y-A2)^2 )- A3",0,0, start )
```

XY Eingangsdaten mit Wertebereich und Startwert, Kreis

```
t = ramp(0,0.001, 1000)*30
A1 = 3
A2 = 10
A3 = 30
X = A1 + A3 * sin(t)
Y = A2 + A3 * cos(t)
input_xy = xyof ( X, Y ) ; test data
limits = "init A1 = 0, A1 < 100, A2 < 100, A2 >= 0"
```

```
A = ApproNonLin ( input_xy, "sqrt ( (X-A1)^2+(Y-A2)^2 )- A3",0,0, limits )
```

Segmentierte Eingangsdaten, gekrümmte Fläche

```
A1 = 4
A2 = -0.1
A3 = 0.002
A4 = -0.3
A5 = 0.2
y=leng(0,1000) ; test data
setseglen(y, 100)
x=ramp(0,1,100)
for i = 1 to 10
   z=i-1
   y[i] = A1 + A2 * x + A3 * x * x + A4 * z + A5 * z * z
end
formel = "A1 + A2 * x + A3 * x * x + A4 * z + A5 * z * z - y"
A = ApproNonLin(y,formel)
```

Viele Eingangskanäle y1 bis y4

```
A1 = 3
A2 = -0.3
A3 = 0.2
A4 = -0.3
x=ramp(0,1,100)
y1 = 0.0003 * x * x - 0.03*x+1 ; test data
y2 = 3 * sin ( x * 0.1 )
y3 = 5 * exp ( -0.03 * x )
y4 = A1 * y1 + A2 * y2 + A3 * y3 + A4
y=leng(0,400)
setseglen(y, 100)
y[1] = y1
y[2] = y2
y[3] = y3
y[4] = y4
A = ApproNonLin(y, "A1 * y1 + A2 * y2 + A3 * y3 + A4 - y4")
```

Abklingfunktion ohne Offset. Gleicht besser aus als eRegr().

```
t = ramp(0,0.001, 1000)
data = 5 * exp ( t * -3 ) + 0.1* random ( 1000, 2, 0, 0, 13 )
A = ApproNonLin(data,"A1 * exp ( A2 * x ) - y")
data_A = A[1] * exp ( A[2] * t )
data_B = eRegr( data )
```

Fit an eine Ebene: y = A1 + A2 * x1 + A3 * x2. Gegeben sind Tripel aus (x1,x2,y), die bei Darstellung von y über x1 und x2 sich in einer Ebene befinden sollen.

```
; test data
A1 = 7
A2 = 3
A3 = 0.5
x1 = 1 + 5 * sin ( ramp(0,1,100) * 0.2 ) * ramp(3,1,100)/100 ; x direction
x2 = 2 + 3 * cos ( ramp(0,1,100) * 0.2 + 0.1 ) * ramp(5,1,100)/100 ; z direction
y = A1 + A2 * x1 + A3 * x2 + 0.0 * random ( 100, 2, 0 , 0, 3 ) ; amplitudes
;Berechnung
data = MatrixInit(leng?(y),3)
data[1] = x1
data[2] = x2
data[3] = y
A = ApproNonLin( data, "A1 + A2 * y1 + A3 * y2 - y3")
```

Siehe auch:

Appro, Regr, eRegr, Poly

ASCII

Einstellen des ASCII-Formats zum anschließenden Laden von Dateien mit dem Befehl LOAD

Deklaration:

```
ASCII EwVorspann EwTyp EwTeiler EwAnzahl EwOption Vorspann-Zeichenfolge
```

Parameter:

EwVorspann	Länge des Vorspanns, in Werten oder Bytes angegeben (abhängig vom 2. Parameter [Typ])
EwTyp	Gibt an, wie der Parameter [Vorspann] zu interpretieren ist.
	0 : Der Vorspann gibt die Zahl der zu überlesenden Bytes an.
	1 : Der Vorspann gibt die Zahl der zu überlesenden Zahlenwerte an.
EwTeiler	Gibt an, jeder wievielte Wert aus den Zahlenwerten der Datei zu lesen ist.
EwAnzahl	Gibt die Anzahl der einzulesenden Zahlenwerte an. Eine Anzahl von Null bedeutet, dass alle Werte bis zum Ende der Datei zu lesen sind.
EwOption	Option
	0 : Zahlen werden mit Dezimalpunkt erwartet. Der erzeugte Datensatz hat das Datenformat "Float" (4 Byte reell).
	1 : Zahlen werden mit Dezimalkomma erwartet. Der erzeugte Datensatz hat das Datenformat "Float" (4 Byte reell).
	10 : Zahlen werden mit Dezimalpunkt erwartet. Der erzeugte Datensatz hat das Datenformat "Double" (8 Byte reell).
	11 : Zahlen werden mit Dezimalkomma erwartet. Der erzeugte Datensatz hat das Datenformat "Double" (8 Byte reell).
Vorspann-Zeichenfolge	Eine Zeichenfolge, nach der in der Datei gesucht wird, bevor mit dem Lesen des Vorspanns und der Zahlenwerte begonnen wird. Die Zeichenfolge darf keine Leerzeichen enthalten. Wird keine Zeichenfolge als Parameter übergeben, wird gleich am Anfang der Datei mit dem Lesen des Vorspanns begonnen.

Beschreibung:

Das Dateiformat für den Befehl LOAD wird auf ein ASCII-Format gestellt.

Werden dem Befehl ASCII keine Parameter hinzugefügt, wird das aktuell definierte ASCII-Format gewählt.

Wenn Parameter angegeben werden, wird ein neues ASCII-Format komplett definiert.

- Wenn Parameter angegeben werden, müssen alle Parameter angegeben werden. Jeder Parameter, mit Ausnahme der Zeichenfolge, darf ein fester Zahlenwert oder eine Variable vom Typ Einzelwert (EW) sein. Die Benutzung von Variablen als Parameter ermöglicht das Lesen von variablen Dateiformaten.
- Der Befehl ASCII stellt eine Automatisierung des Dialogfeldes 'Optionen'/ 'Datei laden'/ 'ASCII' dar. Dieses Dialogfeld erreichen Sie beispielsweise, wenn Sie im 'Datei laden'-Dialog bei eingestelltem ASCII-Format die Schaltfläche 'Optionen' betätigen. Eine genauere Erläuterung und weitere komplette Beispiele finden Sie im Benutzerhandbuch, Kapitel 'Dateiverwaltung'.
- Eine Alternative zum ASCII-Import mit der Befehlskombination ASCII/LOAD (besonders für komplexere Formate) ist die Verwendung des ASCII-Import-Assistenten ('Datei laden'-Dialog, Format "ASCII/Excel Import...", Schaltfläche 'Optionen') oder des imc-Datei-Assistenten zur Erstellung eines Importfilters und die Anwendung des Filters mit dem Befehl FASLOAD oder der Funktion FileOpenFAS().
- Zum zeilenweisen Lesen von Textdateien können Sie die Funktion FileOpenASCII() verwenden.
- Multithreading: Der Befehl wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
ASCII
LOAD DATA.ASC
```

Das zuletzt eingestellte ASCII-Format wird wieder eingestellt, nachdem zwischendurch einige Dateien z.B. im imc/FAMOS- Format geladen wurden. Danach wird die ASCII-Datei DATA.ASC geladen.

```
ASCII 0 0 2 0 0 test
LOAD DATA.ASC
```

Aus einer Datei, die eine Tabelle von ASCII-Werten enthält, wird eine Spalte gelesen. Die Überschrift der Tabelle heißt "test". Die erste der zwei Spalten wird gelesen.

```
Vleng = 10
Data = 100
ASCII Vleng 1 1 Data 11
LOAD DATA.ASC
```

Aus einer ASCII-Datei werden 100 Werte gelesen. Es wird ein Vorspann mit einer Länge von 10 Bytes am Anfang der Datei übersprungen. Die Zahlen werden mit Dezimalkomma gelesen. Der Ergebnisdatensatz hat das Datenformat 8 Byte reell.

Siehe auch:

LOAD, BINARY, FileOpenASCII, FileOpenFAS

ASCOPTION

Spezifizierung des Dateiformates für den Befehl ASCSAVE (Speichern im ASCII-Format).
Statt den Befehlen ASCOPTION und ASCSAVE können Sie auch die Funktionen FileSave() und FileOpenASCII2() verwenden. Diese erlauben einen weitaus flexibleren Dateiexport, wobei das Format über eine ASCII-Exportvorlage spezifiziert wird und können auch verwendet werden, um mehrere Datensätze in mehrspaltigen ASCII-Dateien zu speichern.

Deklaration:

```
ASCOPTION EwZahlenformat EwDezimaltrenner EwGenauigkeit EwBreite EwMitKopf EwMitXWerten
```

Parameter:

EwZahlenformat	Zahlenformat
	0 : Alle Werte werden in Exponentialdarstellung abgespeichert
	1 : Alle Werte werden in Festkomma-Darstellung abgespeichert.
	2 : Abhängig vom Wert wird die kürzeste Schreibweise gewählt.
EwDezimaltrenner	Dezimaltrenner
	0 : Dezimalpunkt
	1 : Dezimalkomma
EwGenauigkeit	Bei Zahlenformat 0 und 1 die Zahl der Nachkommastellen. Bei Zahlenformat 2 (automatisch) die Gesamtzahl der signifikanten Stellen. Wertebereich (0..7).
EwBreite	Minimale Ausgabebreite, falls die Breite des Wertes kleiner als diese Angabe ist, wird links mit Leerzeichen aufgefüllt. Wertebereich (0..99)
EwMitKopf	Gibt an, ob ein standardisierter Dateikopf vorangestellt werden soll.
	0 : Kein Dateikopf.
	1 : Ein standardisierter Kopf, der wichtige Kennwerte der Variable enthält, wird am Anfang der Datei eingetragen.
EwMitXWerten	Gibt an, ob eine zweispaltige Ausgabe inklusive x-Koordinaten erfolgen soll.
	0 : Die Y-Werte werden einspaltig untereinander untereinander geschrieben
	1 : In jeder Zeile der Datei steht zuerst der X-Wert, dann ein Tabulator und dann der zugehörige Y-Wert.

Beschreibung:

Das Dateiformat zum Sichern im ASCII-Format (Befehl ASCSAVE) wird spezifiziert.

Der Befehl ASCOPTION stellt eine Automatisierung des Dialogfeldes 'Extras'/'Optionen'/'Datei speichern'/ 'ASCII (3.2 kompatibel)' dar. Dieses können Sie beispielsweise auch aufrufen, wenn Sie im 'Datei speichern'-Dialog bei eingestelltem ASCII-Format die Schaltfläche 'Optionen' betätigen. Eine genauere Erläuterung und Beispiele finden Sie im Bedienerhandbuch im Kapitel 'Dateiverwaltung'. Dort finden sie auch eine Beschreibung des Headers.

Die Parameter können feste Zahlen oder Variablen vom Typ Einzelwert sein

Multithreading: Der Befehl wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

Eine Datei im imc/FAMOS-Format wird geladen und als zweispaltige ASCII-Datei wieder gespeichert.

```
FAMOS
LOAD sintest1.dat data
ASCOPTION 1 0 5 10 0 1
ASCSAVE data "c:\my data\sintest1.txt"
```

Siehe auch:

ASCSAVE, FileSave, FileOpenASCII2, FileLineWrite

ASCSAVE

Sichern von Daten im ASCII-Format
Alternativer Name: **ASCSICHERN**

Statt den Befehlen ASCOPTION und ASCSAVE können Sie auch die Funktionen FileSave() und FileOpenASCII2() verwenden. Diese erlauben einen weitaus flexibleren Dateiexport, wobei das Format über eine ASCII-Exportvorlage spezifiziert wird und können auch verwendet werden, um mehrere Datensätze in mehrspaltigen ASCII-Dateien zu speichern.

Deklaration:

```
ASCSAVE Variablenname Dateiname
```

Parameter:

Variablenname	Variable, die gespeichert werden soll.
Dateiname	Dateiname (oder auch kompletter Pfad), unter dem die Variable gesichert werden soll.

Beschreibung:

Die als Parameter angegebene Variable wird als ASCII-Datei gesichert. Die Variable darf ein Einzelwert oder ein normaler Datensatz sein. Ist kein Dateiname angegeben, wird die Variable unter ihrem Namen in dem momentan für das Sichern von Variablen eingestellten Pfad gespeichert. Als Dateinamen-Erweiterung wird ".ASC" benutzt.

Wenn der Dateiname angegeben ist, wird die Variable unter diesem Namen abgespeichert. Falls es sich um eine komplette Pfadangabe handelt, wird dieser Pfad benutzt.

Der Dateiname darf auch in Anführungszeichen angegeben werden. Dies ist z.B. dann notwendig, wenn der Pfad Leerzeichen enthält.

Die Konfiguration des Dateiformats erfolgt gemäß der Einstellungen unter "Optionen"/"Datei Speichern/Export"/"ASCII (FAMOS 3.2 kompatibel)" oder durch den Befehl ASCOPTION.

Mit diesem Befehl können keine kompletten Datengruppen gesichert werden. Sichern Sie dann die einzelnen Kanäle einzeln. Das Speichern von Texten ist ebenfalls nicht möglich, verwenden Sie dafür die Funktion FileSave() mit der Formatkennung "imc/Text" oder die Funktion FileLineWrite().

Beispiele:

```
Data = Int(Data)
ASCSAVE Data
```

Die Variable Data wird integriert und in einer Datei "DATA.ASC" gesichert.

```
Hist = Histo(Data1, 10, 6)
ASCSAVE Hist c:\data\histogr\hist001
Hist = Histo(Data2, 10, 6)
ASCSAVE Hist c:\data\histogr\hist002
```

Von den Datensätzen Data1 und Data2 wird je ein Histogramm berechnet und diese im angegebenen Verzeichnis unter den Namen 'HIST001.ASC" und 'HIST002.ASC' im ASCII-Format gesichert.

```
ASCSAVE Hist "c:\data\histogr\My Histogram"
```

Der Dateiname enthält Leerzeichen und muss deshalb in Anführungszeichen angegeben werden.

Siehe auch:

ASCOPTION, FileSave, FileOpenASCII2, FileLineWrite

asin

Arkussinus, Umkehrfunktion Sinus

Deklaration:

```
asin ( Daten ) -> Winkel
```

Parameter:

Daten	Daten. Erlaubte Typen: [ND],[XY].
Winkel	Arkussinus des Parameters (Winkel in Bogenmaß)

Beschreibung:

Es wird die arcsin-Funktion gebildet.

Die Funktion liefert einen Winkel im Bogenmaß ohne Einheit zurück.

Die x-Koordinate(n) von Ergebnis und Parameter sind gleich.

Anmerkungen

- Das Argument sollte keine Einheit haben, wenn doch, wird eine Warnung generiert, die Einheit wird unverändert übernommen
- Der zulässige Wertebereich des Parameters liegt zwischen -1 und +1.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

asin(0)= 0, asin(1)= PI/2, asin(-1)= -PI/2)

```
pihalf = asin(1)
```

Aus einem sinusförmigen Datensatz wird ein zackiger dreieckiger Datensatz:

```
NDspikes = asin(NDsine)
```

Mit der vordefinierten Konstante PI oder INDEGR wird der zurückgegebene Winkel vom Bogenmaß in das Gradmaß konvertiert:

```
value_90 = asin(1.0) * 180 '°' / PI
value_90 = asin(1.0) * INDEGR
```

Siehe auch:

sin, acos, atan2

atan

Arkustangens, Umkehrfunktion Tangens

Deklaration:

```
atan ( Daten ) -> Winkel
```

Parameter:

Daten	Daten. Erlaubte Typen: [ND],[XY].
Winkel	Arkustangens des Parameters (Winkel in Bogenmaß)

Beschreibung:

Die Funktion ist zur Kompatibilität noch enthalten. Im Allgemeinen ist die Funktion atan2() besser geeignet.

Es wird die arctan-Funktion gebildet.

Die Funktion liefert einen Winkel im Bogenmaß ohne Einheit zurück.

Die x-Koordinate(n) von Ergebnis und Parameter sind gleich.

Anmerkungen

- Das Argument sollte keine Einheit haben, wenn doch, wird eine Warnung generiert, die Einheit wird unverändert übernommen
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

atan(0)= 0, atan(1)= P1/4, atan(-1)= -PI(4))

```
pi025 = atan(1)
```

Korrektur von Daten über die Kennlinie des Tangens, z.B. bei Sättigungseffekten:

```
NDcorr = atan(NDdata)
```

Mit der vordefinierten Konstante INDEGR wird der zurückgegebene Winkel vom Bogenmaß in das Gradmaß konvertiert.

```
value_45 = atan(1.0) * INDEGR
```

Siehe auch:

tan, atan2, asin, acos

atan2

Arkustangens, Umkehrfunktion Tangens

Deklaration:

```
atan2 ( y, x ) -> Winkel
```

Parameter:

y	y-Koordinate(n) in der kartesischen Ebene. Erlaubte Typen: [ND].
x	x-Koordinate(n) in der kartesischen Ebene. Erlaubte Typen: [ND].
Winkel	Arkustangens des Parameters (Winkel in Bogenmaß)

Beschreibung:

Im Gegensatz zur atan() Funktion liefert diese Funktion den Winkel im Bereich von -PI bis +PI zurück.

Die Vorzeichen von [x] und [y] werden verwendet, um den korrekten Quadranten für das Ergebnis zu bestimmen.

Die Funktion ist definiert für jeden Punkt außer dem Ursprung (0,0). Die Einheit des Ergebnisses ist in Bogenmaß ("Rad"). Zur Umrechnung in "Grad" siehe untenstehendes Beispiel.

Falls einer der Parameter ein Einzelwert ist, werden alle Werte des anderen Parameters mit diesem Wert verrechnet. Falls beide Parametersätze mehr als einen Punkt besitzen, wird Wert für Wert der Arcustangens berechnet, bis das Ende des kürzeren Datensatzes erreicht ist.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

```
RectAngle = atan2(1, 0) * INDEGR   ;liefert 90 Grad;
Angle_Minus90 = atan2(-1, 0) * INDEGR  ;liefert -90 Grad
Angle_180 = atan2(0, -1) * INDEGR  ;liefert 180 Grad
```

2 Wege, um die Phasen für einen komplexen Datensatz in kartesischer Darstellung zu ermitteln:

```
phase1 = atan2(CplxRect.I, CplxRect.R)
phase2 = Cmp2(Pol(CplxRect)) * INDEGR
```

Siehe auch:

tan, atan2, asin, acos

BEGIN_PARALLEL

Verfügbar ab: Professional Edition

Eröffnet einen Code-Block mit parallel auszuführenden Sequenzfunktionen.

Deklaration:

```
BEGIN_PARALLEL
```

Beschreibung:

Innerhalb eines Parallel-Blockes (also alle Zeilen zwischen einem BEGIN_PARALLEL und END_PARALLEL-Schlüsselwort) werden alle Anweisungen parallel ausgeführt. Als Anweisungen in einem Parallel-Block sind nur direkte Aufrufe von Sequenzfunktionen erlaubt. Die Sequenzfunktionen werden jeweils in einem separaten Ausführungskontext (Thread) gestartet, dann wird sofort zur nächsten Zeile gesprungen, ohne das Ende der Ausführung abzuwarten.

Das Ende eines solchen Blockes wird durch das END_PARALLEL-Schlüsselwort markiert. Die Sequenz wartet bei END_PARALLEL, bis alle zuvor angestoßenen Sequenzfunktionen beendet sind und setzt dann erst die Ausführung fort.

In FAMOS sind immer mindestens 2 Threads aktiv. Der Haupt-Thread kümmert sich um die Anzeige der Programmoberfläche und die Verarbeitung von Benutzereingaben. Standardmäßig werden alle Sequenz-Anweisungen in FAMOS in einem Standard-Ausführungs-Thread abgearbeitet, der ebenfalls immer aktiv ist und gegebenenfalls auf Aufgaben wartet. Der aktuelle Ausführungs-Thread einer Sequenzfunktion in einem BEGIN_PARALLEL-Block ist temporär und wird dem Ende der Abarbeitung der Sequenzfunktion beendet.

Das Betriebssystem kümmert sich darum, die parallelen Ausführungen optimal auf die vorhandenen Prozessorkerne zu verteilen. Im Unterschied zur sequentiellen Ausführung, die nur einen Prozessorkern verwendet, ist damit eine deutliche Beschleunigung der Sequenzabarbeitung möglich.

Anwendungs-Hinweise

- Nur wirklich zeitaufwendige Berechnungen und Auswertungen sollten parallelisiert werden. Bei kurzen Ausführungszeiten ist der Verlust durch den zusätzlichen Verwaltungsaufwand oft größer als der Zeitgewinn durch die parallele Ausführung.
- Bitte achten Sie darauf, dass nur Aufgaben parallelisiert werden, die tatsächlich auch voneinander unabhängig sind. Die Reihenfolge der Ausführung ist zufällig, Ergebnisse einer parallelen Sequenzfunktion (nicht nur explizite Ergebnis-Variablen, sondern auch z.B. neu erzeugte Dateien) dürfen also keinesfalls als Eingangsparameter oder implizite Voraussetzung für eine andere parallel ausgeführte Sequenzfunktion verwendet werden. Hier hilft die Überlegung: Könnte ich bei einer sequentiellen Ausführung die Reihenfolge der Aufrufe im PARALLEL-Block beliebig verändern? Wenn nein, kann nicht parallelisiert werden.
- In PARALLEL-Blöcken aufgerufene Sequenzfunktionen sollten möglichst nur lokale Variablen verwendet werden, da globale Variablen durch nebenläufige Zugriffe unerwartet verändert werden könnten. Beispiel: die selbe Sequenzfunktion wird mit verschiedenen Parametern 2x in einem Parallel-Block aufgerufen. In der Sequenz wird in einer FOR-Schleife eine nicht-lokale Schleifenvariable i verwendet. Beide Ausführungen greifen parallel auf diese Variable und ändern diese. Wie oft die Schleife in jeder Sequenz dann tatsächlich durchlaufen wird, ist damit zufällig.

Geltungsbereich von Voreinstellungen und Initialwerten

- Einige Funktionen legen Voreinstellungen oder Initialwerte fest, die dann von nachfolgenden Funktionsaufrufen verwendet werden. Diese Einstellungen können global oder Thread-lokal gültig sein.
- Global heißt hier, dass ein einziges globales Gedächtnis vorhanden ist und dieses Gedächtnis in allen Ausführungs-Threads gültig ist und benutzt wird. Beispiele sind z.B. Funktionen wie SetOption(), SelMeasListSetName(), FFTOPTION und alle Funktionen zur Panel- und Reportfernsteuerung.
- Thread-Lokal bedeutet, dass das interne Gedächtnis separat für jeden Ausführungs-Thread verwaltet wird, es muss also auch ggf. in jeder parallel aufgerufenen Sequenzfunktion neu initialisiert werden. Beispiele sind z.B. Stat(), FileOpenDSF(), SelUseMeasurement(), TerzI(), CwSelectWindow(), OtrTachoMode() und die Funktionen zur Excel- und Powerpoint-Fernsteuerung. Das konkrete Verhalten ist in der jeweiligen Funktions-Hilfe beschrieben.

Einschränkungen

- Die maximale Zahl von parallel auszuführenden Sequenzfunktionen in einem PARALLEL-Block ist **64**. Eine praktisch sinnvolle Obergrenze ist die Zahl der virtuellen Prozessorkerne im System. Diese kann mit der Funktion GetSystemInfo() ermittelt werden.
- PARALLEL-Blöcke können nicht geschachtelt werden. Innerhalb eines PARALLEL-Blockes werden z.B. BEGIN_PARALLEL-Aufrufe innerhalb der aufgerufenen Sequenzfunktion ignoriert.
- Einige Funktionen können innerhalb von BEGIN_PARALLEL-Blöcken nicht verwendet werden. Beispiele dafür sind z.B. die Funktionen des Datenbank-Kits oder des ASAM-ODS-Kits. Andere Funktionen können zwar an beliebiger Stelle aufgerufen werden, werden aber intern in den FAMOS-Haupt-Thread geschoben und von dort ausgeführt, z.B. die Funktionen des R- und des Pythons-Kits. Solche Einschränkungen sind in der jeweiligen Funktionshilfe beschrieben.
- PARALLEL-Blöcke sind innerhalb von Zeitgeber-Ereignis-Sequenzen (Panele) nicht erlaubt.
- PARALLEL-Blöcke werden bei der Ausführung im Debug-Modus (z.B. "Einzelschritt", "Ausführen (Haltepunkte beachten)" ignoriert.
- Beim Unterbrechen/Abbrechen der Ausführung (z.B. weil ein Fehler aufgetreten ist), wird gewartet, bis alle aktiven parallelen Ausführungen beendet sind.

Verriegelung von Variablen-Zugriffen

Durch PARALLEL-Blöcke ist die quasi-gleichzeitige Ausführung von Codezeilen möglich. Wenn eine Codezeile in einer parallelen Sequenzfunktion ausgeführt wird, so werden die verwendeten Variablen (z.B. Funktionsparameter) für die Zeit der Ausführung einer Teilfunktion in dieser Zeile intern exklusiv reserviert. Wenn nun parallel eine andere Sequenzzeile ausgeführt werden soll, in der bereits reservierte Variablen verwendet werden sollen, so wird mit der Abarbeitung gewartet, bis alle benötigten Variablen wieder verfügbar sind. Dadurch wird verhindert, dass während der Abarbeitung eine verwendete Variable von anderer Stelle unerwartet verändert wird. Die Freigabe der Variablen erfolgt normalerweise nach kompletter Abarbeitung der Sequenzzeile.

Eine Ausnahme gibt es beim Aufruf von Sequenzfunktionen. Da beim Einsprung in die Sequenzfunktion i.Allg. eine lokale Kopie der Parameter angelegt wird, kann die Freigabe der reservierten Parameter bereits mit Beginn der Abarbeitung der Funktion erfolgen. Dadurch wird es auch möglich, dass innerhalb eines BEGIN_PARALLEL-Blocks Sequenzfunktionen, die auf dieselbe(n) Variable(n) arbeiten, auch tatsächlich parallel ausgeführt werden können. Aber Achtung: wenn eine Datengruppe als Parameter übergeben wird und der Datentyp des entsprechenden Sequenzfunktionsparameters nicht als Datengruppe definiert ist, so wird intern eine Schleife über alle Kanäle der Gruppe ausgeführt. Die Freigabe des Parameters kann dann erst mit dem Eintritt in den letzten Schleifendurchlauf erfolgen.

Beim Duplizieren eines Parameters für die lokale Kopie gibt es allerdings die Besonderheit, dass Original und Kopie zunächst auf denselben internen Speicherbereich für die eigentlichen Daten verweisen. Diese in FAMOS übliche Optimierung (die z.B. auch bei einer einfachen Zuweisung "a = b" verwendet wird) spart natürlich besonders bei großen Datensätzen Speicherplatz und Zeit beim Duplizieren, kann aber bei parallelen Aufrufen von Nachteil sein und zu unerwünschten Verzögerungen führen! Wenn parallele Ausführungen gleichzeitig auf die Daten der lokalen Kopien desselben Parameters zugreifen wollen, muss wegen des gemeinsamen internen Speicherbereichs eine entsprechende Verriegelung wirksam werden. Wenn also ein Parameter in einer länger arbeitenden Funktion verwendet wird, müssen parallele Zugriffe auf die Daten des selben Parameters in anderen Sequenzfunktionen so lange warten, bis die aktuelle Funktion ihre Arbeit beendet hat.

Wenn also die Ausführungszeit eines PARALLEL-Blocks, in dem derselbe Datensatz wiederholt als Parameter für die aufgerufenen Sequenzfunktionen verwendet wird, nicht Ihren Erwartungen entspricht, sollten Sie versuchen, statt dem Original-Parameter eine temporäre Kopie mit eigenem (ungeteiltem) Speicherbereich zu übergeben. Das Erzwingen einer echten Kopie mit eigenem Speicherbereich kann am einfachsten durch Addieren einer 0 geschehen. Dieses Vorgehen ist in Beispiel 3 demonstriert.

Beispiele:

```
Drei Datensätze werden einer Auswertung unterzogen, die als Ergebnis eine einzelne Kennzahl liefert. Anschließend wird der Mittelwert der Ergebnisse berechnet.
BEGIN_PARALLEL
   t1 = !CalcSpecificValue(channel_1)
   t2 = !CalcSpecificValue(channel_2)
   t3 = !CalcSpecificValue(channel_3)
END_PARALLEL
t = (t1+t2+t3)/3
```

Ein Textfeld enthält eine größere Anzahl von Dateinamen. Die entsprechenden Dateien sollen geladen und ausgewertet werden.

```
allFiles = FsGetFileNames("c:\data", "*.raw", 0, 1, 0)
FOREACH ELEMENT file in allFiles
   !WorkWithFile(file) ; lädt eine Datei und führt die Auswertung durch
END
```

Dieser Ablauf soll durch Parallelisierung der Ausführung beschleunigt werden:

```
allFiles = FsGetFileNames("c:\data", "*.raw", 0, 1, 0)
count = TxArrayGetSize(allFiles)
chunksize = floor(count/4)
BEGIN_PARALLEL
   !WorkWithFiles(TxArrayPart(allFiles, 1,             chunksize))
   !WorkWithFiles(TxArrayPart(allFiles, 1+ chunksize,  chunksize))
```

```
   !WorkWithFiles(TxArrayPart(allFiles, 1+ chunksize*2, chunksize))
   !WorkWithFiles(TxArrayPart(allFiles, 1+ chunksize*3, count-3*chunksize))
END_PARALLEL
```

Die Sequenzfunktion !WorkWithFiles ist dabei wie folgt definiert:

```
; Deklaration: !WorkWithFiles(Par1 [Datentyp: Textfeld])
FOREACH ELEMENT file in Par1
   !WorkWithFile(file)
END
```

```
testdata= ...
f = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 500, 1000, 5000, 10000]* 1e-3
```

Ein langer Datensatz soll nacheinander einer Hochpassfilterung mit verschiedenen Grenzfrequenzen unterworfen werden.

```
FOR i = 1 TO lang?(f)
   Name = "Filt "+ TForm(i, "")
   <Name> = FiltHP(testdata, 0, f[i])
END
```

Dieser Ablauf soll durch Parallelisierung der Ausführung beschleunigt werden:

```
count = lang?(f)
chunksize = Floor(count/4)
BEGIN_PARALLEL
   !DoHighPass_Chunk(testdata+0, f+0, 1, chunksize)
   !DoHighPass_Chunk(testdata+0, f+0, 1+  chunksize, chunksize)
   !DoHighPass_Chunk(testdata+0, f+0, 1+2*chunksize, chunksize)
   !DoHighPass_Chunk(testdata+0, f+0, 1+3*chunksize, count-3*chunksize)
END_PARALLEL
```

Die Sequenzfunktion !DoHighPass_Chunk ist dabei wie folgt definiert:

```
;   Deklaration:
;   !DoHighPass_Chunk (data [Datentyp:Normal], freq, index, count  [Einzelwerte])
;   (Option "Neu angelegte Variablen sind standardmäßig lokal" ist ausgeschaltet.)
LOCAL i
FOR i = index TO index+count-1
   LOCAL Name = "Filt_"+ TForm(i, "")
   <Name> = FiltHP(data, 0, 0, freq[i])
END
```

Die Übergabe von "testdata+0" statt "testdata" an die Sequenzfunktion sorgt dafür, dass jeder Aufruf mit einer kompletten, eigenständigen Kopie von "testdata" ausgeführt wird. Dasselbe gilt für den Parameter "f". Die FiltHP()-Aufrufe in der Sequenzfunktion sind damit komplett entkoppelt und können parallel ausgeführt werden. Ohne diesen Trick würden alle "data"- und "freq"-Instanzen in den Sequenzfunktionen sich intern denselben Datenspeicher teilen und die FiltHP()-Aufrufe müssten praktisch nacheinander ausgeführt werden. Zu beachten sind nur Parameter, die direkt an potentiell lange rechnende Funktionen übergeben werden und deswegen länger gesperrt werden müssen. Die Variable "chunksize" im Beispiel (lokaler Name "count") wird nur sehr kurz beim Lesen verriegelt, so dass der "+0"-Trick hier nicht notwendig ist. Siehe auch vorstehende Erläuterung unter "Verriegelung von Variablenzugriffen".

Bei einer Gruppe mit vielen Kanälen soll die FFT eines jeden Kanals berechnet werden.

```
GrFFTs = FFT(GrInput, 0, 0)
```

Diese Berechnung soll durch Parallelisierung der Ausführung beschleunigt werden:

```
count = GrChanNum?(GrInput)
chunksize = floor(count/4)
BEGIN_PARALLEL
   GrFFT1 = !MyFFT(GrPart(GrInput, 1,              chunksize))
   GrFFT2 = !MyFFT(GrPart(GrInput, 1+ chunksize,   chunksize))
   GrFFT3 = !MyFFT(GrPart(GrInput, 1+ chunksize*2, chunksize))
   GrFFT4 = !MyFFT(GrPart(GrInput, 1+ chunksize*3, count-3*chunksize))
END_PARALLEL
GrFFTs = GrConcat(GrFFT1, GrFFT2, GrFFT3, GrFFT4)
```

Die Sequenzfunktion !MyFFT ist dabei wie folgt definiert:

```
; Deklaration:
; !MyFFT(Par [Datentyp: Normal]) => Result [Komplex]
Result = FFT(Par, 0, 0)
```

Siehe auch:

END_PARALLEL

Unterstützt ab:

Version 2022, Editionen: Professional, Enterprise, Runtime

BINARY

Einstellen des BINÄR-Formats zum anschließenden Laden von Dateien mit dem Befehl LOAD
Alternativer Name: **BINÄR**

Deklaration:

```
BINARY EwFormat EwBytes EwVorspann EwZwischenraum EwAnzahl EwBits EwVorzeichen EwMinimum EwMaximum Zeichenfolge
```

Parameter:

EwFormat	Format
	1 : Reelle Gleitkomma-Zahlen im Intel-Format
	2 : Ganze Zahlen im Intel-Format
	3 : Reelle Gleitkomma-Zahlen im Motorola-Format
	4 : Ganze Zahlen im Motorola-Format
EwBytes	Die Anzahl der Bytes einer Zahl
	1 : 1 Byte (nur ganzzahlig)
	2 : 2 Byte (nur ganzzahlig)
	4 : 4 Bytes (ganzzahlig oder reell)
	6 : 6 Bytes (nur reell)
	8 : 8 Bytes (nur reell)
EwVorspann	Gibt die Länge des zu überlesenden Vorspanns in Bytes an.
EwZwischenraum	Gibt an, wie viele Bytes zwischen den einzelnen zu lesenden Zahlen zu überlesen sind.
EwAnzahl	Gibt die Anzahl der einzulesenden Zahlenwerte an. Eine Anzahl von 0 deutet an, dass alle Werte bis zum Ende der Datei zu lesen sind.
EwBits	Nur für Typ= 2 oder 4 (ganzzahlig): Die Zahl der gültigen Bits in einer ganzen Zahl. Wird zusammen mit Minimum und Maximum zur Skalierung der ganzen Zahlen benutzt.
EwVorzeichen	Nur für Typ= 2 oder 4 (ganzzahlig): Gibt an, ob die ganzen Zahlen mit Vorzeichen (Zweierkomplementdarstellung) oder ohne zu interpretieren sind. 0 (kein Vorzeichen) oder 1 (mit Vorzeichen).
EwMinimum	Nur für Typ= 2 oder 4 (ganzzahlig): Realer Wert der kleinsten darstellbaren Zahl im angegebenen Format, ist selbst eine reelle Zahl, wird auch als Untergrenze des Messbereichs verstanden.
EwMaximum	Nur für Typ= 2 oder 4 (ganzzahlig): Realer Wert der größten darstellbaren Zahl im angegebenen Format, ist selbst eine reelle Zahl, wird auch als Obergrenze des Messbereichs verstanden.
Zeichenfolge	Eine Zeichenfolge, nach der in der Datei gesucht wird, bevor mit dem Lesen des Vorspanns und der Zahlenwerte begonnen wird. Die Zeichenfolge darf keine Leerzeichen enthalten. Wird keine Zeichenfolge als Parameter übergeben, wird gleich am Anfang der Datei mit dem Lesen des Vorspanns begonnen.

Beschreibung:

Das Dateiformat für den Befehl LOAD wird auf ein Binär-Format gestellt.

Werden dem Befehl BINARY keine Parameter hinzugefügt, wird das aktuell eingestellte Binär-Format gewählt.

Wenn Parameter angegeben werden, wird ein neues Binär-Format komplett definiert. Die anzugebenden Parameter hängen ab vom Typ des Binärformates. Wenn Parameter angegeben werden, müssen alle Parameter angegeben werden.

Der Befehl BINARY stellt eine Automatisierung des Dialogfeldes 'Optionen'/ 'Datei laden'/ 'Binär' dar. Dieses Dialogfeld erreichen Sie beispielsweise, wenn Sie im 'Datei laden'-Dialog bei eingestelltem Binär-Format die Schaltfläche 'Optionen' betätigen. Eine genauere Erläuterung und weitere komplette Beispiele finden Sie im Benutzerhandbuch, Kapitel 'Dateiverwaltung'.

- Jeder Parameter, mit Ausnahme der Zeichenfolge, darf ein fester Zahlenwert oder eine Variable vom Typ Einzelwert (EW) sein. Die Benutzung von Variablen als Parameter ermöglicht das Lesen von variablen Dateiformaten.
- Eine Alternative zum BINÄR-Import mit der Befehlskombination ASCII/BINARY (besonders für komplexere Formate) ist die Verwendung des imc-Datei-Assistenten zur Erstellung eines Importfilters und die Anwendung des Filters mit dem Befehl FASLOAD oder der Funktion FileOpenFAS().
- Multithreading: Der Befehl wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
BINARY
LOAD DATA.BIN
```

Das zuletzt eingestellte Binärformat wird wieder eingestellt, nachdem zwischendurch einige Dateien z.B. im imc/FAMOS-Format geladen wurden. Danach wird die Datei DATA.BIN geladen.

```
BINARY 1 4 20 8 100 Data:
LOAD DATA.BIN
```

Das Format zum Lesen folgender binärer Datei wird festgelegt: Die Datei enthält an ihrem Anfang Informationen über ein Experiment, die eigentlichen binären Daten werden durch die Zeichenfolge "Data:" eingeleitet. Es werden danach reelle Zahlen in 4-Byte-Darstellung gelesen. Die ersten 5 Zahlen (= 20 Bytes) werden nicht gelesen. Danach wird auch nur jede dritte Zahl gelesen (d. h.: 2 * 4 = 8 Bytes Zwischenraum zwischen den Zahlen selbst). Es werden 100 Zahlen eingelesen. Der in FAMOS erzeugte Datensatz hat damit die Länge 100.

```
Mini = -10.0
Maxi = 10.0
Bits = 12
BINARY 2 2 0 0 0 Bits 0 Mini Maxi
LOAD DATA.BIN
```

Es sollen aus einer Datei alle Zahlen gelesen werden. In der Datei ist neben den Zahlen kein Vorspann vorhanden. Die Zahlen liegen als 12-Bit-Werte in 2-Byte-Darstellung vor. Sie stammen ohne Vorverarbeitung direkt aus einem AD-Wandler, haben also kein Vorzeichen. Der Messbereich des Wandlers beträgt -10V ... 10V.

Siehe auch:

LOAD, ASCII, FileOpenASCII, FileOpenFAS

BitAnd

Bitweise "UND"-Verknüpfung

Deklaration:

```
BitAnd ( Daten, Maske, EwDatenformat ) -> Ergebnis
```

Parameter:

Daten	Erster zu verknüpfender Wert bzw. Datensatz. Erlaubte Datentypen: [ND],[XY].
Maske	Zweiter zu verknüpfender Wert bzw. Datensatz (Maske) [ND].
EwDatenformat	Anzunehmendes (ganzzahliges) Datenformat. Erlaubte Werte sind:
	64 : 64 Bit ohne Vorzeichen
	32 : 32 Bit ohne Vorzeichen
	16 : 16 Bit ohne Vorzeichen
	8 : 8 Bit ohne Vorzeichen
	1 : 1 Bit (digital)
	-8 : 8 Bit mit Vorzeichen
	-16 : 16 Bit mit Vorzeichen
	-32 : 32 Bit mit Vorzeichen
	-64 : 64 Bit mit Vorzeichen
Ergebnis	Ergebnis der bitweisen Verknüpfung.

Beschreibung:

Bitweise 'UND'-Verknüpfung zweier Zahlen. Beide Operanden werden in das angegebene ganzzahlige Datenformat konvertiert und eine logische 'UND'-Verknüpfung auf jedem Paar korrespondierender Bits durchgeführt. Das Ergebnisbit ist 1, falls beide Bits 1 sind, ansonsten 0.

Der zweite Parameter muss entweder ein einzelner Wert oder ein Datensatz mit exakt identischer Struktur (Länge, Segmentierung, Events) wie der erste Parameter sein, in diesem Fall erfolgt eine punktweise Verknüpfung der korrespondierenden Werte.

Negative ganze Zahlen werden als Zweierkomplement abgebildet.

Bei Konvertierung in den Datentyp 'Digital' werden alle Werte, die ungleich 0 sind, als 1 betrachtet. Das Ergebnis ist identisch zum logischen AND-Operator.

Das Ergebnis hat das im Parameter [EwDatenformat] angegebene Format.

Beispiele:

In einem 1-Byte-Datensatz (Wertebereich 0..255) werden die oberen 4 Bits auf 0 gesetzt.

```
Erg = BitAnd(Data, 0x0F, 8)
```

In einem 4 Byte breiten Datensatz wird der Wert des 2. Bytes ermittelt.

```
Byte2 = BitShift(BitAnd(Data, 0x00FF0000, 32), -16, 32)
```

Es werden alle Punkte ermittelt, an denen in 2 Datensätzen das 7. Bit gleichzeitig 1 ist.

```
Result = BitAnd(BitGet(Port1, 7), BitGet(Port2, 7), 1)
; ist kompatibel zu:
; Result = BitGet(Port1, 7) AND BitGet(Port2, 7)
```

Siehe auch:

BitOr, BitNot, BitShift, BitGet, BitSet

BitGet

Abfragen eines Bits

Deklaration:

```
BitGet ( Daten, EwBitNummer ) -> NullOderEins
```

Parameter:

Daten	Abzufragender Einzelwert bzw. Datensatz. Erlaubte Datentypen: [ND],[XY].
EwBitNummer	Nummer des zu setzenden Bits. Das rechteste Bit (LSB) hat die Nummer 1. Erlaubter Bereich: 1<= [EwBitNummer] <= 64.
NullOderEins	Ergebnis der Abfrage.

Beschreibung:

Abfragen eines Bits im Bitmuster einer ganzen Zahl. Die Werte des übergebenen Datensatzes werden in eine ganze Zahl konvertiert und der Inhalt (0/1) des entsprechenden Bits zurückgegeben.

Negative ganze Zahlen werden als Zweierkomplement abgebildet.

Das Ergebnis hat das Datenformat 'Digital'.

Beispiele:

Es werden alle Punkte ermittelt, an denen in 2 Datensätzen das 7. Bit gleichzeitig 1 ist.

```
Result = BitAnd(BitGet(Port1, 7), BitGet(Port2, 7), 1)
; ist kompatibel zu:
; Result = BitGet(Port1, 7) AND BitGet(Port2, 7)
```

In einem 1-Byte-Datensatz wird das 2. Bit auf den Wert des 1. Bits gesetzt.

```
Bit1 = BitGet(Data, 1, 8)
Result = BitSet(Data, Bit1, 2)
```

Siehe auch:

BitAnd, BitOr, BitNot, BitShift, BitSet

BitNot

Bitweise Invertierung

Deklaration:

```
BitNot ( Parameter, EwDatenformat ) -> Ergebnis
```

Parameter:

Parameter	Zu invertierender Einzelwert bzw. Datensatz. Erlaubte Datentypen: [ND],[XY].
EwDatenformat	Anzunehmendes (ganzzahliges) Datenformat. Erlaubte Werte sind:
	64 : 64 Bit ohne Vorzeichen
	32 : 32 Bit ohne Vorzeichen
	16 : 16 Bit ohne Vorzeichen
	8 : 8 Bit ohne Vorzeichen
	1 : 1 Bit (digital)
	-8 : 8 Bit mit Vorzeichen
	-16 : 16 Bit mit Vorzeichen
	-32 : 32 Bit mit Vorzeichen
	-64 : 64 Bit mit Vorzeichen
Ergebnis	Ergebnis der bitweisen Verknüpfung.

Beschreibung:

Bitweise Invertierung des Wertes des Parameters. Jeder Wert wird in das angegebene ganzzahlige Datenformat konvertiert und eine logische Invertierung für jedes Bit durchgeführt. Das Ergebnisbit ist 0, falls das Eingangsbit 1 ist, ansonsten 1.

Negative ganze Zahlen werden als Zweierkomplement abgebildet.

Bei Konvertierung in den Datentyp 'Digital' werden alle Werte, die ungleich 0 sind, als 1 betrachtet. Das Ergebnis ist identisch zum logischen NOT-Operator.

Das Ergebnis hat das im Parameter [EwDatenformat] angegebene Format.

Beispiele:

Alle Bits eines 2 Byte breiten Datensatzes werden invertiert.

```
Result = BitNot(Port, 16)
```

Invertierung eines digitalen Datensatzes.

```
Result = BitNot(DigChannel1, 1)
; ist kompatibel zu:
; Result = Not(DigChannel)
```

Realisierung einer bitweisen XOR-Verknüpfung

```
Result = BitOr(BitAnd(x, BitNot(y, 32), 32), BitAnd(BitNot(x, 32), y, 32), 32)
```

Siehe auch:

BitAnd, BitOr, BitShift, BitGet, BitSet

BitOr

Bitweise "ODER"-Verknüpfung

Deklaration:

```
BitOr ( Daten, Maske, EwDatenformat ) -> Ergebnis
```

Parameter:

Daten	Erster zu verknüpfender Wert bzw. Datensatz. Erlaubte Datentypen: [ND],[XY].
Maske	Zweiter zu verknüpfender Wert bzw. Datensatz (Maske) [ND].
EwDatenformat	Anzunehmendes (ganzzahliges) Datenformat. Erlaubte Werte sind:
	64 : 64 Bit ohne Vorzeichen
	32 : 32 Bit ohne Vorzeichen
	16 : 16 Bit ohne Vorzeichen
	8 : 8 Bit ohne Vorzeichen
	1 : 1 Bit (digital)
	-8 : 8 Bit mit Vorzeichen
	-16 : 16 Bit mit Vorzeichen
	-32 : 32 Bit mit Vorzeichen
	-64 : 64 Bit mit Vorzeichen
Ergebnis	Ergebnis der bitweisen Verknüpfung.

Beschreibung:

Bitweise 'ODER'-Verknüpfung zweier Zahlen. Beide Operanden werden in das angegebene ganzzahlige Datenformat konvertiert und eine logische 'ODER'-Verknüpfung auf jedem Paar korrespondierender Bits durchgeführt. Das Ergebnisbit ist 0, falls beide Bits 0 sind, ansonsten 1.

Der zweite Parameter muss entweder ein einzelner Wert oder ein Datensatz mit exakt identischer Struktur (Länge, Segmentierung, Events) wie der erste Parameter sein, in diesem Fall erfolgt eine punktweise Verknüpfung der korrespondierenden Werte.

Negative ganze Zahlen werden als Zweierkomplement abgebildet.

Bei Konvertierung in den Datentyp 'Digital' werden alle Werte, die ungleich 0 sind, als 1 betrachtet. Das Ergebnis ist identisch zum logischen OR-Operator.

Das Ergebnis hat das im Parameter [EwDatenformat] angegebene Format.

Beispiele:

In einem 1-Byte-Datensatz werden unteren 4 Bits auf 1 gesetzt.

```
Erg = BitOr(Data, 0x0F, 8)
```

Es werden alle Punkte ermittelt, an denen in einem Datensatz das 2. und/oder 7. Bit gesetzt ist.

```
Result = BitOr(BitGet(Port1, 2), BitGet(Port1, 7), 1)
; ist kompatibel zu:
; Result = BitGet(Port1, 2) OR BitGet(Port1, 7)
```

Realisierung einer bitweisen XOR-Verknüpfung

```
Result = BitOr(BitAnd(x, BitNot(y, 32), 32), BitAnd(BitNot(x, 32), y, 32), 32)
```

Siehe auch:

BitAnd, BitNot, BitShift, BitGet, BitSet

BitSet

Setzen eines Bits

Deklaration:

```
BitSet ( Daten, NullOderEins, EwBitNummer ) -> Ergebnis
```

Parameter:

Daten	Zu bearbeitender Einzelwert bzw. Datensatz. Erlaubte Datentypen: [ND],[XY].
NullOderEins	Zu setzender Wert bzw. Werte. [ND].
EwBitNummer	Nummer des zu setzenden Bits. Das rechteste Bit (LSB) hat die Nummer 1. Erlaubter Bereich: 1<= [EwBitNummer] <= 64.
Ergebnis	Ergebnis der Operation.

Beschreibung:

Setzen eines Bits im Bitmuster einer ganzen Zahl. Die Werte des Eingangs-Datensatzes werden ggf. in ganze Zahlen gewandelt und der Inhalt des angegebenen Bits auf 0/1 gesetzt. Anschließend wird das resultierende Bitmuster wieder in das Datenformat des Eingangsdatensatzes zurückkonvertiert. Werte mit Betrag > 10^14 können nicht verlustfrei konvertiert werden, in diesem Falle wird eine Warnung erzeugt.

Der zweite Parameter muss entweder ein einzelner Wert oder ein Datensatz mit exakt identischer Struktur (Länge, Segmentierung, Events) wie der erste Parameter sein, in diesem Fall erfolgt eine punktweise Verknüpfung der korrespondierenden Werte. Hat der 2. Parameter einen Wert von 0, so wird das entsprechende Bit auf 0 gesetzt, sonst auf 1.

Negative ganze Zahlen werden als Zweierkomplement abgebildet.

Das Ergebnis hat das selbe Datenformat wie der Eingangsparameter.

Beispiele:

In einem 1-Byte-Datensatz wird das 7. Bit auf 1 gesetzt.

```
Result = BitSet(Data, 1, 7)
; kompatibel zu:
; Result = BitOr(Data, 0x40, 8)
```

In einem 1-Byte-Datensatz wird das 2. Bit auf den Wert des 1. Bits gesetzt.

```
Bit1 = BitGet(Data, 1, 8)
Result = BitSet(Data, Bit1, 2)
```

Siehe auch:

BitAnd, BitOr, BitNot, BitShift, BitGet

BitShift

Verschiebung des Bitmusters einer ganzen Zahl um eine angegebene Stellenanzahl.

Deklaration:

```
BitShift ( Parameter, EwStellen, EwDatenformat ) -> Ergebnis
```

Parameter:

Parameter	Zu bearbeitender Einzelwert bzw. Datensatz. Erlaubte Datentypen: [ND],[XY].
EwStellen	Anzahl der Stellen, um die verschoben werden soll. Eine Zahl größer 0 bedeutet eine Linksverschiebung, eine Zahl kleiner 0 eine Rechtsverschiebung.
EwDatenformat	Anzunehmendes (ganzzahliges) Datenformat. Erlaubte Werte sind:
	64 : 64 Bit ohne Vorzeichen
	32 : 32 Bit ohne Vorzeichen
	16 : 16 Bit ohne Vorzeichen
	8 : 8 Bit ohne Vorzeichen
	-8 : 8 Bit mit Vorzeichen
	-16 : 16 Bit mit Vorzeichen
	-32 : 32 Bit mit Vorzeichen
	-64 : 64 Bit mit Vorzeichen
Ergebnis	Ergebnis der Bitverschiebung.

Beschreibung:

Der Parameter wird in das angegebene ganzzahlige Datenformat konvertiert und eine Bitmuster-Verschiebung um die angegebene Stellenzahl ausgeführt.

Eine positive Angabe für die Stellenzahl bedeutet eine Linksverschiebung, von rechts werden Nullen nachgeschoben. Links heraus geschobene Bits verschwinden. Eine Linksverschiebung um 1 Stelle entspricht einer Multiplikation mit 2.

Eine negative Angabe für die Stellenzahl bedeutet eine Rechtsverschiebung, von links wird mit 0 oder 1 abhängig von Zahlenformat und Vorzeichen nachgeschoben. Rechts heraus geschobene Bits verschwinden. Für Zahlenformate ohne Vorzeichen werden die durch den Verschiebevorgang frei gewordenen Bitpositionen mit Nullen angefüllt. Für Zahlenformate mit Vorzeichen wird das Vorzeichenbit verwendet, um die frei gewordenen Bitpositionen zu füllen. Wenn die Zahl positiv ist, wird also 0 verwendet, und wenn die Zahl negativ ist, wird 1 verwendet. Eine Rechtsverschiebung um 1 Stelle entspricht einer Division durch 2 (ggf. mit Rundung auf die nächstkleinere Zahl).

Der Betrag der Stellenzahl muss kleiner sein als die durch den dritten Parameter definierte Bitbreite der Zahl, beispielweise muss für [EwDatenformat] = 16 der Wert für [EwStellen] im Bereich von -15..15 liegen.

Negative ganze Zahlen werden als Zweierkomplement abgebildet.

Das Ergebnis hat das im Parameter [EwDatenformat] angegebene Format.

Beispiele:

2 Datensätze, die jeweils 1 Byte breit sind, werden zu einem 2 Byte breiten Datensatz kombiniert, wobei der 2. Datensatz in die oberen 8 Bit verschoben wird.

```
WordData = BitOr(BitShift(ByteData2, 8, 16), ByteData1, 16)
```

In einem 4 Byte breiten Datensatz wird der Wert des 2. Bytes ermittelt.

```
Byte2 = BitShift(BitAnd(Data, 0x00FF0000, 32), -16, 32)
```

Siehe auch:

BitAnd, BitNot, BitOr, BitGet, BitSet

BoxMessage

Text-Ausgabefenster mit abfragbaren Schaltflächen
Alternativer Name: **BoxNachricht**

Deklaration:

```
BoxMessage ( TxTitel, TxText, TxOption ) -> EwRückgabe
```

Parameter:

TxTitel	Titel des Fensters
TxText	Text im Fenster
TxOption	Bestimmt Typ der Knöpfe und Art des Symbols
	"!1" : Ausrufezeichen, OK
	"!2" : Ausrufezeichen, OK/Abbrechen
	"!3" : Ausrufezeichen, Wiederholen/Abbrechen
	"!4" : Ausrufezeichen, Ja/Nein
	"?1" : Fragezeichen, OK
	"?2" : Fragezeichen, OK/Abbrechen
	"?3" : Fragezeichen, Wiederholen/Abbrechen
	"?4" : Fragezeichen, Ja/Nein
	"S1" : Stoppzeichen, OK
	"S2" : Stoppzeichen, OK/Abbrechen
	"S3" : Stoppzeichen, Wiederholen/Abbrechen
	"S4" : Stoppzeichen, Ja/Nein
	"I1" : Info(i)-Symbol, OK
	"I2" : Info(i)-Symbol, OK/Abbrechen
	"I3" : Info(i)-Symbol, Wiederholen/Abbrechen
	"I4" : Info(i)-Symbol, Ja/Nein
EwRückgabe	Welche Schaltfläche wurde betätigt?
	1 : Die Schaltfläche "Ja", "OK" bzw. "Wiederholen" wurde betätigt.
	0 : Die Schaltfläche "Nein" bzw. "Abbrechen" wurde betätigt.

Beschreibung:

Erzeugt ein Fenster mit Überschrift, Text, Symbol und Schaltflächen, welches durch Betätigung einer Schaltfläche quittiert werden muss. Welche Schaltfläche gedrückt wurde, wird in [EwRückgabe] geliefert.

- Die Abfrage des Rückgabewertes kann in imc FAMOS weggelassen werden (sinnvoll dann, wenn nur eine Schaltfläche vorhanden ist).
- Die Position, an der die Box erscheint, kann mit der Funktion SetBoxPos() vorgegeben werden.
- Sie können die Position der Ausgabebox auch manuell ändern (z.B. durch Verschieben mit der Maus). Die Position wird für die Dauer der aktuellen Sitzung gemerkt.
- Multithreading: Die Funktion darf nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

Erzeugt ein Fenster mit einem 'Fragezeichen'-Symbol und 'Ja'/'Nein' Schaltflächen:

```
TxQuestion = "Do you really want to stop the evaluation?"
Stop = BoxMessage("Evaluation", TxQuestion, "?4")
```

Siehe auch:

BoxOutput, BoxValue?, BoxText?, SetBoxPos, DlgFileName, Dialog

BoxOutput

Ausgabe von Text und/oder einer Zahl in einem Ausgabefenster.
Alternativer Name: **BoxAusgabe**

Deklaration:

```
BoxOutput ( TxAusgabe, EwWert, TxFormat, EwOption )
```

Parameter:

TxAusgabe	Auszugebender Text
EwWert	Auszugebende Zahl
TxFormat	Formatierung für den Wert
EwOption	Wie soll die Ausgabe erfolgen?
	0 : Ausgabe in einem separaten Ausgabefenster. Die Meldung muss quittiert werden.
	1 : Die Anzeige erfolgt im Ausgabefeld im imc FAMOS-Hauptfenster.

Beschreibung:

Wenn [EwWert] eine einzelne Zahl darstellt, wird diese in das durch [TxFormat] angegebene Format konvertiert und an den Ausgabetext angehängt. Ansonsten ist die Pseudokonstante 'EMPTY' anzugeben, [TxFormat] wird dann ignoriert.

Ansonsten können für die Formatangabe [TxFormat] die folgenden Optionen gewählt werden. Der erste Buchstabe ist die obligatorische Formatkennung, danach ist eine bzw. zwei Zahlen entsprechend der gewünschten Formatierung anzugeben.

TxFormat	Beschreibung	Beispiel
"e.N"	Fließkomma-Format N: Anzahl der Nachkommastellen.	3.456 mit "e.2" -> "3.46E+00"
"FV.N"	Festkomma-Format. V: Minimale Anzahl der Vorkommastellen. N: Anzahl der Nachkommastellen. Überzählige Vorkomma-Stellen werden mit Leerzeichen aufgefüllt.	3.456 mit "f2.2" -> " 3.46"
"gV.N"	Festkomma-Format. V: Anzahl der Vorkommastellen. N: Anzahl der Nachkommastellen. Überzählige Vorkomma-Stellen werden mit Nullen aufgefüllt.	3.456 mit "g2.2" -> "03.46"
"a.N"	Automatisch. N: Maximal ausgegebene Stellen. Je nach Zahlenwert wird die kürzere Darstellung gewählt. Abschließende Nullen nach dem Dezimalpunkt werden weggelassen, ggf. auch der Dezimalpunkt selbst. Somit ideal für ganze Zahlen.	3.40056 mit "a.2" -> "3.4"
""	Entspricht "a.6"	
"xG"	Hexadezimal-Format. G: Anzahl der Stellen insgesamt	340056 mit "x8" -> "340056"
"bG"	Binär-Format. G: Anzahl der Stellen insgesamt	34.56 mit "b4" -> "100011"

Komma oder Punkt

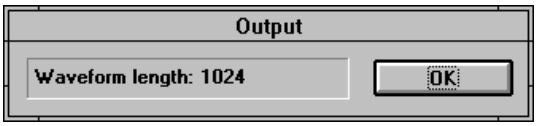
Wenn in diesen Formatstrings der Punkt durch ein Komma ersetzt wird, erfolgt die Ausgabe mit einem Dezimalkomma statt einem Dezimalpunkt. Ansonsten kann der Punkt auch weggelassen werden (Beispiel: "f23" ist gleichwertig zu "f2.3", "f2,3" dagegen erzwingt ein Dezimalkomma.

Wenn in diesen Formatstrings der Punkt durch ein Semikolon ersetzt wird, erfolgt die Ausgabe entsprechend der globalen Voreinstellung für das bevorzugte Dezimaltrennzeichen für reelle Zahlen ('Extra'/'Optionen'/'Anzeige' bzw. SetOption("Display.DecimalSeparator",...).

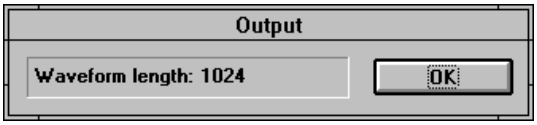
- Das Ausgabefeld im Hauptfenster wird geleert, bevor Sie eine einzelne Formel ausführen. Wenn Sie eine Sequenz im Einzelschritt-Modus abarbeiten oder eine einzelne Formel ausführen, ist also immer nur die Ausgabe des letzten BoxOutput()-Befehls sichtbar.
- Die Position, an der die Dialogbox erscheint, kann mit der Funktion SetBoxPos() vorgegeben werden.
- Sie können die Ausgabebox auch manuell in Position und Größe verändern. Die Position und Größe wird für die Dauer der aktuellen Sitzung gemerkt.
- Falls versehentlich eine Endlosschleife programmiert worden ist, kann mit der Tastenkombination "STRG"+"UNTBR" die Sequenz abgebrochen werden.
- Multithreading: Die Funktion darf mit [EwOption] = 0 (also separate Ausgabebox) nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

```
leng = Leng?(data)
BoxOutput("Length of data: ", leng, "", 0)
```




```
VMean = Mean(data)
BoxOutput("Calculated Mean : ", VMean, "f23", 0)
DELETE data
BoxOutput("Calculation Completed!", EMPTY, "", 1)
```



Siehe auch:

BoxMessage, BoxValue?, BoxText?, SetBoxPos, DlgFileName, Dialog

BoxText?

Aufruf einer Dialogbox zur Eingabe eines Textes.

Deklaration:

```
BoxText? ( TxTitel, TxInit, EwOption ) -> TxEingabe
```

Parameter:

TxTitel	Titel der Box
TxInit	Vorschlag im Eingabefeld beim Aufruf
EwOption	Legt die Größe des Fensters fest.
	0 : Position und Größe der Box wie zuletzt oder wie mit SetBoxPos() festgelegt.
	1 : Die Größe der Box wird so berechnet, dass [TxTitel] komplett angezeigt wird. Position der Box wie zuletzt oder wie mit SetBoxPos() festgelegt. Explizite Zeilenumbrüche in [TxTitel] (mit CarriageReturn/LineFeed-Kombination "~013~010") werden berücksichtigt.
TxEingabe	Eingegebener Text

Beschreibung:

Eine kleine Dialogbox zur Eingabe eines Textes wird aufgerufen. Nach erfolgter Eingabe und Bestätigung mit [OK] wird der eingegebene Wert zurückgegeben.

- Die Position, an der die Dialogbox erscheint, kann mit der Funktion SetBoxPos() vorgegeben werden.
- Sie können die Eingabebox auch manuell in Position und Größe verändern. Die Position und Größe wird für die Dauer der aktuellen Sitzung gemerkt und gilt auch für die Funktion BoxValue?().
- Die maximale Länge des zurückgegebenen Textes ist 255 Zeichen.
- Falls versehentlich eine Endlosschleife programmiert worden ist, kann mit der Tastenkombination 'STRG'+'UNTBR' die Sequenz abgebrochen werden.
- Multithreading: Die Funktion darf nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

```
TxName = BoxText?("Please enter the file name:", "default.dat", 0)
```

Mit automatischer Größenanpassung (2-zeilig):

```
TxName = BoxText?("Please enter the file name:~013~010(EXCEL format required)", "default.xls", 1)
```

Siehe auch:

BoxValue?, BoxOutput, BoxMessage, SetBoxPos, DlgFileName, Dialog

BoxValue?

Aufruf einer Dialogbox zur Eingabe einer Zahl.
Alternativer Name: **BoxWert?**

Deklaration:

```
BoxValue? ( TxTitel, EwInit, EwOption ) -> EwEingabe
```

Parameter:

TxTitel	Titel der Box
EwInit	Eintrag im Eingabefeld beim Aufruf
EwOption	Legt die Größe des Fensters fest.
	0 : Position und Größe der Box wie zuletzt oder wie mit SetBoxPos() festgelegt.
	1 : Die Größe der Box wird so berechnet, dass [TxTitel] komplett angezeigt wird. Position der Box wie zuletzt oder wie mit SetBoxPos() festgelegt. Explizite Zeilenumbrüche in [TxTitel] (mit CarriageReturn/LineFeed-Kombination "~013~010") werden berücksichtigt.
EwEingabe	Eingegebener Wert

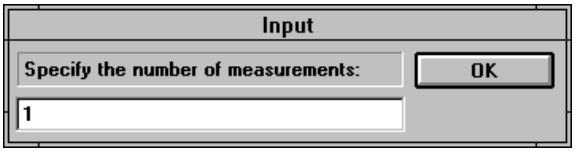
Beschreibung:

Eine kleine Dialogbox zur Eingabe einer Zahl wird aufgerufen. Nach erfolgter Eingabe und Bestätigung mit [OK] wird der eingegebene Wert zurückgegeben. Falls die Eingabe nicht in eine Zahl konvertiert werden kann, wird der Dialog nicht beendet.

- Die Position, an der die Dialogbox erscheint, kann mit der Funktion SetBoxPos() vorgegeben werden.
- Sie können die Eingabebox auch manuell in Position und Größe verändern. Die Position und Größe wird für die Dauer der aktuellen Sitzung gemerkt und gilt auch für die Funktion BoxText?().
- Falls versehentlich eine Endlosschleife programmiert worden ist, kann mit der Tastenkombination 'STRG'+'UNTBR' die Sequenz abgebrochen werden.
  Multithreading: Die Funktion darf nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

```
Count = BoxValue?("Please specify the number of measurements:", 1, 0)
```



Mit automatischer Größenanpassung (2-zeilig):

```
Count = BoxValue?("Please specify the number of measurements.~013~010Must be smaller than 10.", 1, 1)
```

Siehe auch:

BoxText?, BoxOutput, BoxMessage, SetBoxPos, Dialog

BoxVarSelector

Ein Dialogfenster mit einer Liste der zum Aufrufzeitpunkt in FAMOS vorhandenen Variablen wird angezeigt. Die Funktion liefert die Namen der vom Anwender ausgewählten Variablen zurück.

Deklaration:

```
BoxVarSelector ( SelektionsTyp [, TxNamensFilter] [, TxTypFilter] [, TxTitel] [, TxBeschreibung] ) -> TxOderTxaVariablenAuswahl
```

Parameter:

SelektionsTyp	Gibt an, ob der Anwender genau 1 Variable (Einzelselektion) oder beliebig viele Variablen (Mehrfachselektion) auswählen kann. Bestimmt auch den Datentyp des Ergebnisses (Text oder Textfeld).
	"single" : Einzelselektion
	"multi" : Mehrfachselektion
TxNamensFilter	Es werden nur solche Variablen angezeigt, deren Namen dem hier definierten Namens-Muster entsprechen. Die Jokerzeichen "*" (beliebig viele beliebige Zeichen) und "?" (genau 1 beliebiges Zeichen) können in ihrer üblichen Interpretation verwendet werden. Ein vorangestelltes "!" negiert die Bedingung, es werden also alle Variablen angezeigt, die nicht dem Muster entsprechen. Ein leerer Text bedeutet, dass keine Filterung nach Namen durchgeführt wird. Das hier eingestellte Filter kann vom Anwender im Dialogfenster verändert werden. (optional , Standardwert: "")
TxTypFilter	Es werden nur solche Variablen angezeigt, deren Datentyp der hier definierten Bedingung entspricht. Ein leerer Text bedeutet, dass keine Filterung nach Typ durchgeführt wird. (optional , Standardwert: "")
	"" : Keine Typ-Filterung
	"SV" : Einzelwert (Normaler Datensatz der Länge 1)
	"ND" : Normaler Datensatz
	"ND(..)" : Normaler Datensatz, Länge > 1
	"ND(->)" : Normaler Datensatz, keine Events, keine Segmente
	"ND(..,->)" : Normaler Datensatz, keine Events, keine Segmente, Länge > 1
	"CX" : Komplexer Datensatz
	"!CX" : Datensatz, nicht komplex
	"CX(->)" : Komplexer Datensatz, keine Events, keine Segmente
	"RI" : Komplexer Datensatz in Real-/Imaginärteil-Darstellung
	"MP" : Komplexer Datensatz in Betrag/Phase-Darstellung
	"DP" : Komplexer Datensatz in Dezibel/Phase-Darstellung
	"XY" : XY-Datensatz
	"!XY" : Datensatz, kein XY
	"XY(->)" : XY-Datensatz, keine Events, keine Segmente
	"XY(/)" : XY-Datensatz mit monoton wachsender x-Spur.
	"XY(->,/)" : XY-Datensatz mit monoton wachsender x-Spur. Keine Events, keine Segmente.
	"TSA" : Datensatz, TimeStamp-ASCII
	"!TSA" : Datensatz, kein TimeStamp-ASCII
	"TSA(->)" : TimeStamp-ASCII, keine Events, keine Segmente
	"DS" : Datensatz (Typ beliebig), also kein Text, Textfeld oder Datengruppe
	"Evn" : Datensatz (Typ beliebig) mit Events
	"Seg" : Datensatz (Typ beliebig) mit Segmenten
	"SegEvn" : Datensatz (Typ beliebig) mit Segmenten und Events
	"!Evn" : Datensatz (Typ beliebig) ohne Events
	"!Seg" : Datensatz (Typ beliebig) ohne Segmente
	"!SegEvn" : Datensatz (Typ beliebig), keine Events, keine Segmente
	"->" : Abkürzung für "!SegEvn", Datensatz (Typ beliebig), keine Events, keine Segmente
	"Monotone" : Einzelwert, Normaler Datensatz oder XY-Datensatz mit monoton wachsender x-Spur.
	"/" : Abkürzung für "Monotone". Einzelwert, Normaler Datensatz oder XY-Datensatz mit monoton wachsender x-Spur.
	"Digital" : Digitaler Datensatz
	"!Digital" : Datensatz, nicht digital.
	"TXT" : Text
	"!TXT" : Kein Text, also Datensatz, Textfeld oder Datengruppe
	"TXA" : Textfeld
	"!TXA" : Kein Textfeld, also Datensatz, Text oder Datengruppe
	"TX*" : Text oder Textfeld
	"!TX*" : Kein Text oder Textfeld, also Datensatz oder Datengruppe
	"Group" : Datengruppe
	"!Group" : Keine Datengruppe
	"Empty" : Variable ist leer. Länge 0 bei Datensätzen, Textlänge 0 bei Texten, Elementanzahl 0 bei Textfeldern und TSA.
	"!Empty" : Variable ist nicht leer.
	"@M" : Die Variable ist einer Messung zugeordnet.
	"!@M" : Die Variable ist keiner Messung zugeordnet.
TxTitel	Der in der Titelzeile des Dialogfensters angezeigte Text. Ein leerer Text bedeutet, dass ein Standard-Titel ("Variablen auswählen") verwendet wird. (optional , Standardwert: "")

TxBeschreibung	Zusätzlicher Text, der gegebenenfalls oberhalb der Auswahlliste angezeigt wird. (optional , Standardwert: "")
TxOderTxaVariablenAuswahl	Bei [SelektionsTyp]="multi" ein Textfeld mit den ausgewählten Variablennamen. Wenn das Dialogfenster abgebrochen wurde oder keine Variable ausgewählt wurde, ist die Dimension des Textfeldes 0. Bei [SelektionsTyp]="single" ein Text mit dem ausgewählten Variablennamen. Wenn das Dialogfenster abgebrochen wurde, wird eine leere Textvariable zurückgegeben.

Beschreibung:

Die Funktion zeigt ein Dialogfenster, in dem der Anwender in einer Liste unter den zum Aufrufzeitpunkt vorhandenen Variablen auswählen kann. Zurückgeliefert werden die Namen der gewählten Variablen.

- Lokale Variablen werden nicht angezeigt.
- Damit ein Variablenname in Formeln angegeben werden kann, muss er bestimmten Regeln gehorchen (z.B. keine Leerzeichen, erstes Zeichen keine Ziffer etc.). Falls dies nicht der Fall ist, muss der Name zusätzlich in geschweifte Klammern {...} eingeschlossen werden. Die durch diese Funktion gelieferten Namen werden ggf. automatisch um die geschweiften Klammern ergänzt.
- Die Position, an der die Box erscheint, kann mit der Funktion SetBoxPos() vorgegeben werden. Sie können die Position der Box auch manuell ändern (z.B. durch Verschieben mit der Maus). Die Position wird für die Dauer der aktuellen Sitzung gemerkt.
- Multithreading: Die Funktion darf nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

```
; Einzelauswahl aus allen Variablen
txName = BoxVarSelector("single")

; Einzelauswahl aus allen Variablen, deren Name mit "channel" beginnt
txName = BoxVarSelector("single", "channel*")

; Mehrfachauswahl aus allen Variablen, deren Name nicht mit "_" beginnt
txaNames = BoxVarSelector("multi", "!_*")

; Mehrfachauswahl aus allen Variablen, die vom Datentyp "XY" sind,
; eine monoton wachsende x-Spur aufweisen und weder Events noch Segmente besitzen
txaNames = BoxVarSelector("multi", "", "XY(->,/)")

; Mehrfachauswahl aus allen Variablen, die zur Messung "0001" gehören, äquidistant abgetastet sind
; und weder Events noch Segmente besitzen. Fenstertitel und Beschreibung wird vorgegeben.
txaNames = BoxVarSelector("multi", "*@0001", "ND(->)", "Spectral Analysis", "Please select the data for which you want to perform a spectral analysis.")
```

Der Anwender wählt eine Variable aus, von der anschließend das Spektrum berechnet wird. Es werden nur Variablen angezeigt, deren Name mit "chan" beginnt und deren Typ als Parameter für die die nachfolgende Berechnung zulässig ist.

```
txaName = BoxVarSelector("single", "chan*", "ND(->)")
IF txName <> ""
spectrum = Spec(<txName>)
   SHOW <txName>
   SHOW spectrum
END
```

Der Anwender trifft eine Auswahl unter den Variablen mit Messungszugehörigkeit, deren Name den Teiltext "engine" enthält. Die ausgewählten Variablen werden geglättet.

```
txaNames = BoxVarSelector("multi", "*engine*", "@M", "Smooth data")
FOREACH ELEMENT txName IN txaNames
   <txName> = Smo5(<txName>)
END
```

Der Anwender wählt Variablen aus, die anschließend zusammen in einer Datei gespeichert werden sollen. Texte und Textfelder werden nicht zur Auswahl angeboten.

```
txaNames = BoxVarSelector("multi", "", "!TX*", "", "Please select the data to be saved.")
c = TxArrayGetSize(txaNames)
IF c > 0
   fh = FileOpenDSF("result.dat", 1)
   IF fh <>> 0
      FOR i = 1 TO c
         txaName = txaNames[i]
         FileObjWrite(fh, <txaName>)
      END
      FileClose(fh)
   END
END
```

Siehe auch:

VarGetInit, VarGetInit2, VarGetName?, VarExist?, BoxText?, SetBoxPos, Dialog

Unterstützt ab:

Version 2024

BREAK

Der Befehl bricht die Abarbeitung der übergeordneten Schleife ab. Die Sequenzausführung wird nach dem Ende des Schleifenkörpers fortgesetzt.

Deklaration:

```
BREAK
```

Beschreibung:

Der Befehl kann innerhalb einer WHILE-, FOR- oder FOREACH-Schleife verwendet werden.

Beispiele:

Aus einer Datei mit mehreren Datenobjekten wird das erste gefundene Textobjekt ausgelesen.

```
fh = FileOpenDSF("test.dat", 0)
IF fh > 0
   n = FileObjNum?(fh)
   FOR i = 1 to n
       IF FileObjType?(fh, i) = 2
         text = FileObjRead(fh, i)
         BREAK ; Schleife abbrechen
      END
   END
   FileClose(fh)
END
```

Siehe auch:

WHILE, FOR, FOREACH, CONTINUE

BSave

Speichern von Datensätzen, Einzelwerten oder Text als Binärdatei.
Alternativer Name: **BSichern**

Deklaration:

```
BSave ( EwOption, Daten, TxDatei ) -> EwFehler
```

Parameter:

EwOption	Konfiguration des Formats der Ausgabedatei. Siehe Beschreibung.
Daten	Zu speichernde Variable. Datensatz, Einzelwert oder Text.
TxDatei	Name der Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet. Wenn keine Dateierweiterung angegeben wurde, wird ".BIN" verwendet.
EwFehler	Fehlercode
	0 : Die Datei wurde erfolgreich gespeichert.
	1 : Nicht genügend Arbeitsspeicher vorhanden.
	2 : Fehler beim Anlegen der Datei. Bitte prüfen Sie den angegebenen Dateinamen.
	3 : Die Datei konnte nicht geschrieben werden. Möglicherweise reicht der Platz auf dem Datenträger nicht aus.

Beschreibung:

Bedeutung des Parameters [EwOption]:

0	Neue Datei erzeugen
1	An bestehende Datei anhängen
addieren Sie:	
0	Intel-Format (LSB zuerst)
10	Motorola Format (MSB zuerst)
addieren Sie:	
0	als Bytes speichern (8 Bit, Bereich 0..255)
100	als Worte speichern (16 Bit, Bereich 0..65535)
200	als Longs speichern (32 Bit, Bereich 0..4.2E9)
300	als 4-Byte Fliesskomma-Zahl nach IEEE
400	als 8-Byte Fliesskomma-Zahl nach IEEE
addieren Sie:	
1000	Die Werte werden im Zweierkomplement gespeichert. Bei Bytes, Worten und Longs verändern sich dabei die Bereiche wie folgt: Bytes: -128..127, Worte: -32768..32767, Long: -2.1E9..2.1E9

Falls kein voller Dateiname angegeben ist, wird das aktuelle Standardverzeichnis verwendet. Das aktuelle Standardverzeichnis wird beim Starten von FAMOS auf die Voreinstellung (Dialog 'Optionen'/ 'Verzeichnisse') gesetzt. Es kann mit der Funktion SetOption() umgesetzt werden.

Beispiele:

Es soll eine Datei "Curve.bin" erzeugt werden, die am Anfang "Length:xxxxxxxx" enthält. xxxxxxxx ist die Zahl der folgenden Werte als 4Byte-Integer-Zahl, die als 2Byte-Integer-Zahlen gespeichert werden.

```
BSave(0, "Length:", "Curve")
Len = Leng?(Data)
BSave(201, Len, "Curve")
BSave(101, Data, "Curve")
```

Siehe auch:

FileSave

CASE

Kennzeichnet eine Alternative in einer durch den Befehl SWITCH eingeleiteten Fallunterscheidung (mehrfache Verzweigung). Der hier angebene Ausdruck wird mit dem beim SWITCH-Block angegebenen Wert verglichen. Bei Gleichheit werden die diesem CASE-Block zugehörigen Anweisungen ausgeführt.

Deklaration:

```
CASE Testausdruck
```

Parameter:

Testausdruck	Wert bzw. Werteliste, die mit dem SWITCH - Vergleichswert verglichen wird.

Beschreibung:

Das Ende der zu diesem CASE gehörigen Anweisungen wird durch ein weiteres CASE, einen DEFAULT-Befehl oder END-Befehl bestimmt.

Nachdem ein CASE-Block durchlaufen wurde, wird die Sequenz-Ausführung beim korrespondierenden END fortgesetzt. Nachfolgende CASE-Anweisungen werden also nicht mehr durchlaufen, auch wenn sie die Eintrittsbedingung ebenfalls erfüllen würden.

Die Auflösung des Testausdrucks muss entweder eine Zahl oder einen Text ergeben und im Typ zum beim SWITCH angegebenen Vergleichswert passen.

Sie können hier z.B. eine Konstante, eine Variable oder einen arithmetischen Ausdruck angeben:

```
CASE 0
CASE "Monday"
CASE var
CASE var+1
```

Ebenfalls erlaubt ist eine Komma-getrennte Liste von Ausdrücken. Der Vergleich ist erfüllt, wenn wenigstens einer der Werte identisch zum Vergleichswert ist.

```
CASE 0, 2, 5
CASE "Monday", "Tuesday", "Friday"
CASE var1, var2
CASE var1+1, var+2
```

Handelt es sich um einen numerischen Vergleichswert, kann auch ein Bereich unter Verwendung des Schlüsselwortes TO ('UnteresLimit TO OberesLimit') angegeben werden. Der Vergleich ist erfüllt, wenn der Vergleichswert im angegebenen Bereich (inklusive Ränder) liegt.

```
CASE 0 TO 10
CASE var1 TO var1+10
```

Beim Textvergleich wird bezüglich Groß-/Kleinschreibung nicht unterschieden.

Beispiele:

Zu einem Wert, der normalerweise im Bereich von 0 bis 100 liegt, wird ein beschreibender Text gebildet.

```
SWITCH Round(value, 1)
CASE 0
   Tx = "Lower limit"
CASE 1 TO 48
    Tx = "Lower half"
CASE 49,50,51
    Tx = "Center"
CASE 52 To 99
   Tx = "Upper half"
CASE 100
   Tx = "Upper limit"
DEFAULT
   Tx = "Invalid Value"
END
```

Der Anwender wird aufgefordert, einen Datei zum Laden auszuwählen. Anhand der Dateierweiterung wird das Dateiformat erkannt und die entsprechende Funktion zum Laden aufgerufen.

```
TxFileName = DlgFileName("", "", "",0)
TxFileExt = FsSplitPath(TxFileName, 3)
SWITCH TxFileExt
CASE ".dat", ".raw"
   ; Load imc data file
   fh = FileOpenDSF(TxFileName, 0)
   ;...
CASE ".xls"
   ; Load EXCEL file
```

```
   fh = FileOpenXLS(TxFileName, 0)
   ;...
DEFAULT
   PAUSE ==> Invalid file format
END
```

Siehe auch:

SWITCH, DEFAULT, IF

CCF

Kreuzkorrelation eines Datensatzes mit einem Referenzdatensatz
Alternativer Name: **KKF**

Deklaration:

```
CCF ( ReferenzDaten, TestDaten ) -> Ergebnis
```

Parameter:

ReferenzDaten	Referenz-Datensatz
TestDaten	Test-Datensatz
Ergebnis	Ergebnis der Kreuzkorrelation

Beschreibung:

Die Kreuzkorrelationsfunktion (KKF) gibt an, wie ähnlich zwei Datensätze einander bei verschiedenen Verschiebungen in x-Richtung sind. Die Kreuzkorrelationsfunktion nimmt Werte zwischen -1 und +1 an.

Dabei bedeutet ein Wert von 1, dass bei der entsprechenden Verschiebung maximale Korrelation vorliegt. Beide Signale haben dann denselben Verlauf. Ein Wert von -1 bedeutet, dass beide Signale entgegengesetzt gleich sind. Wenn ein Signal positiv ist, ist das andere gleich groß, aber negativ. Ein Wert von 0 bedeutet, dass die beiden Signale bei dieser Verschiebung nichts miteinander zu tun haben (nicht korreliert sind). Es können alle Werte zwischen -1 und +1 auftreten.

Mit der Kreuzkorrelationsfunktion lässt sich feststellen, ob ein Signal in einem anderen (versteckt) vorhanden ist und außerdem, wie stark verzögert ein Signal im anderen Datensatz auftritt. Dazu ist das Maximum der Kreuzkorrelationsfunktion besonders interessant. Seine Lage gibt an, wie stark verzögert das Signal im anderen Datensatz auftritt. Seine Höhe gibt an, wie ähnlich das verzögerte Signal dem anderen ist.

Um die Verzögerung der Signale zueinander richtig interpretieren zu können, ist die Reihenfolge der Parameter der Funktion CCF() wichtig. Der erste Parameter ist der Referenzdatensatz, der das nicht verzögerte Originalsignal enthält. Der zweite Parameter enthält den Testdatensatz, der ein verzögertes (oft auch gestörtes) Signal enthält. Die x-Koordinate des Maximums der Kreuzkorrelationsfunktion gibt dann direkt die Verschiebung des Testdatensatzes gegenüber dem Referenzdatensatz an. In der vorliegenden Implementierung wird die Kreuzkorrelationsfunktion mit Hilfe der Funktionen FFT() und iFFT() berechnet, um eine akzeptable Rechenzeit zu erzielen. Das hat neben der Rechenzeitersparnis einige wichtige Konsequenzen:

Zunächst einmal wird intern der kürzere von beiden Datensätzen mit Nullen verlängert, so dass beide gleich lang sind. Dann werden beide in ihrer Länge auf die nächstkleinere Zweierpotenz verkürzt, wenn ihre Länge keine Zweierpotenz ist. Benutzen Sie die Funktion Red2(), um ein Verkürzen zu vermeiden. Weiterhin hat eine Benutzung der FFT zur Folge, dass die nun vorliegenden Datensätze als periodisch angesehen werden, d. h. die Datensätze werden in beide Richtungen (nach links und rechts) periodisch fortgesetzt gedacht. Stellt ein Datensatz einen Impuls dar, so wird das Signal interpretiert, als läge eine Kette von vielen Impulsen vor.

Die Kreuzkorrelationsfunktion selbst wird über die volle Länge der verkürzten, als periodisch angesehenen Datensätze berechnet. Es ist nicht sinnvoll, die Kreuzkorrelationsfunktion über einen anderen Bereich zu berechnen, da sie periodisch ist und dieselbe Periode aufweist. Wenn Sie in der Kreuzkorrelationsfunktion eine große Verschiebung von 0.9 Perioden ablesen, so hat das dieselbe Bedeutung wie eine kleine Verschiebung in negativer Richtung, nämlich -0.1 Perioden.

Bei der Berechnung der Kreuzkorrelationsfunktion wird wie bei der FFT eine Fensterfunktion berücksichtigt. Beide zu korrelierenden Datensätze werden nach einer eventuellen Kürzung mit der für die FFT eingestellten Fensterfunktion gewichtet. Sie können damit Randeffekte und Sprungstellen unterdrücken, die durch die periodische Fortsetzung der Signale entstehen. Siehe dazu die Funktion FFT(). Möchten Sie keine Fensterung, wählen Sie die Rechteck-Fensterfunktion.

- Die x-Skalierung der KKF ist die der übergebenen Datensätze. Diese beiden sollten sinnvollerweise die gleiche x-Skalierung aufweisen. Ansonsten wird eine Warnung erzeugt und die Skalierung des Referenzdatensatzes übernommen.
- Die KKF hat keine y-Einheit, da sie normiert ist.
- Wenn die Länge eines Parameters 2^27 überschreitet, wird eine Fehlermeldung ausgegeben. Es ist dann keine Berechnung möglich. Verkürzen Sie dann den entsprechenden Datensatz mit den Funktionen Leng() oder Red2(). Die maximal bearbeitbare Länge der Datensätze ist 134.217.728.
- Wenn die Länge eines Datensatzes keine Zweierpotenz ist, wird meistens eine entsprechende Warnung erzeugt. Der Datensatz wird dann gegebenenfalls automatisch verkürzt (abgeschnitten).
- Die Funktion CCF() benutzt intern die Funktion FFT. Diese benötigt temporären Speicher (Arbeitsspeicher). Ist nicht ausreichend Speicherplatz im Arbeitsspeicher vorhanden, wird die Berechnung abgebrochen.
- Die Funktion CCF() ist so normiert, dass sie von den Amplituden der übergebenen Datensätze unabhängig ist. Einer der beiden Datensätze darf vorher mit einem festen Faktor multipliziert werden, ohne dass sich das Korrelationsergebnis verändert. Sind die zu vergleichenden Datensätze mit einem hohen Mittelwert (y-Offset) versehen, lohnt es sich, die Datensätze vorerst mittelwertfrei zu machen. Ansonsten beeinflusst der Mittelwert das Korrelationsergebnis wesentlich stärker als das eigentliche Signal. In der Tat wird damit nicht mehr die Kreuzkorrelationsfunktion sondern die Kreuzkovarianzfunktion berechnet.
- Die hier implementierte KKF ist auf das Produkt der Effektivwerte der beiden übergebenen Datensätze normiert. Damit hat der erzeugte Datensatz keine y-Einheit. Der Effektivwert ist die Wurzel aus der Autokorrelationsfunktion an der Stelle Null.

Beispiele:

```
NDccf = CCF(NDref, NDtest)
```

Einfachste Anwendung der KKF, Datensätze werden ggf. auf eine Zweierpotenz von Punkten verkürzt.

```
NDccf= CCF(NDref-Mean(NDref),NDtest-Mean(NDtest))
```

Einfache Anwendung der KKF, Datensätze werden ggf. auf eine Zweierpotenz von Punkten verkürzt. Die Datensätze werden jedoch zuvor mittelwertfrei gemacht. Sind die Signale mit einem hohen y-Offset (Mittelwert) behaftet, ist so ein besserer Vergleich möglich.

```
NDtest1 = Red2(NDtest)
NDref1 = RSamp(NDref, NDtest1)
NDccf = CCF(NDref1, NDtest1)
maxi = Max(NDccf)
```

Es ist zu prüfen, ob ein kurzes Referenzmuster (NDref) in einem längeren, leicht verrauschten Datensatz NDtest wieder zu finden ist. Der längere Datensatz wird mit der Funktion Red2 auf eine geeignete Länge gebracht, ohne ihn abzuschneiden. Der kürzere Datensatz wird mit der Funktion RSamp so abgetastet, dass er in seiner Abtastzeit genau passend ist. Die beiden erzeugten Datensätze werden korreliert. Wenn das Maximum der KKF größer als 0.8 ist, kann angenommen werden, dass das Testsignal sehr viel mit dem Referenzsignal zu tun hat.

Siehe auch:

ACF, CorrCoeff, FFT, Red2

CFCFilter

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Berechnung eines CFC-Filters nach SAE J211/1.

Deklaration:

```
CFCFilter ( Signal, CFC_Wert, Einschwingen ) -> Ergebnis
```

Parameter:

Signal	Zu filternde Zeitdaten
CFC_Wert	CFC-Wert, z.B. 60, 180, 600, 1000.
Einschwingen	Behandlung des Einschwingens, (filter start-up):
	0 : Punktsymmetrisch um 1. Wert: Die nach SAE vorgeschlagene Methode der punktsymmetrischen (spiegelverkehrten) Erweiterung des Signals in beide Richtungen. Symmetrie-Zentrum ist dabei jeweils der Endwert. (Data transposed about end points)
	1 : Punktsymmetrisch um 0.0: Die nach SAE vorgeschlagene Methode der punktsymmetrischen (spiegelverkehrten) Erweiterung des Signals in beide Richtungen. Symmetrie-Zentrum ist dabei der Punkt 0.0.(Data transposed about zero magnitude)
	2 : Konstant 0.0: Das Signal wird mit Nullen konstant außerhalb seines Randes fortgesetzt.
	3 : Achsensymmetrisch um Rand: Um den Rand wird das Signal gespiegelt fortgesetzt. Dabei erfolgt eine Spiegelung an der senkrechten Achse x = Randwert.
Ergebnis	Gefilterter Datensatz

Beschreibung:

Das CFC-Filter ist ein digitales, nicht kausales Filter. Dieses Filter wird auf das Zeitsignal angewendet. Das Filter wirkt glättend, versucht aber dabei, Kanten zeitlich nicht zu verschieben.

Algorithmus nach Norm SAE J211/1, März 1995.

Aus dem CFC-Wert wird ein Tiefpassfilter mit folgender Grenzfrequenz entworfen:

- fg = 2.0775 * CFC_Wert

Die Grenzfrequenz des Filters muss deutlich unter der halben Abtastfrequenz liegen, damit das Filter überhaupt gerechnet werden kann.

Das Filter beginnt allerdings erst sinnvoll zu werden (wirklich glättende Wirkung zu haben), wenn der CFC-Wert deutlich unterhalb 1/8 der Abtastfrequenz liegt, möglichst aber noch wesentlich tiefer.

Beispiele:

```
Glatter = CFCFilter ( Signal, 600, 0 )
```

Ein Signal wird mit einem CFC-Wert von 600 geglättet. Das Einschwingverhalten nach SAE wird benutzt.

Siehe auch:

FilterAnalog, FiltTP, dFilt, Glatt

CharToSv

Der ASCII-Code des ersten Zeichens eines Textes wird bestimmt.
Alternativer Name: **ZeichenT**

Deklaration:

```
CharToSv ( TxText ) -> EwCode
```

Parameter:

TxText	Der Text, von dem der ASCII-Code des ersten Zeichens zu bestimmen ist.
EwCode	ASCII-Code des Zeichens

Beschreibung:

Ein Text wird als Parameter angegeben. Der ASCII-Code des ersten Zeichens des Textes wird ermittelt und zurückgegeben.

- Zulässige ASCII-Codes sind 1..255.
- Für einen leeren Text wird eine 0 zurückgegeben.
- Ein Tabulator-Zeichen hat beispielsweise den Code 9, ein Carriage Return 13, ein Line Feed 10.
- Nicht alle ASCII-Codes können unter Windows angezeigt werden. Beachten Sie vor allem, dass der ANSI-Zeichensatz benutzt wird. ASCII-Tabellen, die z.B. für die DOS-Umgebung gemacht sind, stellen die Zeichen im OEM-Zeichensatz dar.

Beispiele:

Die ASCII-Codes von einem kleinen und einem großen Buchstaben werden ermittelt:

```
aLower = CharToSv("a")
aUpper = CharToSv("A")
```

Siehe auch:

SvToChar, TtoSv, TReplace, TConv

Chrct

Korrektur über Kennlinie
Alternativer Name: **Kenn**

Deklaration:

```
Chrct ( Daten, Kennliniendaten ) -> Korrigiert
```

Parameter:

Daten	Datensatz, der über eine Kennline korrigiert werden soll. Erlaubte Datentypen: [ND],[XY]
Kennliniendaten	Kennlinen-Datensatz. Erlaubte Datentypen: [ND],[XY]
Korrigiert	Korrigierter Datensatz

Beschreibung:

Ein Datensatz wird mit einer Kennlinie korrigiert. Dabei wird für jeden Originalwert aus der Kennlinie ein Ergebniswert berechnet. Liegt ein Originalwert zwischen zwei Kennlinienwerten, wird dazwischen linear interpoliert.

- Der Kennlinien-Datensatz kann maximal 8388608 Werte enthalten.
- Wenn der Kennlininen-Datensatz vom Typ XY ist, muss die X-Spur monoton wachsend sein.
- Der zu korrigierende Datensatz darf strukturiert sein (Events/ Segmente).
- Wegen der linearen Interpolation zwischen den Kennlinienwerten sollte bei stark nichtlinearen Kennlinien die Anzahl der Datenpunkte für die Kennlinie ausreichend groß gewählt werden, um Fehler zu vermeiden.

Beispiele:

Ein Datensatz [sensor_data] wird mit einem Kennlinien-Datensatz korrigiert.

```
sensor_corr = Chrct(sensor_data, characteristic)
```

Siehe auch:

LFit, eFit, MatrixGet

Clip

Begrenzt den Y-Wertebereich auf ein vorgegebenes Band.
Alternativer Name: **Band**

Deklaration:

```
Clip ( Daten, EwOben, EwUnten ) -> Ergebnis
```

Parameter:

Daten	Zu begrenzender Datensatz. Erlaubte Typen: [ND],[XY].
EwOben	Obere Grenze
EwUnten	Untere Grenze
Ergebnis	Begrenzter Datensatz. Y-Wertebereich liegt zwischen [EwUnten] und [EwOben]

Beschreibung:

Alle y-Werte (Amplitude) des Eingangs-Datensatzes werden auf den Wertebereich eingeschränkt, der durch den zweiten und dritten Parameter gebildet wird. Alle Werte, die größer sind als dieser Bereich, werden auf die Obergrenze des Bereichs gesetzt, alle Werte, die kleiner sind, auf die Untergrenze. Als Obergrenze wird stets der größere Wert des zweiten und dritten Parameters benutzt. Die Reihenfolge der letzten beiden Parameter ist also beim Aufruf der Funktion beliebig.

- Der erste Parameter darf strukturiert sein (Events/ Segmente).
- Die Einheiten der Einzelwerte sollten mit denen des Datensatzes übereinstimmen, werden aber großzügigerweise nicht überprüft, was besonders beim Aufruf mit Zahlenwerten anstatt Variablen bequem ist.
- Es wird keine Bandbegrenzung im Sinne von Frequenzen durchgeführt.

Beispiele:

Daten werden auf einen Bereich zwischen positiver und negativer Versorgungsspannung begrenzt, um einen Sättigungseffekt zu simulieren:

```
NDsaturated = Clip(NDdata, Uplus, Uminus)
```

Siehe auch:

Cut, STri, Scale

CLIPBOARD

Kurvenbild in Windows-Zwischenablage kopieren
Alternativer Name: **ABLAGE**

Der Befehl ist veraltet, statt dessen sollten die leistungsfähigere Funktion CwAction("Clipboard.Copy") verwendet werden.

Deklaration:

```
CLIPBOARD Variablenname
```

Parameter:

Variablenname	Variable, deren Kurvenbild in die Windows-Zwischenablage kopiert werden soll.

Beschreibung:

Mit diesem Befehl wird ein Kurvenbild in die Zwischenablage kopiert. Von dort kann es in andere Windows-Applikationen (DTP, Textverarbeitung, Zeichenprogramm) eingefügt werden

Beispiele:

```
CLIPBOARD slope
```

Das Kurvenbild der Variable "slope" wird in die Zwischenablage kopiert. Wurde der Datensatz vorher nicht angezeigt, so wird er entsprechend den Voreinstellungen angezeigt, kopiert und das Kurvenfenster anschließend wieder gelöscht

Siehe auch:

CwAction

ClsOff2ChannelHistogram

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Berechnung der Verbunddichte, 2dimensionales Histogramm.

Deklaration:

```
ClsOff2ChannelHistogram ( Kanal1, Kanal2, Min1, Max1, Klassen1, Min2, Max2, Klassen2, Optionen ) -> Ergebnis
```

Parameter:

Kanal1	Der 1. Eingangskanal. Dessen Klassen tauchen in der Matrix in x-Richtung, also entlang einer Spalte auf.
Kanal2	Der 2. Eingangskanal. Dessen Klassen tauchen in der Matrix in z-Richtung, also entlang einer Zeile auf.
Min1	Minimum des Bereichs für den 1. Kanal
Max1	Maximum des Bereichs für den 1. Kanal
Klassen1	Anzahl Klassen für den 1. Kanal
Min2	Minimum des Bereichs für den 2. Kanal
Max2	Maximum des Bereichs für den 2. Kanal
Klassen2	Anzahl Klassen für den 2. Kanal
Optionen	
	0 : Randklassen geschlossen, Achsen in Klassen beschriften
	1 : Randklassen offen, Achsen in Klassen beschriften
	2 : Randklassen geschlossen, physikal. Einheit nutzen
	3 : Randklassen offen, physikal. Einheit nutzen
Ergebnis	

Beschreibung:

Die Werte von Kanal1 und Kanal2 werden in Klassen entsprechend den Grenzen und den Klassenanzahlen eingeteilt.

In der Matrix wird gezählt, wie oft jede Kombination von Klassen aus Kanal1 und Kanal2 auftritt.

Für jedes Paar von Messwerten aus Kanal1 und Kanal2 wird in der Matrix der Zähler um 1 erhöht.

Die Randklassen können offen oder geschlossen sein.

Bei offenen Randklassen werden Werte außerhalb des Bereich von Min..Max in die äußerste Klasse (Randklasse) einsortiert.

Bei geschlossenen Randklassen werden Werte, die außerhalb des Bereichs Min..Max liegen, ignoriert, also nicht gezählt.

Die Beschriftung der Achsen (x-Achse und z-Achse) kann in Klassen erfolgen (Klasse 0, 1, 2, ...) oder in den physikalischen Einheiten der Eingangskanäle (z.B. -0.1 Nm .. +0.1 Nm).

Min1, Max1, Klassen1: Für den ersten Kanal der Bereich. Klassierung von Min1 (<Max1) .. Max1 mit einer Klassenanzahl Klassen1 (>=4).

Min2, Max2, Klassen2: Für den ersten Kanal der Bereich. Klassierung von Min2 (<Max2) .. Max2 mit einer Klassenanzahl Klassen2 (>=4).

Beispiele:

```
Min1 = -0.1
Max1 = 0.1
Klassen1 = 30
Min2 = -10.0
Max2 = 10.0
Klassen2 = 40
Optionen = 3
Verbunddichte = ClsOff2ChannelHistogram ( a1, a2, Min1, Max1, Klassen1, Min2, Max2, Klassen2, Optionen)
```

Verbunddichteklassierung über 2 Kanäle a1 und a2.

Der erste Kanal wird in 30 Klassen von -0.1 .. 0.1 unterteilt, der zweite Kanal in 40 Klassen von -10.0 .. +10.0

Die Randklassen sind offen. Die Beschriftung der Matrix erfolgt in physikalischen Einheiten.

Siehe auch:

ClsTimeAtLevel, ClsOffRevolutionsHistogram, ClsOffRevolutionsMatrix

ClsOffFromRainflowGetLevelCrossing

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassendurchgangsverfahren (level crossing counting) aus Rainflow

Deklaration:

```
ClsOffFromRainflowGetLevelCrossing ( ClsHandle, Bezug ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Bezug	die Amplitude des Signals, die als Bezug (Null-Linie) gelten soll
Ergebnis	Verteilung

Beschreibung:

Aus der aktuell gehaltenen Rainflow-Matrix und dem aktuellen Residuum wird die Verteilung Klassendurchgang (level crossing counting) ermittelt.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix)
ClsOffRainflowFeedResidue ( ClsHandle, Residue, 1.0)
_Bezug = 0.0 ; Bezugspegel für einige DIN-Verfahren
DurchgangOff = ClsOffFromRainflowGetLevelCrossing( ClsHandle, _Bezug)
```

Zuerst wird die Rainflow-Analyse initialisiert.

Eine bereits bestehende Matrix wird samt Residuum eingespeist.

Aus diesen Information wird das DIN-Verfahren abgeleitet und die gewünschte Verteilung bestimmt.

Siehe auch:

ClsOffRainflowInit1, KlsDurch, ClsOffFromRainflowGetRangePair

ClsOffFromRainflowGetMinMaxPeak

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Spitzenwert III aus Rainflow

Deklaration:

```
ClsOffFromRainflowGetMinMaxPeak ( ClsHandle, Min_Oder_Max ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Min_Oder_Max	Was soll klassiert werden?
	0 : Alle Minima werden klassiert
	1 : Alle Maxima werden klassiert
Ergebnis	Verteilung

Beschreibung:

Aus der aktuell gehaltenen Rainflow-Matrix und dem aktuellen Residuum wird die Verteilung Spitzenwert III ermittelt.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix)
ClsOffRainflowFeedResidue ( ClsHandle, Residue, 1.0)
Spitzen3PosOff = ClsOffFromRainflowGetMinMaxPeak( ClsHandle, 1)
Spitzen3NegOff = ClsOffFromRainflowGetMinMaxPeak( ClsHandle, 0)
```

Zuerst wird die Rainflow-Analyse initialisiert.

Eine bereits bestehende Matrix wird samt Residuum eingespeist.

Aus diesen Information wird das DIN-Verfahren abgeleitet und die gewünschte Verteilung bestimmt.

Die Resultate werden für die Minima und die Maxima getrennt bestimmt.

Siehe auch:

ClsOffRainflowInit1, KlsSpit3, ClsOffFromRainflowGetZeroCrossingPeak, ClsOffFromRainflowGetLevelCrossing

ClsOffFromRainflowGetPeak

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Spitzenwert II (peak counting) aus Rainflow

Deklaration:

```
ClsOffFromRainflowGetPeak ( ClsHandle, Bezug ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Bezug	der Wert des Signals, der als Bezug (Null-Linie) gelten soll.
Ergebnis	Verteilung

Beschreibung:

Aus der aktuell gehaltenen Rainflow-Matrix und dem aktuellen Residuum wird die Verteilung Spitzenwert II (peak counting) ermittelt.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix)
ClsOffRainflowFeedResidue ( ClsHandle, Residue, 1.0)
_Bezug = 0.0 ; Bezugspegel für einige DIN-Verfahren
Spitzen2Off = ClsOffFromRainflowGetPeak( ClsHandle, _Bezug)
```

Zuerst wird die Rainflow-Analyse initialisiert.

Eine bereits bestehende Matrix wird samt Residuum eingespeist.

Aus diesen Information wird das DIN-Verfahren abgeleitet und die gewünschte Verteilung bestimmt.

Siehe auch:

ClsOffRainflowInit1, KlsSpit2, ClsOffFromRainflowGetZeroCrossingPeak, ClsOffFromRainflowGetLevelCrossing

ClsOffFromRainflowGetRangePair

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Spannenpaar (range-pair counting) aus Rainflow

Deklaration:

```
ClsOffFromRainflowGetRangePair ( ClsHandle ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Ergebnis	Verteilung

Beschreibung:

Aus der aktuell gehaltenen Rainflow-Matrix und dem aktuellen Residuum wird die Verteilung Spannenpaar (range-pair counting) ermittelt.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix)
ClsOffRainflowFeedResidue ( ClsHandle, Residue, 1.0)
SpannenpaarOff = ClsOffFromRainflowGetRangePair( ClsHandle )
```

Zuerst wird die Rainflow-Analyse initialisiert.

Eine bereits bestehende Matrix wird samt Residuum eingespeist.

Aus diesen Information wird das DIN-Verfahren abgeleitet und die gewünschte Verteilung bestimmt.

Siehe auch:

ClsOffRainflowInit1, KlsSPaar, ClsOffFromRainflowGetLevelCrossing

ClsOffFromRainflowGetReconstruction

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Rekonstruktion von Zeitdaten aus Rainflow

Deklaration:

```
ClsOffFromRainflowGetReconstruction ( ClsHandle, RandWert, LageMin, LageMax, ExVerlängerung, EW_1, EW_2, EW_3 ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
RandWert	Diesen Wert soll das Signal am Anfang und am Ende haben. Wenn >= 1e30 angegeben wird, dann wird kein Wert künstlich ergänzt. Achtung: Beim Ergänzen von Werten kann sich das Klassierergebnis durchaus merklich ändern, da ggf. neue Extremwerte eingefügt werden.
LageMin	Relative Lage eines relativen Minimums zwischen den Klassengrenzen. Wert zwischen 0.0 und 1.0. Insbesondere folgende Werte: 0.0 Das Minimum wird immer am unteren Klassenrand erzeugt; 1.0 Genau am oberen Klassenrand; 0.5 Genau in der Mitte. Bei Zweifel immer 0.5 angeben. Werte nahe 0.0 und 1.0 sind riskant, weil bei einer nachfolgenden Klassierung die Werte schon in die benachbarten Klassen gezählt werden können. Der Wert kann bei einem Minimum durchaus auf 0.1 gesetzt werden. Das vergrößert dann die Spannen etwas. Wenn es wegen zu geringer Hysterese vorkommen kann, dass Start- und Ziel-Klasse einer Schwingung gleich sind, dann sollte LageMax > LageMin sein.
LageMax	Wie LageMin, aber für ein relatives Maximum. Bei Zweifel auf 0.5. Kann auch auf 0.9 gesetzt werden, zur Unterscheidung von einem Minimum in derselben Klasse.
ExVerlängerung	Um so viele Messwerte wird ein Extremwert im Ergebnis verlängert. Bei 0 keine Verlängerung. Dann ist ein Extremwert 1 Abtastschritt breit. Bei Zweifel auf 1 setzen. Wenn ein Extremwert nur ganz kurz erreicht wird, dann erreicht ihn bei solch kurzer Anregung der Prozess evtl. noch nicht gleich. Dann ist die Verlängerung sinnvoll. 0 <= ExVerlängerung <= 100
EW_1	Reserviert. = 0
EW_2	Reserviert. = 0
EW_3	Reserviert. = 0
Ergebnis	Rekonstruktion

Beschreibung:

Aus der aktuellen Rainflow-Matrix und dem aktuellen Residuum wird ein Zeitsignal rekonstruiert, das etwa dieselbe Belastung verursacht wie die ursprünglichen Daten.

Eine Klassierung des rekonstruierten Signals mit Hysterese null führt auf etwa dieselbe Matrix und dasselbe Residuum.

Der Zeitverlauf kann aber beliebig abweichen.

Die Funktion liefert nur korrekte Ergebnisse, wenn die Matrix und das Residuum zueinander passen, wenn also beide von derselben Klassierung herrühren. Wurde eins von beiden unstimmig verändert, ist das Ergebnis undefiniert.

Genau die Anzahl von Schwingungen, die in der Matrix stehen, wird rekonstruiert. Das können so viele Schwingungen sein, dass kein so langer Ergebnisdatensatz erzeugt werden kann.

Dann sind vorher die Werte der Matrix entsprechend zu verkleinern, z.B. Division durch einen festen Faktor > 1.

Das Residuum sollte noch vorhanden sein. Es sollte noch nicht in die Matrix hineingezählt worden sein.

Ggf. gibt es Abweichungen beim Residuum des erzeugten Datensatz gegenüber dem Original.

Der Randwert verursacht ggf. zusätzliche Spannen, die gezählt werden.

Denn die Verbindung zum Randwert muss ggf. noch einen zusätzlichen Extremwert enthalten, damit die benachbarten Extremwerte als solche auch gelten.

Es wird versucht, die Orientierung der Spannen zu erhalten. D.h. eine Spanne von Klasse 5 nach Klasse 7 wird unterschieden von einer von 7 nach 5. Allerdings kann das nicht immer erreicht werden, vor allem, wenn beide Richtungen gleichzeitig auftreten.

Rückgabe ist Reconstruction, der rekonstruierte Zeitdatensatz. Abtastzeit und Einheiten müssen nachträglich noch gesetzt werden.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix)
ClsOffRainflowFeedResidue ( ClsHandle, Residue, 1.0)
_LageMin = 0.2 ; 0 .. 1, standard 0.5
_LageMax = 0.8 ; 0 .. 1, standard 0.5
_Verlänger = 0
_Randwert = 1e30 ; 1e30 falls nicht gewünscht. Sonst der Zahlenwert für Randwert
Reconstruction = ClsOffFromRainflowGetReconstruction ( ClsHandle, _Randwert, _LageMin, _LageMax, _Verlänger, 0, 0, 0)
yeinheit Reconstruction Nm
xdelta Reconstruction 0.001
```

Eine bereits früher durchgeführte Rainflow-Analyse liegt als Rainflow-Matrix mit Residuum vor.

Beides wird eingespeist in die initialisierte Rainflow-Analyse.

Danach wird ein kompakter Zeitdatensatz rekonstruiert

Siehe auch:

ClsOffRainflowInit1

ClsOffFromRainflowGetSpans

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Häufigkeit der Spannen aus der Rainflow-Matrix. Für jede Klasse (Höhe der Spanne) wird die Anzahl der insgesamt in der Matrix vorliegenden Lastspiele bestimmt.

Deklaration:

```
ClsOffFromRainflowGetSpans ( ClsHandle ) -> Spannenhäufigkeit
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Spannenhäufigkeit	Spannenhäufigkeit

Beschreibung:

Die Funktion ermittelt für jede Spanne (bzw. Amplitude) die Anzahl der Lastspiele. Grundlage dafür bildet die aktuell vorliegende Rainflow-Matrix.

Aus der Rainflow-Matrix wird ein Vektor gebildet.

Bei einer Amplituden-Mittelwert-Zählung werden jeweils alle zu einer Amplitude gezählten Lastspiele addiert. Die Information über den Mittelwert geht dabei verloren. Die x-Achse des Ergebnisses ist in Amplituden skaliert.

Bei einer Spannen-Mittelwert-Zählung werden jeweils alle zu einer Spanne gezählten Lastspiele addiert. Die Information über den Mittelwert geht dabei verloren. Die x-Achse des Ergebnisses ist in Spannen skaliert.

Bei einer Start-Zielklasse-Zählung wird die Spanne aus der Differenz von Start- und Zielklasse ermittelt. Für jede Spanne werden alle in der Rainflow-Matrix enthaltenen Lastspiele addiert. Die Richtung des Lastspiels geht dabei verloren. Die x-Achse des Ergebnisses ist in Spannen skaliert.

Die tatsächlich gemessene Anzahl von Lastspielen wird der Rainflow-Matrix entnommen. Das Residuum wird dabei nicht beachtet. Wenn es berücksichtigt werden soll, muss es vorher in die Matrix gezählt werden.

Wenn bei ClsOffRainflowInit1 ( ) eine Klassierung in physikalischen Einheiten gewählt wurde, ist auch die x-Achse des Ergebnisses in physikalischen Einheiten angegeben. Ansonsten in Klassen.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1)
ClsOffRainflowAddResidue ( ClsHandle, 1 )
Spans = ClsOffFromRainflowGetSpans ( ClsHandle )
```

Zuerst wird die Rainflow-Analyse initialisiert.

Eine Rainflow-Zählung wird vorgenommen. Die Anzahl der Spannen wird ermittelt.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowSetMatrix, ClsOffRainflowFeedSamples

ClsOffFromRainflowGetZeroCrossingPeak

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Spitzenwert I (zero crossing peak counting) aus Rainflow

Deklaration:

```
ClsOffFromRainflowGetZeroCrossingPeak ( ClsHandle, Bezug ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Bezug	der Wert des Signals, der als Bezug (Null-Linie) gelten soll.
Ergebnis	Verteilung

Beschreibung:

Aus der aktuell gehaltenen Rainflow-Matrix und dem aktuellen Residuum wird die Verteilung Spitzenwert I (zero crossing peak counting) ermittelt.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix)
ClsOffRainflowFeedResidue ( ClsHandle, Residue, 1.0)
_Bezug = 0.0 ; Bezugspegel für einige DIN-Verfahren
Spitzen1Off = ClsOffFromRainflowGetZeroCrossingPeak( ClsHandle, _Bezug)
```

Zuerst wird die Rainflow-Analyse initialisiert.

Eine bereits bestehende Matrix wird samt Residuum eingespeist.

Aus diesen Information wird das DIN-Verfahren abgeleitet und die gewünschte Verteilung bestimmt.

Siehe auch:

ClsOffRainflowInit1, KlsSpit1, ClsOffFromRainflowGetPeak, ClsOffFromRainflowGetLevelCrossing

ClsOffMarkov

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Berechnung der Markov-Matrix der Umkehrpunkte, Von-Bis-Zählung

Deklaration:

```
ClsOffMarkov ( Eingangsdaten, Min, Max, Klassen, Hysterese, EinheitOption ) -> Ergebnis
```

Parameter:

Eingangsdaten	Datensatz mit Eingangsdaten, von denen die Markov-Matrix zu berechnen ist.
Min	Minimum des Bereichs
Max	Maximum des Bereichs
Klassen	Anzahl Klassen
Hysterese	Hysterese zur Unterdrückung kleinerer Schwankungen, angegeben in physikalischen Einheiten
EinheitOption	
	0 : in Klassen beschriften
	1 : physikal. Einheit nutzen
	2 : in Klassen beschriften, aber auch kleine Spannen innerhalb einer Klasse zählen.
Ergebnis	Markov-Matrix

Beschreibung:

Für jede Spanne zwischen 2 benachbarten Extremwerten wird der Übergang von einer Klasse (Startklasse) in eine andere (Zielklasse) gezählt. Die Extremwerte werden dazu in Klassen einsortiert. Die gleich großen Klassen sind durch ihren Bereich (Min..Max) und ihre Anzahl festgelegt.

Von-Bis-Matrix, Übergangsmatrix, Transitionsmatrix, transition counting sind alternative Bezeichnungen der berechneten Markov-Matrix.

Das Ergebnis ist ein segmentierter Datensatz mit einer N mal N Matrix, wobei N die Anzahl der Klassen ist.

Die Segmente des Ergebnisses stellen die Zeilen der Matrix dar. Das 1. Segment entspricht der 1. Spalte.

Jeder Spalte ist eine Startklasse zugeordnet, jeder Zeile eine Zielklasse. So sind z.B. in der ersten Spalte (1. Segment) alle Übergänge von der 1. Startklasse in die 1. bis N-te Zielklasse eingetragen.

In x-Richtung des segmentierten Datensatzes ist also die Zielklasse aufgetragen. In z-Richtung ist die Startklasse aufgetragen.

Ansteigende Spannen werden also in die untere Dreiecksmatrix gezählt. Dabei ist der Zeilenindex größer als der Spaltenindex. Fallende Spannen gehen in die obere Dreiecksmatrix.

Die Diagonale der Matrix enthält stets nur Nullen bei EinheitOption gleich 1 oder 2.

Wenn Extremwerte außerhalb des Bereichs von Min und Max liegen, werden sie auf Min und Max begrenzt. Das entspricht offenen Randklassen.

Die Hysterese wird in der physikalischen Einheit der Eingangsdaten angegeben. Ein sinnvoller Wert für die Hysterese ist z.B. eine Klassenbreite.

Beispiele:

```
MarkovMatrix = ClsOffMarkov ( Data, 0.0, 1000.0, 100, 10.0, 1 )
```

Der Bereich von 0.0 bis 1000.0 wird in 100 Klassen eingeteilt, von denen jede die Breite 10 hat. Eine Hysterese von 10.0 wird angewendet, was einer vollen Klassenbreite entspricht.

Als Ergebnis werden diese ansteigenden Spannen ermittelt: 4 von Klasse 1 nach 2, 5 von 1 nach 3 und 2 von 2 nach 3. Außerdem werden diese fallenden Spannen ermittelt: 6 von 2 nach 1, 3 von 3 nach 1 und 1 von 3 nach 2:

Input Matrix:

0 6 3

4 0 1

5 2 0

Segment 1 = Spalte 1: 0 4 5

Segment 2 = Spalte 2: 6 0 2

Segment 3 = Spalte 3: 3 1 0

Z.B. ist MarkovMatrix[1,3] = 5. Also 5 Übergänge von 1 nach 3.

ClsOffMatrixSum1Triangle

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Dreiecksmatrix erstellen

Deklaration:

```
ClsOffMatrixSum1Triangle ( Matrix, Oben ) -> Ergebnis
```

Parameter:

Matrix	Matrix, z.B. Rainflow-Matrix
Oben	Obere oder untere Dreiecksmatrix bestimmen
	0 : Obere Dreiecksmatrix wird gefüllt. Von der ersten Spalte (1.Segment) wird nur das erste Element gefüllt. Die allerletzte Spalte wird komplett gefüllt.
	1 : Untere Dreiecksmatrix wird gefüllt. Die erste Spalte (1. Segment) also komplett. Von der letzten Spalte nur noch das letzte Element.
Ergebnis	Dreiecksmatrix

Beschreibung:

Von einer quadratischen Matrix werden die Elemente des oberen bzw. unteren Dreiecks den entsprechend gegenüberliegenden Elementen hinzuaddiert und selbst auf null gesetzt. Damit entsteht eine Dreiecksmatrix.

Die Funktion ist vor allem für Rainflow-Matrizen, bei denen die Spalten und Zeilen Start- und Zielklassen sind. Nach Ausführung der Funktion ist die Orientierung der Spannen nicht mehr erkennbar.

Prinzip des Algorithmus:

Für alle i > k

x[i,k] := x[i,k] + x[k,i]

x[k,i] := 0

Nur für quadratische Matrizen.

Beispiele:

```
RainflowDreieck = ClsOffMatrixSum1Triangle ( RainflowMatrix, 1 )
```

Alle Zyklen werden in das untere Dreieck gezählt.

Die Diagonale bleibt unverändert.

Input Matrix:

2 2 2

4 1 6

5 2 3

Segment1 = Spalte 1: 2 4 5

Segment2 = Spalte 2: 2 1 2

Segment3 = Spalte 3: 2 6 3

Output Matrix:

2 0 0

6 1 0

7 8 3

Segment1 = Spalte 1: 2 6 7

Segment2 = Spalte 2: 0 1 8

Segment3 = Spalte 3: 0 0 3

Siehe auch:

ClsOffRainflowGetMatrix, MatrixSumLines

ClsOffMatrixSumLines

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Zeilensummen, Spaltensummen einer Matrix
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten. Bitte benutzen Sie stattdessen MatrixSumLines().

Deklaration:

```
ClsOffMatrixSumLines ( Matrix, Zeilen ) -> Ergebnis
```

Parameter:

Matrix	Matrix, z.B. Rainflow-Matrix
Zeilen	Summe von Zeilen oder Spalten
	0 : Zeilen. Summe jeweils über den Inhalt einer Zeile. Die erste Zeile besteht aus dem 1. Element eines jeden Segmentes.
	1 : Spalten. Summe jeweils über den Inhalt einer Spalte, also eines Segmentes.
Ergebnis	Summe

Beschreibung:

Die Funktion bestimmt den Vektor der Zeilensummen oder der Spaltensummen einer Matrix.

Die Matrix muss nicht quadratisch sein.

Beispiele:

```
ColumnSum = ClsOffMatrixSumLines ( Matrix, 1 )
```

Input Matrix:

2 1 0

4 1 1

5 2 0

Segment 1 = Spalte 1: 2 4 5

Segment 2 = Spalte 2: 1 1 2

Segment 3 = Spalte 3: 0 1 0

Result ist dieser Vektor:

(11, 4, 1)

Bei diesem Vektor ist jedes Element je die Summe einer Spalte der Matrix

```
RowSum = ClsOffMatrixSumLines ( Matrix, 0 )
```

Das liefert (3, 6, 7).

Siehe auch:

MatrixSumLines

ClsOffRainflowAddResidue

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Das aktuell gehaltene Residuum wird in die Matrix hineingezählt und anschließend geleert.

Deklaration:

```
ClsOffRainflowAddResidue ( ClsHandle, Gewichtsfaktor )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Gewichtsfaktor	0... 1: Bei Zweifel auf 1 setzen. Während dieser Funktion wird in der Matrix nicht um 1, sondern um diesen Faktor hochgezählt. Damit kann das Residuum schwächer als 1 bewertet werden.

Beschreibung:

Das Residuum wird in die Matrix hineingezählt. Jede gefundene Schwingung wird aber mit dem Gewichtsfaktor bewertet.

Damit kann beeinflusst werden, um welchen Betrag entsprechende Stellen der Matrix inkrementiert werden.

Gewichtsfaktor = 0.0: Matrix bleibt unverändert

Gewichtsfaktor = 1.0: worst case Annahme. Jede Schwingung zählt voll. Auch in Wirklichkeit unvollendete Schwingspiele werden damit als vollendet angenommen.

Gewichtsfaktor = 0.5: Kompromiss. Alle Schwingspiele im Residuum können als halbe Schwingungen angesehen werden.

Der Gewichtsfaktor darf jeden beliebigen Wert annehmen.

Das Residuum wird in die Matrix hineingezählt, indem das Residuum selbst noch einmal als Zeitdatensatz in den Zählalgorithmus eingespeist wird.

Damit das Residuum komplett berücksichtigt wird, werden ggf. geeignete Zwischenwerte zu Hilfe genommen (d.h. eingefügt).

Ein Residuum der Länge 1 wird ignoriert.

Danach wird das Residuum geleert.

Eine Diskontinuität wird vorher und nachher eingefügt.

Auch bei ASTM E1049 muss die Funktion aufgerufen werden. Das Residuum wird dabei mit Gewicht = 0.5 in die Matrix gezählt. Der Parameter Gewichtsfaktor wird ignoriert und darf auf 0 gesetzt werden.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1)
ClsOffRainflowAddResidue ( ClsHandle, 1 )
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
```

Für einen Datensatz data_chan1 wird eine Rainflow-Zählung durchgeführt.

Nach dem Zählen aller Spannenpaare in die Matrix bleibt ein Residuum übrig. Das soll allerdings nicht separat ausgewiesen werden. Deshalb wird es in die Matrix hineingezählt.

Die Matrix wird also an einigen Stellen inkrementiert.

Danach ist dann das Residuum leer und braucht nicht abgefragt zu werden.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowFeedResidue, ClsOffRainflowGetResidue, ClsOffRainflowFeedMatrix

ClsOffRainflowClearMatrix

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Die aktuell gehaltene Matrix wird auf null gesetzt.

Deklaration:

```
ClsOffRainflowClearMatrix ( ClsHandle )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz

Beschreibung:

Alle vorher gezählten Werte gehen damit verloren. Das aktuelle Residuum bleibt unverändert.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
... ; weitere Operationen
ClsOffRainflowClearMatrix ( ClsHandle )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix2)
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
```

Hier wird mit *ClearMatrix() die aktuelle Matrix geleert. Dann wird mit *FeedMatrix() eine neue dazugezählt.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowSetMatrix, ClsOffRainflowClearResidue

ClsOffRainflowClearResidue

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Das aktuell gehaltene Residuum wird geleert.

Deklaration:

```
ClsOffRainflowClearResidue ( ClsHandle )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz

Beschreibung:

Das Residuum wird nicht in die Matrix hineingezählt, sondern einfach gelöscht. Damit geht es verloren.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1)
ClsOffRainflowClearResidue ( ClsHandle )
...
```

Die Messwerte data_chan1 werden in die Matrix gezählt.

Danach wird das Residuum gelöscht.

Anschließend wird weiteres gemacht, wo das Residuum sich nicht mehr auswirken soll.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowGetResidue, ClsOffRainflowFeedResidue, ClsOffRainflowAddResidue, ClsOffRainflowClearMatrix

ClsOffRainflowFeedDiscontinuity

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Eine Diskontinuität wird eingefügt.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten.

Deklaration:

```
ClsOffRainflowFeedDiscontinuity ( ClsHandle )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz

Beschreibung:

Anschließend eingespeiste Messwerte werden als Daten einer neuen Messung betrachtet, die nicht lückenlos an die alten Messwerte anschließt.

Siehe auch:

ClsOffRainflowFeedSamples

ClsOffRainflowFeedMatrix

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Die aktuelle Matrix wird um die Werte der übergebenen Matrix hochgezählt. Das aktuelle Residuum bleibt unverändert.

Deklaration:

```
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Matrix	Die zu addierende Matrix, muss dieselbe Dimension haben wie die intern gehaltene aktuelle Matrix.

Beschreibung:

Die aktuelle Matrix wird Wert für Wert inkrementiert.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix1)
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix2)
ClsOffRainflowFeedResidue ( ClsHandle, Residue2, 1.0)
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
Residue = ClsOffRainflowGetResidue ( ClsHandle )
```

Es liegen bereits für einen Kanal mehrere Klassierergebnisse vor. Von diesen alten Messungen ist noch die Rainflow-Matrix (und das Residuum) jedes Mal da.

Die Matrizen werden eingespeist (und damit zusammen addiert).

Falls vorhanden, werden auch die Residuen eingespeist. Hier gab es nur ein Residuum zur 2. Messung.

Die Gesamtbeanspruchung wird dann abgeholt.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowFeedResidue, ClsOffRainflowSetMatrix

ClsOffRainflowFeedResidue

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Ein Residuum, das vorher einmal ermittelt wurde, wird in die aktuelle Matrix gezählt. Dabei werden die aktuelle Matrix und das aktuelle Residuum verändert.

Deklaration:

```
ClsOffRainflowFeedResidue ( ClsHandle, Residue, Gewichtsfaktor )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Residue	Der Datensatz mit dem Residuum.
Gewichtsfaktor	0... 1: Bei Zweifel auf 1 setzen. Während dieser Funktion wird in der Matrix nicht um 1, sondern um diesen Faktor hochgezählt. Damit kann das Residuum schwächer als 1 bewertet werden.

Beschreibung:

Zu beachten: Die Hysterese wird unterhalb Präzise=6 ignoriert. Die y-Skalierung des Residuums wird entsprechend der EinheitNutzung aus der ClsOffRainflowInit1() Funktion verstanden.

Das Residuum wird inklusive der evtl. erforderlichen künstlichen Zwischenwerte eingespeist (Präzise unterhalb 5). Ist das intern vorhandene aktuelle Residuum leer, werden keine Zwischenwerte eingefügt.

Wenn das Residuum in physikalischen Einheiten vorliegt, sollten Klassenmitten benutzt werden. In der Tat werden beim Residuum in physikalischen Einheiten die Klassen um 1/1000 der Klassenbreite nach unten verschoben (Präzise unterhalb 5).

Das Residuum muss gültig sein. D.h. z.B. nur Werte im erlaubten Bereich, jeweils benachbarte Werte in unterschiedlichen Klassen. Ungültige Residuen führen zu unerwarteten Resultaten! Ein Residuum der Länge 1 wird ignoriert (Präzise unterhalb 6).

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix1)
ClsOffRainflowFeedResidue ( ClsHandle, Residue1, 1.0)
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix2)
ClsOffRainflowFeedResidue ( ClsHandle, Residue2, 1.0)
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
Residue = ClsOffRainflowGetResidue ( ClsHandle )
```

Es liegen bereits für einen Kanal mehrere Klassierergebnisse vor. Von diesen alten Messungen ist noch die Rainflow-Matrix und das Residuum jedes Mal da. Neben den Matrizen werden die Residuen auch eingespeist. Alles wird zusammengezählt. Die Gesamtbeanspruchung wird dann abgeholt.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowFeedMatrix, ClsOffRainflowGetResidue, ClsOffRainflowFeedSamples

ClsOffRainflowFeedSamples

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Neue Messwerte werden in die Matrix gezählt. Die Matrix und das Residuum werden entsprechend aktualisiert.

Deklaration:

```
ClsOffRainflowFeedSamples ( ClsHandle, Samples )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Samples	ein oder mehrere neue Messwerte; Beanspruchungs-Zeit-Funktion; Lastkollektiv

Beschreibung:

Es ist besonders effektiv, zuerst mehrere Messwerte zu einem Datensatz zu verbinden und dann auf einmal dieser Funktion zu übergeben.

Mit der Funktion ClsOffTM() kann die Genauigkeit verbessert werden.

Die neuen Messwerte so verstanden, als ob sie nahtlos an die zuletzt eingespeisten Messwerte angefügt wären. Damit werden auch Extremwerte direkt am Übergang sicher erkannt.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_test1)
ClsOffRainflowFeedSamples ( ClsHandle, data_test2)
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
delete ClsHandle
```

Eine Rainflow-Analyse wird initialisiert und durchgeführt. Die Messdateien wurden aufgespalten, weil jede nur die Messung über 1 Stunde enthält. Jede der Messdatensätze wird eingespeist und gezählt.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowInit2, ClsOffRainflowInit3, ClsOffRainflowGetMatrix, ClsOffTM

ClsOffRainflowGetMatrix

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Die aktuell gehaltene Matrix wird zurückgegeben.

Deklaration:

```
ClsOffRainflowGetMatrix ( ClsHandle ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Ergebnis	Matrix

Beschreibung:

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1)
ClsOffRainflowAddResidue ( ClsHandle, 1 )
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
```

Eine Rainflow-Zählung wird durchgeführt.

Anschließend wird das Ergebnis, nämlich die Matrix abgeholt.

Da hier kein separates Residuum gewünscht wurde, wurde das Residuum noch vorher in die aktuelle Matrix hineingezählt.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowSetMatrix, ClsOffRainflowGetResidue

ClsOffRainflowGetResidue

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Das aktuell gehaltene Residuum wird zurückgegeben.

Deklaration:

```
ClsOffRainflowGetResidue ( ClsHandle ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Ergebnis	Residuum

Beschreibung:

Wenn das Residuum in physikalischen Einheiten bestimmt wird, dann werden Klassenmitten zurückgegeben.

Ansonsten wird das Residuum in Klassen (0, 1, 2, ...) zurückgegeben.

Die x-Achse des Residuums ist ohne Bedeutung. Sie listet nur die Werte auf.

Je zwei aufeinander folgende Werte des Residuums bilden eine Spanne.

Bei Präzise = 4 sind die Werte des Residuums nicht gerundet und können beliebige Werte zwischen den Klassengrenzen annehmen.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1)
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
Residue = ClsOffRainflowGetResidue ( ClsHandle )
```

Für einen Datensatz data_chan1 wird die Rainflow-Zählung durchgeführt.

Die Matrix und das Residuum werden ermittelt.

Das Residuum ist nicht in die Matrix eingerechnet worden, sondern wird separat ausgewiesen.

Das Residuum wird als Vektor angegeben, in dem alle restlichen Extremwerte stehen. Die Spannen zwischen den Extremwerten sind direkt bei grafischer Darstellung ablesbar.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowGetResidue, ClsOffRainflowGetResidueMtx, ClsOffRainflowFeedResidue, ClsOffRainflowAddResidue

ClsOffRainflowGetResidueMtx

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Das aktuell gehaltene Residuum wird in Matrixform zurückgegeben.

Deklaration:

```
ClsOffRainflowGetResidueMtx ( ClsHandle ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Ergebnis	Matrix

Beschreibung:

Dabei ist die Matrix genauso aufgebaut wie die Klassiermatrix.

Der Zählerstand allerdings gibt die Häufigkeit einer Spanne im Residuum an.

Der Wert ist also 0, wenn diese Spanne im Residuum nicht auftritt.

Der Wert ist z.B. 1, wenn die Spanne einmal aufgetreten ist.

Der Wert kann auch größer als 1 werden.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1)
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
ResidueMtx = ClsOffRainflowGetResidueMtx ( ClsHandle )
```

Für einen Datensatz data_chan1 wird die Rainflow-Zählung durchgeführt.

Die Matrix und das Residuum werden ermittelt.

Das Residuum ist nicht in die Matrix eingerechnet worden, sondern wird separat ausgewiesen.

Das Residuum wird als Matrix bestimmt, in der alle Spannen markiert sind.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowGetResidue, ClsOffRainflowFeedResidue, ClsOffRainflowAddResidue, ClsOffRainflowGetMatrix

ClsOffRainflowInit1

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Initialisierung einer Rainflow-Zählung. Immer als erste Funktion aufrufen.

Deklaration:

```
ClsOffRainflowInit1 ( Klassen, EinheitNutzung, EinheitSpalte, EinheitZeile, EinheitZaehler, Einheit_Y_Residuum, EW_0 ) -> Ergebnis
```

Parameter:

Klassen	Die Anzahl der Klassen, z.B. 32 oder 64. >= 4 und <= 1000
EinheitNutzung	Wie werden die Klassenbreite der Matrix und anderer Ergebnisse und wie werden die Werte des Residuums skaliert?
	0 : Klassen (0, 1, ...)
	1 : Skalierung in der physikalischen Einheit der Messgröße
EinheitSpalte	Diese Enheit wird in x-Richtung an die Matrix geschrieben und beschreibt damit die Elemente entlang einer Spalte der Matrix.
EinheitZeile	Diese Enheit wird in z-Richtung an die Matrix geschrieben und beschreibt damit die Elemente entlang einer Zeile der Matrix.
EinheitZaehler	Die Einheit für die y-Achse der Matrix, die die Anzahl der Schwingungen zählt.
Einheit_Y_Residuum	Die Einheit für die y-Achse des Residuums. Wenn aus der Rainflow-Matrix andere Zählverfahren bestimmt werden, wird diese Einheit für die x-Achse benutzt.
EW_0	Immer = 0. Reserviert für später.
Ergebnis	Der Rückgabewert wird in allen anderen ClsOffRainflow* Funktionen benutzt und darf nicht verändert werden. Er wird aber von den Funktionen selbst verändert.

Beschreibung:

Alle Einheiten können auch auf leere Texte gesetzt werden.

Beispiele:

```
_NumberClasses = 50
_TypeOfUnit = 1 ; 0 Klassen, 1 physikalische Einheit
_UnitX = "Mitte [Nm]"
_UnitZ = "Spanne [Nm]"
_UnitCount = "Werte"
_UnitRes = "Klassen"
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitX, _UnitZ, _UnitCount, _UnitRes, 0)
_Min = -5
_Max = 5
_Hysteresis = ( _Max - _Min ) / _NumberClasses
_Axis = 1 ; 0 x ist Zielklasse oder Amplitude, 1 x ist Startklasse oder Mittelwert
_Type = 2 ; 0 Start- und Zielklasse, 1 Amplitude und Mittelwert, 2 Spanne und Mittelwert
_CalcOptions = 0 ; 0 Basis Algorithmus, 1 Clormann Seeger Korrektur, 2 ASTM E1049, 3 HCM
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1)
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
delete ClsHandle
```

Eine Rainflow-Analyse wird initialisiert und durchgeführt.

Siehe auch:

ClsOffRainflowInit2, ClsOffRainflowInit3

ClsOffRainflowInit2

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Fortsetzung der Initialisierung der Rainflow-Zählung. Immer direkt hinter ClsOffRainflowInit1 () aufrufen. Setzt die aktuelle Matrix auf null und leert das Residuum.

Deklaration:

```
ClsOffRainflowInit2 ( ClsHandle, Min, Max, Hysterese, Achsen, Typ, Randklassen, Berechnung )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Min	Unteres Ende des Klassierbereichs
Max	Oberes Ende des Klassierbereichs. Min .. Max: Der Bereich des zu klassierenden Signals. Das legt den Wertebereich der Matrix fest. Max > Min. Dieser Bereich sollten vollen Amplituden-Bereich des Eingangssignals abdecken, i. Allg. mit etwas Reserve. Beispiel: Min = -1Nm und Max = +1Nm bei 32 Klassen führt auf sehr krumme Klassenbreiten, nämlich 2/32 Nm. Wenn die Klassenbreite einen runden Wert haben soll, dann Min und Max etwas aufrunden, z.B. +- 1.28 Nm oder +- 1.6Nm.
Hysterese	Hysterese, in physikal. Einheiten, nicht in Prozent der Klassenbreite. Betrag der Hysterese, die bei der Extremwertsuche angewendet wird. >= 0. Empfohlen ist ein Wert von einer Klassenbreite. Bei Zweifel auf 0 setzen.
Achsen	Was ist an der x-Achse aufgetragen?
	0 : Entlang der Spalten (x-Richtung) ist die Zielklasse oder Amplitude aufgetragen. Die x-Achse kann also mit Zielklasse oder Amplitude bezeichnet werden. Der Index eines Elementes innerhalb einer Spalte gibt an, welche Zielklasse oder Amplitude vorliegt. Der Mittelwert bzw. die Startklasse wählt aus, welche Spalte gemeint ist. Eine Spalte der Matrix ist ein Segment.
	1 : Entlang der Spalten (x-Richtung) ist die Startklasse oder der Mittelwert aufgetragen. Also genau umgekehrt.
Typ	Was bedeuten die x- und z-Achse?
	0 : Start- und Zielklasse. Das ist der eigentliche Zähl-Algorithmus. Die Startklasse ist die Klasse des ersten Extremwertes einer Schwingung, die Zielklasse die des zweiten. Es wird unterschieden zwischen einer Schwingung von 10 nach 16 und einer von 16 nach 10.
	1 : Amplitude und Mittelwert. Der Mittelwert ist die Mittel-Lage einer Schwingung, also (Startklasse + Zielklasse) / 2. Die Amplitude ist der Betrag von (Startklasse-Zielklasse) / 2. Da genauso viele Amplituden-Klassen wie Mittelwert-Klassen vorliegen (die Matrix ist quadratisch), ist die Auflösung der Amplituden doppelt so gut. Die Amplitude ist der halbe Wert von [peak to peak] (Spitze-Spitze).
	2 : Spanne und Mittelwert. Der Mittelwert ist die Mittel-Lage einer Schwingung, also (Startklasse + Zielklasse) / 2. Die Spanne ist der Betrag von (Startklasse-Zielklasse).
Randklassen	Behandlung von Werten außerhalb
	0 : Geschlossen. Vor der Extremwertsuche wird das Signal auf den Bereich Min..Max eingeschränkt.
	1 : Offen. Vor der Extremwertsuche wird das Signal auf den Bereich Min..Max eingeschränkt. Empfohlen
Berechnung	Algorithmus
	0 : Standard-Algorithmus 4 Punkte
	1 : Clormann Seeger Korrektur. Berücksichtigung der Null-Lagen-Abhängigkeit. Zählt Übergang x[1] -> x[2], falls 1) Von x[1] und x[2] ist eines > 0 und eines < 0; 2) Betrag (x[2]) <= Betrag (x[1]); 3) Betrag (x[1]) <= Betrag (x[3]).
	2 : Berechnung nach ASTM E1049, reapproved 1990
	3 : RAINFLOW-HCM Verfahren. U.H. Clormann, T. Seeger, Stahlbau 3/1986

Beschreibung:

Bei ASTM E1049 wird die vor Schritt (6) entstandene Folge als Residuum verstanden. Die Funktion ClsOffRainflowAddResidue muss aufgerufen werden, um die Matrix nach der Norm zu erhalten!

Bei ASTM E1049 wird das Residuum mit Gewicht 0.5 in die Matrix gezählt. Die Option "Präzise" muss wenigstens 5 sein.

Bei RAINFLOW-HCM muss die Option "Präzise" mindestens 5 sein.

Beispiele:

```
_NumberClasses = 50
_TypeOfUnit = 1 ; 0 Klassen, 1 physikalische Einheit
_UnitX = "Mitte [Nm]"
_UnitZ = "Spanne [Nm]"
_UnitCount = "Werte"
_UnitRes = "Klassen"
```

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitX, _UnitZ, _UnitCount, _UnitRes, 0)
_Min = -5
_Max = 5
_Hysteresis = ( _Max - _Min ) / _NumberClasses
_Axis = 1 ; 0 x ist Zielklasse oder Amplitude, 1 x ist Startklasse oder Mittelwert
_Type = 2 ; 0 Start- und Zielklasse, 1 Amplitude und Mittelwert, 2 Spanne und Mittelwert
_CalcOptions = 0 ; 0 Basis Algorithmus, 1 Clormann Seeger Korrektur, 2 ASTM E1049, 3 HCM
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1)
RainflowMatrix1 = ClsOffRainflowGetMatrix ( ClsHandle )

ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan2)
RainflowMatrix2 = ClsOffRainflowGetMatrix ( ClsHandle )
delete ClsHandle
```

Eine Rainflow-Analyse wird initialisiert und durchgeführt. Anschließend wird ein anderer Kanal bearbeitet.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowInit3

ClsOffRainflowInit3

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Fortsetzung der Initialisierung der Rainflow-Zählung. Immer direkt hinter ClsOffRainflowInit2 () aufrufen.

Deklaration:

```
ClsOffRainflowInit3 ( ClsHandle, IgnoriereKleineSpannen, Präzise, StartEndeZählen, EW_Null1, EW_Null2, EW_Null3
)
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
IgnoriereKleineSpannen	Sind kleine Spannen, die sich innerhalb einer Klasse abspielen, zu ignorieren?
	0 : Beachten. Die kleinen Spannen werden auch gezählt.
	1 : Ignorieren: Die kleinen Spannen werden nicht gezählt.
Präzise	Genaue Berechnung der Spannen? Höchster Wert empfohlen
	0 : kompatibel: Extremwerte werden erst in Klassen eingeteilt. Dann wird bei Amplituden/Mittelwertberechnung die Amplitude bzw. Spanne als Differenz bestimmt. Das ist nicht ganz präzise.
	1 : Präzise Berechnung der Spannen und Mittelwerte und anschließende Einteilung in Klassen.
	2 : Präzise Berechnung der Spannen und Mittelwerte und anschließende Einteilung in Klassen. Außerdem präziseres Entfernen aus dem Residuum.
	3 : Präzise Berechnung der Spannen und Mittelwerte und anschließende Einteilung in Klassen. Außerdem präziseres Entfernen aus dem Residuum, sogar bei abklingendem Residuum
	4 : Wie Präzise 3, aber zusätzlich präzises (ungerundetes) Residuum, ohne Runden auch bei mehreren Aufrufen von ClsOffRainflowFeedSamples
	5 : Wie Präzise 4, aber erhöhte Genauigkeit
	6 : Wie Präzise 5, aber erhöhte Genauigkeit auch bei Randwerten
StartEndeZählen	Randwerte der Zeitdaten beachten?
	0 : Randwerte der Zeitdaten werden nicht weiter beachtet.
	1 : Die Randwerte der Zeitdaten werden als Extremwerte gezählt. Empfohlen!
EW_Null1	0
EW_Null2	0
EW_Null3	0

Beschreibung:

Präzise = 6 ist empfohlen.

Bei ASTM E1049 bzw. HCM muss der Parameter "Präzise" mindestens 5 sein.

Beispiele:

```
_IgnoreSmallSpans = 1 ; 1 Kleine Spannen ignorieren, 0 zählen
_Precise = 6
_CountStartEnd = 0 ; 1 Start- und Endwert als Extremwert zählen. 0 nicht
ClsOffRainflowInit3 ( ClsHandle, _IgnoreSmallSpans, _Precise, _CountStartEnd, 0, 0, 0 )
```

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowInit2, ClsOffRainflowFeedSamples

ClsOffRainflowSetMatrix

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Die aktuelle Matrix wird durch die neue Matrix ersetzt.

Deklaration:

```
ClsOffRainflowSetMatrix ( ClsHandle, Matrix )
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Matrix	Die neue Matrix, muss dieselbe Dimension haben wie die intern gehaltene aktuelle Matrix.

Beschreibung:

Alle alten bislang gezählten Werte in der Matrix gehen verloren.

Das aktuelle Residuum bleibt unverändert.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
... ; weitere Operationen
ClsOffRainflowSetMatrix ( ClsHandle, Matrix1)
ClsOffRainflowFeedMatrix ( ClsHandle, Matrix2)
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
```

Hier wird mit *SetMatrix() die aktuelle Matrix gesetzt. Dann wird mit *FeedMatrix() eine neue dazugezählt.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowGetMatrix, ClsOffRainflowFeedMatrix

ClsOffRevolutionsHistogram

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Die Anzahl der Überrollungen wird in Abhängigkeit vom Kanal gezählt.

Deklaration:

```
ClsOffRevolutionsHistogram ( Kanal, Drehzahl, Min, Max, Klassen, Randklassen, EinheitOption ) -> Ergebnis
```

Parameter:

Kanal	Der Eingangskanal (z.B. Zeitverlauf des Moments). Dieser Kanal wird in Klassen eingeteilt.
Drehzahl	Aus diesem Kanal werden die Überrollungen berechnet. Die Drehzahl ist immer in (Umdrehungen pro Minute: "upm" bzw. "U/min" bzw. "rpm") skaliert.
Min	Minimum für den 1. Kanal
Max	Maximum für den 1. Kanal
Klassen	Anzahl Klassen für den 1. Kanal
Randklassen	
	0 : Randklassen geschlossen
	1 : Randklassen offen
EinheitOption	
	0 : in Klassen beschriften
	1 : physikal. Einheit nutzen
Ergebnis	Umdrehungshistogramm

Beschreibung:

Die Überrollungen werden dabei aus der Drehzahl berechnet.

Jeder Messwert des Kanals wird dabei einer der Klassen zugeordnet, wobei in die Klasse selbst die aus dem entsprechenden Messwert der Drehzahl gewonnene Anzahl von Überrollungen gezählt wird.

Die Klassen selbst werden durch den Bereich von Min .. Max mit der Anzahl 'Klassen' gebildet.

Von der Drehzahl wird der Absolutbetrag gebildet.

Erst danach werden die Überrollungen berechnet.

Eine Überrollung ist dabei eine vollständige Umdrehung der drehende Achse.

Wenn sich eine Achse mit 6000U/min dreht, dann entspricht das 100 Überrollungen innerhalb von 1 Sekunde bzw. 6000 Überrollungen innerhalb von 1 Minute.

Innerhalb des Abtastintervalls werden beide Kanäle als konstant angenommen.

Min, Max, Klassen: Für den ersten Kanal der Bereich. Klassierung von Min (<Max) .. Max mit einer Klassenanzahl Klassen (>=4).

Die Randklassen können offen oder geschlossen sein.

Bei offenen Randklassen werden Werte außerhalb des Bereich von Min..Max in die äußerste Klasse (Randklasse) einsortiert.

Bei geschlossenen Randklassen werden Werte, die außerhalb des Bereichs Min..Max liegen, ignoriert, also nicht gezählt.

Die Beschriftung der x-Achse kann in Klassen erfolgen (Klasse 0, 1, 2, ...) oder in den physikalischen Einheiten des Eingangskanals (Moment), z.B. -0.1 Nm .. +0.1 Nm.

Beispiele:

```
MinTo = 0   ;Nm
MaxTo = 200 ; Nm
KlassenTo = 40
RandOffen = 1 ; Randklassen, 0 geschlossen, 1 offen
EinheitOption = 1 ; 0 in Klassen, 1 in physikalischen Einheiten
RevHistogram = ClsOffRevolutionsHistogram ( Torque, RPMs, MinTo, MaxTo, KlassenTo, RandOffen, EinheitOption)
```

Ein Drehzahlverlauf RPMs (in Umdrehungen pro Minute) und ein Drehmomentverlauf Torque (in Nm) werden als Eingangskanäle für die Zählung der Überrollungen benutzt.

Dabei wird der Drehmomentbereich von 0Nm .. 200Nm in 40 Klassen eingeteilt.

Offene Randklassen und eine Beschriftung des entstehenden Histogramms in physikalischen Einheiten wird gewählt.

Siehe auch:

ClsTimeAtLevel, ClsOff2ChannelHistogram, ClsOffRevolutionsMatrix

ClsOffRevolutionsMatrix

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Die Anzahl der Überrollungen wird in Abhängigkeit von Kanal und von der Drehzahl in eine Matrix gezählt.

Deklaration:

```
ClsOffRevolutionsMatrix ( Kanal, Drehzahl, Min, Max, Klassen, MinDrehzahl, MaxDrehzahl, KlassenDrehzahl, Optionen ) -> Ergebnis
```

Parameter:

Kanal	Der Eingangskanal (z.B. Zeitverlauf des Moments). Dieser Kanal wird in Klassen eingeteilt.
Drehzahl	Aus diesem Kanal werden die Überrollungen berechnet. Die Drehzahl ist immer in (Umdrehungen pro Minute: "upm" bzw. "U/min" bzw. "rpm") skaliert. Die Drehzahl selbst wird auch in Klassen eingeteilt.
Min	Minimum für den 1. Kanal
Max	Maximum für den 1. Kanal
Klassen	Anzahl Klassen für den 1. Kanal
MinDrehzahl	Minimum der Drehzahl
MaxDrehzahl	Maximum der Drehzahl
KlassenDrehzahl	Anzahl Klassen für die Drehzahl
Optionen	Optionen für Rand, Einheit und Orientierung
	0 : Randklassen geschlossen, in Klassen beschriften, Drehzahl in z-Richtung
	1 : Randklassen offen, in Klassen beschriften, Drehzahl in z-Richtung
	2 : Randklassen geschlossen, physikal. Einheit nutzen, Drehzahl in z-Richtung
	3 : Randklassen offen, physikal. Einheit nutzen, Drehzahl in z-Richtung
	4 : Randklassen geschlossen, in Klassen beschriften, Drehzahl in x-Richtung
	5 : Randklassen offen, in Klassen beschriften, Drehzahl in x-Richtung
	6 : Randklassen geschlossen, physikal. Einheit nutzen, Drehzahl in x-Richtung
	7 : Randklassen offen, physikal. Einheit nutzen, Drehzahl in x-Richtung
Ergebnis	Zweidimensionales Histogramm

Beschreibung:

Die Überrollungen werden dabei aus der Drehzahl berechnet.

Jeder Messwert des Kanals wird dabei einer der Klassen zugeordnet und jeder zugehörige Messwert der Drehzahl wird einer der Drehzahlklassen zugeordnet, wobei in die Klasse selbst die aus dem entsprechenden Messwert der Drehzahl gewonnene Anzahl von Überrollungen gezählt wird. Dabei entsteht eine Matrix.

Von der Drehzahl wird der Absolutbetrag gebildet.

Erst danach werden die Umdrehungen berechnet und auch die Klasseneinteilung durchgeführt.

Eine Überrollung ist dabei eine vollständige Umdrehung der drehende Achse.

Wenn sich eine Achse mit 6000U/min dreht, dann entspricht das 100 Überrollungen innerhalb von 1 Sekunde bzw. 6000 Überrollungen innerhalb von 1 Minute.

Innerhalb des Abtastintervalls werden beide Kanäle als konstant angenommen.

Die Randklassen können offen oder geschlossen sein.

Bei offenen Randklassen werden Werte außerhalb des Bereich von Min..Max in die äußerste Klasse (Randklasse) einsortiert.

Bei geschlossenen Randklassen werden Werte, die außerhalb des Bereichs Min..Max liegen, ignoriert, also nicht gezählt.

Die Beschriftung der Achsen (x-Achse und z-Achse) kann in Klassen erfolgen (Klasse 0, 1, 2, ...) oder in den physikalischen Einheiten der Eingangskanäle (z.B. -0.1 Nm .. +0.1 Nm).

Ferner kann festgelegt werden, welche der Eingangskanäle in die Spalten und welcher in die Zeilen der entstehenden Matrix einsortiert wird.

Wird die Drehzahl in z-Richtung einsortiert, dann ist die z-Achse bei Matrix-Darstellung die Drehzahl-Achse, der 1. Eingangskanal ist dann in x-Richtung aufgetragen.

Ein Auftragen in x-Richtung ist das Einsortieren in eine Spalte der Matrix (ein Segment des entstehenden Datensatzes).

Min, Max, Klassen: Für den ersten Kanal der Bereich. Klassierung von Min (<Max) .. Max mit einer Klassenanzahl Klassen (>=4).

MinDrehzahl, MaxDrehzahl, KlassenDrehzahl: Für den Drehzahlkanal der Bereich. Klassierung von MinDrehzahl (<MaxDrehzahl) .. MaxDrehzahl mit einer Klassenanzahl KlassenDrehzahl (>=4).

Beispiele:

```
MinTo = 0   ;Nm
MaxTo = 200 ; Nm
KlassenTo = 40
MinDrehzahl = 0   ;U/min
MaxDrehzahl = 6000 ; U/min
KlassenDrehzahl = 30
Option = 3 ; Randklassen offen, physikal. Einheit nutzen, Drehzahl in z-Richtung
RevMatrix = ClsOffRevolutionsMatrix ( Torque, RPMs, MinTo, MaxTo, KlassenTo, MinDrehzahl, MaxDrehzahl, KlassenDrehzahl, Option )
```

Ein Drehzahlverlauf RPMs (in Umdrehungen pro Minute) und ein Drehmomentverlauf Torque (in Nm) werden als Eingangskanäle für die Zählung der Überrollungen benutzt.

Dabei wird der Drehmomentbereich von 0 .. 200Nm in 40 Klassen eingeteilt.

Die Drehzahl wird in 30 Klassen von 0 .. 6000U/min eingeteilt.

Siehe auch:

ClsTimeAtLevel, ClsOff2ChannelHistogram, ClsOffRevolutionsHistogram, ClsOffRevolutionsMatrix2

ClsOffRevolutionsMatrix2

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Die Anzahl der Überrollungen wird in Abhängigkeit von Kanal1 und Kanal2 gezählt.

Deklaration:

```
ClsOffRevolutionsMatrix2 ( Kanal1, Kanal2, Drehzahl, MinKanal1, MaxKanal1, KlassenKanal1, MinKanal2, MaxKanal2, KlassenKanal2 ) -> Ergebnis
```

Parameter:

Kanal1	Der 1. Eingangskanal (z.B. Zeitverlauf des Moments).
Kanal2	Der 2. Eingangskanal (z.B. Zeitverlauf der Temperatur).
Drehzahl	Aus diesem Drehzahlkanal werden die Überrollungen berechnet.
MinKanal1	Unteres Ende des Klassierbereichs des 1. Kanals
MaxKanal1	Oberes Ende des Klassierbereichs des 1. Kanals
KlassenKanal1	Anzahl der Klassen für den 1. Kanal, >= 4
MinKanal2	Unteres Ende des Klassierbereichs des 2. Kanals
MaxKanal2	Oberes Ende des Klassierbereichs des 2. Kanals
KlassenKanal2	Anzahl der Klassen für den 2. Kanal, >= 4
Ergebnis	Umdrehungsmatrix

Beschreibung:

Die Anzahl der Überrollungen wird in Abhängigkeit von Kanal1 und Kanal2 gezählt.

Die Überrollungen werden dabei aus der Drehzahl berechnet.

Jeder Messwert eines Kanals wird dabei einer der Klassen zugeordnet, wobei in die Klasse selbst die aus dem entsprechenden Messwert der Drehzahl gewonnene Anzahl von Überrollungen gezählt wird.

Dabei entsteht eine Matrix.

Von der Drehzahl wird der Absolutbetrag gebildet.

Erst danach werden die Umdrehungen berechnet.

Eine Überrollung ist dabei eine vollständige Umdrehung der drehende Achse.

Wenn sich eine Achse mit 6000U/min dreht, dann entspricht das 100 Überrollungen innerhalb von 1 Sekunde bzw. 6000 Überrollungen innerhalb von 1 Minute.

Die Drehzahl ist immer in (Umdrehungen pro Minute: "upm" bzw. "U/min" bzw. "rpm") skaliert.

Innerhalb des Abtastintervalls werden beide Kanäle als konstant angenommen.

Die Randklassen sind geschlossen, d.h. Werte, die außerhalb des Bereichs Min..Max liegen, werden ignoriert, also nicht gezählt.

Die Beschriftung der Matrix (x- und z-Richtung) erfolgt in den physikalischen Einheiten der Eingangskanäle.

Der 1. Kanal wird in x-Richtung, also in eine Spalte der Matrix (ein Segment) einsortiert.

Der 2. Kanal bildet die Zeile (z-Richtung).

Beispiele:

```
Min1 = 0   ; Nm
Max1 = 200 ; Nm
Klassen1 = 40
Min2 = 0   ; Nm
Max2 = 300 ; Nm
Klassen2 = 60
RevMatrix = ClsOffRevolutionsMatrix2 ( Torque1, Torque2, RPMs, Min1, Max1, Klassen1, Min2, Max2, Klassen2 )
```

Ein Drehzahlverlauf RPMs (in Umdrehungen pro Minute) und 2 Drehmomentverläufe Torque1 und Torque2 (in Nm) werden als Eingangskanäle für die Zählung der Überrollungen benutzt.

Dabei wird der Drehmomentbereich von 0 .. 200Nm in 40 Klassen für den 1. Kanal und von 0 .. 300Nm in 60 Klassen für den 2. Kanal eingeteilt.

Siehe auch:

ClsTimeAtLevel, ClsOff2ChannelHistogram, ClsOffRevolutionsHistogram, ClsOffRevolutionsMatrix

ClsOffTM

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Realisierung des imc TrueMax Verfahrens.

Deklaration:

```
ClsOffTM ( Input ) -> Ergebnis
```

Parameter:

Input	Eingangsdaten
Ergebnis	Gefiltertes Signal

Beschreibung:

Wenn ein Messsignal abgetastet wird, dann treffen die Abtastwerte nicht immer die wirklichen Extremwerte des physikalschen Signalverlaufs.

Insbesondere bei höheren Signalfrequenzen macht sich das besonders unangenehm bemerkbar:

Maximalwerte werden oft (viel zu) niedrig erfasst, umgekehrt bei Minimalwerten.

Bei der anschließenden Rainflow-Zählung werden dann entsprechende Fehler gemacht.

Für bandbegrenzte Signale (Signale, die mit einem richtig eingestellten Antialiasing-Filter abgetastet wurden) lässt sich entsprechend dem Abtasttheorem der komplette Verlauf rekonstruieren und damit auch der genaue Wert der Extremstellen bestimmen.

Das Verfahren arbeitet mit guter Näherung und verbessert den Signalverlauf, indem die Extremwerte (leicht) verändert werden. Der resultierende Datensatz ist nur noch für die Klassierung geeignet.

Der Zeitverlauf wirkt teilweise unnatürlich.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
ClsOffRainflowInit2 ( ClsHandle, _Min, _Max, _Hysteresis, _Axis, _Type, 1, _CalcOptions)
ClsOffRainflowInit3 ( ClsHandle, 1, 6, 1, 0, 0, 0 )
data_chan1_TM = ClsOffTM ( data_chan1 )
ClsOffRainflowFeedSamples ( ClsHandle, data_chan1_TM )
RainflowMatrix = ClsOffRainflowGetMatrix ( ClsHandle )
```

Vor dem Einspeisen der Messreihe in den Rainflow-Zählalgorithmus werden die Extremwerte korrigiert.

Damit wird die Genauigkeit für die nachfolgende Zählung erhöht.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowFeedSamples

ClsOffWoehlerSN

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Lineare Schadensakkumulation aus der Rainflow-Matrix. Der Algorithmus nach Palmgren-Miner wird benutzt.

Deklaration:

```
ClsOffWoehlerSN ( ClsHandle, Wöhler-Linie, Klassenzuordnung, Interpolation ) -> Ergebnis
```

Parameter:

ClsHandle	Der von ClsOffRainflowInit1 () zurückgegebene Datensatz
Wöhler-Linie	Datensatz, der im XY-Format die Wöhler-Linie beschreibt (Nennspannung über Bruchlastspielzahl). Die Y-Koordinate enthält die Nennwerte der (mechanischen) Spannung. Diese Werte sind fallend angeordnet. Die X-Koordinate enthält die Werte für die zugehörige Bruchlastspielzahl, eine Anzahl von Werten, die i. Allg. ansteigend ist, typisch etwa: 10, 1000, 1000000, ...Für X und Y sind nur Werte > 0 erlaubt.
Klassenzuordnung	Wie wird aus dem Wertebereich einer Klasse der Rainflow-Matrix die zugehörige Y-Koordinate in der Wöhler-Linie gefunden, um damit wiederum über die Wöhler-Linie die zugehörige Bruchlastspielzahl abzulesen?
	0 : Automatisch: Annahme, dass alle Lastspiele in einer Klasse der Rainflow-Matrix gleichmäßig über die volle Ausdehnung der Klasse verteilt sind.
	1 : Der Maximalwert (oberer Rand einer Klasse) wird für alle Lastspiele der Klasse angenommen.
	2 : Der mittlere Wert (Mitte zwischen oberem und unterem Rand einer Klasse) wird für alle Lastspiele der Klasse angenommen.
Interpolation	Mit welcher Interpolation werden die einzelnen Punkte der Wöhler-Linie verbunden?
	0 : Automatisch. Die Wöhler-Linie ist auf doppelt logarithmischem Maßstab aufgetragen. Auf diesem Maßstab werden die angegebenen Punkte der Linie mit Geraden verbunden. Für Metalle.
	1 : Die Wöhler-Linie ist auf einem Maßstab aufgetragen, bei dem die Bruchlastspielzahl logarithmisch, die Spannung linear aufgetragen ist. Auf diesem Maßstab werden die angegebenen Punkte der Linie mit Geraden verbunden. Für Beton.
Ergebnis	Ergebnis ist die akkumulierte Schädigung. Der typische Wertebereich liegt zwischen 0 und 1. Dabei bedeutet eine 0 keine Schädigung, eine 1 eine Schädigung, die zum Bruch führt. Werte größer 1 sind auch möglich.

Beschreibung:

Die tatsächlich gemessene Anzahl von Lastspielen wird der Rainflow-Matrix entnommen. Das Residuum wird dabei nicht beachtet. Wenn es berücksichtigt werden soll, muss es vorher in die Matrix gezählt werden.

Die Rainflow-Matrix muss vorher mit Funktionen ClsOffRainflowInit1() etc. erstellt worden sein. Dabei können z.B. Funktionen wie ClsOffRainflowFeedSamples () benutzt werden, wenn ein Lastkollektiv vorliegt, oder Funktionen wie ClsOffRainflowSetMatrix(), wenn bereits eine Rainflow-Matrix selbst vorliegt.

Die Y-Koordinate der Wöhler-Linie wird immer als Spanne angegeben. Die Spanne ist die Differenz zwischen Maximalwert und Minimalwert einer Schwingung, also die zweifache Amplitude. Dabei ist es egal, ob die Rainflow-Zählung mit Amplituden, Spannen oder Start-Zielklasse gebildet wurde.

Die Y-Koordinate der Wöhler-Linie muss in derselben Einheit vorliegen wie das Lastkollektiv. Wurde die Rainflow-Zählung mit einem Lastkollektiv in N/mm^2 durchgeführt, muss auch die Wöhler-Linie in N/mm^2 vorliegen.

Die Y-Koordinate der Wöhler-Linie kann typisch (mechanische) Spannung, Kraft oder Drehmoment sein. Zu beachten ist, dass Wöhler-Linien für einen Werkstoff oft normiert vorliegen. Dann muss die Wöhler-Linie erst angepasst werden, damit die Einheiten zum Lastkollektiv passen.

Die Wöhler-Linie wird oft durch 2 Punkte vorgegeben. Die zugrunde liegende Gleichung ist S = a * N ^ b, mit S Spannung, N Bruchlastspielzahl, a und b Konstanten. Auf doppelt logarithmischem Maßstab ist das eine Gerade. Da die Funktion passend interpoliert, ist dann die Angabe von diesen beiden Punkten ausreichend.

Die Wöhler-Linie sollte über einen ausreichend weiten Spannungsbereich definiert sein, also i. Allg. mindestens so weit wie der Klassierbereich.

Wenn die Spannung den größten in der Wöhler-Linie angegeben Wert (das ist der erste Y-Wert) überschreitet, wird die dem größten Wert zugeordnete Bruchlastspielzahl angenommen.

Wenn die Spannung den kleinsten in der Wöhler-Linie angegebenen Wert unterschreitet, wird eine unendlich große Bruchlastspielzahl angenommen.

Bitte beachten Sie, dass die Wöhler-Linie nur näherungsweise ein reales Werkstück beschreibt. Die Palmgren-Miner Regel ist auch nur ein Näherungsverfahren, das z.B. die Mittelwerte der Lastspiele und ihre Reihenfolge ignoriert. Dementsprechend ist der hier berechnete akkumulierte Schaden auch nur eine Näherung für den Schaden, den das Werkstück wirklich erlitten hat.

Ein Rückgabewert von z.B. 0.1 bedeutet, dass das Werkstück 10% seiner erwarteten Lebensdauer hinter sich gebracht hat.

Sind für eine bestimmte Spannung z.B. 10 Zyklen in der Rainflow-Matrix gezählt und wird für diese Spannung in der Wöhler-Linie eine Bruchlastspielzahl von 10000 abgelesen, so bedeutet diese Belastung 0.1% (=10/10000) der Lebensdauer.

Beispiele:

```
ClsHandle = ClsOffRainflowInit1 ( _NumberClasses, _TypeOfUnit, _UnitRow, _UnitColumn, _UnitCount, _UnitRes, 0)
.... diverse andere wie ClsOffRainflowFeedSamples() oder ClsOffRainflowSetMatrix
Y = binde ( 2000, 20 ) ; Spannung
X = binde (1e3, 1e8 ) ; Bruchlastspielzahl
SN = xyvon ( X, Y) ; Wöhler-Linie
Damage = ClsOffWoehlerSN ( ClsHandle, SN, 0, 0 )
```

Zuerst wird die Rainflow-Analyse durchgeführt und die Rainflow-Matrix ermittelt.

Dann wird die Wöhler-Linie aus 2 Punkten für ein Werkstück aus Stahl erstellt und die Schädigung ermittelt.

Siehe auch:

ClsOffRainflowInit1, ClsOffRainflowSetMatrix, ClsOffRainflowFeedSamples

ClsQuantile

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Quantil einer Häufigkeitsverteilung. Bis zu welcher x-Koordinate liegt ein gewisser Prozentsatz aller Werte.

Deklaration:

```
ClsQuantile ( Häufigkeitsverteilung, Anteil, Rundung ) -> Ergebnis
```

Parameter:

Häufigkeitsverteilung	Häufigkeitsverteilung, zu der das Quantil zu ermitteln ist
Anteil	Anteil, angegeben in Prozent, 0 bis 100
Rundung	Rundung
	0 : Liegt das Quantil nicht genau auf einer Klassengrenze, wird ein Zwischenwert ausgerechnet. Dabei wird angenommen, dass die Werte innerhalb einer Klasse gleichmäßig verteilt sind. Das entspricht einer Darstellung des Signals in Treppenstufen
	1 : Liegt das Quantil nicht genau auf einer Klassengrenze, wird die nächst untere Klassengrenze zurückgegeben. Bis dorthin ist dann ggf. ein etwas geringerer Anteil enthalten.
	2 : Liegt das Quantil nicht genau auf einer Klassengrenze, wird die nächst höhere Klassengrenze zurückgegeben. Bis dorthin ist dann ggf. ein etwas höherer Anteil enthalten.
Ergebnis	Quantil

Beschreibung:

Die Summe aller y-Werte wird als Gesamtheit angesehen und zu 100% erklärt.

Die Funktion summiert alle y-Werte vom Anfang des Datensatzes (also von links) an, bis die Summe den gewünschten Prozentsatz von der Gesamtheit erreicht.

Die x-Koordinate an dieser Stelle wird zurückgegeben.

Die Summation erfolgt vorzeichenrichtig.

Das Quantil teilt eine Verteilung in 2 Bereiche.

So ist z.B. ein 75%-Quantil die x-Koordinate, links von der 75% der Fläche liegt, rechts 25%.

Die Eingangsdaten werden wie Treppen/Säulen behandelt.

Hat z.B. die Verteilung Werte an den x-Positionen 3, 4, 5, so gibt es drei Treppenstufen mit Häufigkeiten, jeweils der Breite 1. Die letzte Treppe erstreckt sich von 5 bis 6. Das 100%-Quantil ergibt sich also zu 6, wenn die Klassen nicht gerade Null-Werte enthalten.

Verfügbar ab imc FAMOS 7.1

Insbesondere ist der Median das 50%-Quantil. Bei Terzilen wird der Bereich in 3 gleiche Bereiche geteilt. Es gibt 4 Quartile, die je 1/4 des Bereichs breit sind. Es gibt 100 Perzentile, die je 1% breit sind.

Beispiele:

```
Quantile = ClsQuantile ( Histogram, 95, 0 )
```

```
Percentil90 = ClsQuantile ( Histogram, 90, 0 ) - ClsQuantile ( Histogram, 89, 0 )
```

```
MedianValue = ClsQuantile ( Histogram, 50, 0 )
```

Siehe auch:

ClsTimeAtLevel

ClsTimeAtLevel

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Histogramm, Verweildauer, Time at level; Bestimmung der Häufigkeit von Amplituden

Deklaration:

```
ClsTimeAtLevel ( Datensatz, Minimalwert, Maximalwert, Klassenanzahl, Interpolation, Randklassen, x-Einheit, Berechnung ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Interpolation	
	0 : Treppenstufen; Standard-Algorithmus für ein Histogramm. Jeder Abtastwert gilt für die gesamte Abtastzeit. Die Abtastwerte des Datensatzes werden pro Klasse gezählt.
	1 : Linear; die einzelnen Abtastwerte des Datensatzes werden durch Geraden verbunden gedacht. Der komplette Signalverlauf wird gezählt, nicht nur die Abtastwerte selbst.
Randklassen	
	0 : Randklassen geschlossen. Liegt ein Wert außerhalb, wird er nicht gezählt.
	1 : Randklassen offen. Liegt ein Wert außerhalb, wird er in die Klasse am Rand gezählt.
x-Einheit	
	0 : in Klassen beschriften
	1 : physikal. Einheit nutzen
Berechnung	
	0 : Anzahl Messwerte
	1 : Zeit in der physikalischen Einheit der x-Achse des zu klassierenden Datensatzes
	2 : Prozent; als Anteil der Gesamtdauer bzw. Gesamtanzahl, Wertebereich 0 .. 100%
	3 : Amplitude probability density, Wahrscheinlichkeitsdichte der Amplitude. Das Integral über alles ergibt 1. Das Integral über einen Bereich ergibt die Wahrscheinlichkeit, dass die Amplitude in diesem Bereich liegt.
	4 : Akkumulierte relative Häufigkeit. Wieviel Prozent der Werte liegen in dieser oder kleineren Klassen. Wertebereich 0 .. 100%
Ergebnis	Histogramm

Beschreibung:

Klassierung von Messdaten mit dem Verweildauerverfahren nach DIN 45667, falls Interpolation mit Treppenstufen

Die Funktion ersetzt KlsVWeil() bzw. ClsTAtLv()

Beispiele:

Histogramm, auch DIN 45667

```
TimeAtLevel = ClsTimeAtLevel ( channel, -10.0, 10.0, 100, 0, 1, 0, 0 )
```

Verweildauer

```
TimeAtLevel = ClsTimeAtLevel ( channel, -10.0, 10.0, 100, 0, 1, 1, 1 )
```

Präzise Verweildauer

```
TimeAtLevel = ClsTimeAtLevel ( channel, -10.0, 10.0, 100, 1, 1, 1, 1 )
```

Amplitude probability density

```
TimeAtLevel = ClsTimeAtLevel ( channel, -10.0, 10.0, 100, 0, 1, 1, 3 )
```

Siehe auch:

KlsVWeil, ClsQuantile

Cmp1

Erste Komponente eines komplexen Datensatzes.
Alternativer Name: **Kmp1**

Deklaration:

```
Cmp1 ( Daten ) -> Komponente1
```

Parameter:

Daten	Komplexer Datensatz, dessen erste Komponente geliefert werden soll. [KK]
Komponente1	Erste Komponente des komplexen Datensatzes.

Beschreibung:

Diese Funktion liefert die erste Komponente eines komplexen Datensatzes. Die erste Komponente ist je nach Typ des komplexen Datensatzes der Realteil oder der Betrag oder der Betrag in dB.

- Bei Zwischenergebnissen innerhalb von Formeln lassen sich die Erweiterungen ".R", ".M" usw. zur Bildung einer Komponente eines komplexen Datensatzes nicht anbringen. Dann ist eine Benutzung der Funktion Cmp1() sinnvoll.
- Ist der komplexe Typ nicht bekannt, kann die Funktion Cmp1() nicht sinnvoll genutzt werden. Schaffen Sie eindeutige Verhältnisse, indem Sie die Funktion Pol() oder Rect() aufrufen.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Der Betrag eines Spektrums ist zu bestimmen, es wird ausgenutzt, dass die Funktion Spec() stets den komplexen Typ BP (Betrag/Phase) zurückliefert:

```
magnitude = Cmp1(Spec(NDdata))
```

Der Realteil eines komplexen Datensatzes wird einer Variablen zugewiesen. Beide Anweisungen sind äquivalent:

```
NDreal = Cmp1(RIdata)
NDreal = RIdaten.R
```

Siehe auch:

Cmp2, Compl, IsCplx, Pol, Rect

Cmp2

Zweite Komponente eines komplexen Datensatzes.
Alternativer Name: **Kmp2**

Deklaration:

```
Cmp2 ( Daten ) -> Komponente2
```

Parameter:

Daten	Komplexer Datensatz, dessen zweite Komponente geliefert werden soll. [KK]
Komponente2	Zweite Komponente des komplexen Datensatzes.

Beschreibung:

Diese Funktion liefert die zweite Komponente eines komplexen Datensatzes. Die zweite Komponente ist je nach Typ des komplexen Datensatzes der Imaginärteil oder die Phase.

- Bei Zwischenergebnissen innerhalb von Formeln lassen sich die Erweiterungen ".I", ".P" usw. zur Bildung einer Komponente eines komplexen Datensatzes nicht anbringen. Dann ist eine Benutzung der Funktion Cmp2() sinnvoll.
- Ist der komplexe Typ nicht bekannt, kann die Funktion Cmp2() nicht sinnvoll genutzt werden. Schaffen Sie eindeutige Verhältnisse, indem Sie die Funktion Pol() oder Rect() aufrufen.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Die Phase eines Spektrums ist zu bestimmen, es wird ausgenutzt, dass die Funktion Spec() stets den komplexen Typ BP (Betrag/Phase) zurückliefert:

```
Phase = Cmp2(Spec(NDdata))
```

Der Imaginärteil eines komplexen Datensatzes wird einer Variablen zugewiesen. Beide Anweisungen sind äquivalent:

```
NDimag = Cmp2(RIdata)
NDimag = RIdaten.I
```

Siehe auch:

Cmp1, Compl, IsCplx, Pol, Rect

CmpX

X-Komponente eines XY-Datensatzes.
Alternativer Name: **KmpX**

Deklaration:

```
CmpX ( Daten ) -> KomponenteX
```

Parameter:

Daten	XY-Datensatz, dessen X-Komponente geliefert werden soll. [XY]
KomponenteX	X-Komponente des Datensatzes.

Beschreibung:

Die X-Komponente eines XY-Datensatzes wird geliefert.

- Sie können zum Zugriff auf die Komponenten eines XY-Datensatzes auch die Suffixe .X und .Y benutzen.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Die X-Komponente eines XY-Datensatzes wird mit 2 multipliziert. Beide Anweisungen sind äquivalent:

```
data.X = CmpX(data) * 3
data.X = data.X * 3
```

Siehe auch:

CmpY, XYof, IsXY

CmpY

Y-Komponente eines XY-Datensatzes.
Alternativer Name: **KmpY**

Deklaration:

```
CmpY ( Daten ) -> KomponenteY
```

Parameter:

Daten	XY-Datensatz, dessen Y-Komponente geliefert werden soll. [XY]
KomponenteY	Y-Komponente des Datensatzes.

Beschreibung:

Die Y-Komponente eines XY-Datensatzes wird geliefert.

- Sie können zum Zugriff auf die Komponenten eines XY-Datensatzes auch die Suffixe .X und .Y benutzen.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Die Y-Komponente eines XY-Datensatzes wird mit 2 multipliziert. Beide Anweisungen sind äquivalent:

```
data.Y = CmpY(data) * 2
data.Y = data.Y * 2
```

Siehe auch:

CmpX, XYof, IsXY

CodeRange

Verfügbar ab: Professional Edition

Einer Liste von Bereichen von Y-Werten des Eingangssignals wird eine Liste von Zahlenwerten (Codes) zugeordnet und zurückgegeben.

Deklaration:

```
CodeRange ( Eingangsdaten, Untergrenzen, Obergrenzen, Codes [, Vergleich] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Eingangsdaten
Untergrenzen	Liste der Untergrenzen aller Wertebereiche; alternativ Tripelfolge
Obergrenzen	Liste der Obergrenzen aller Wertebereiche; 0 bei Tripelfolge
Codes	Liste der Codes zu allen Wertebereichen; 0 bei Tripelfolge
Vergleich	Formulierung des Vergleichs (optional , Standardwert: "L<=y<=H")
	"L<=y<=H" : Untergrenze <= Eingangsdaten <= Obergrenze
	"L<y<=H" : Untergrenze < Eingangsdaten <= Obergrenze
	"L<=y<H" : Untergrenze <= Eingangsdaten < Obergrenze
	"L<y<H" : Untergrenze < Eingangsdaten < Obergrenze
Ergebnis	Ergebnis

Beschreibung:

Durchführung einer Bereichskodierung

Die Datensätze mit den Untergrenzen, Obergrenzen und Codes sind alle gleich lang und enthalten in selber Reihenfolge für jeden Wertebereich dessen Beschreibung.

Die Liste der Wertebereiche muss nicht sortiert sein. Die Funktion arbeitet die Liste von vorn nach hinten ab. Bei erster Übereinstimmung wird der gefundene Code zurückgegeben.

Ein Eingangswert gilt einem Wertebereich zugeordnet, wenn der Vergleich entsprechend seiner genauen Formulierung (< bzw. <=) erfüllt ist.

Die Funktion kann wie eine switch Anweisung mit case Anweisungen für jeden Wertebereich gedacht werden. Dabei wird die switch Anweisung für jeden Messwert des Eingangssignals einmal durchlaufen.

Liegt der Wert der Eingangsdaten in keinem der Wertebereiche, wird eine null zurückgegeben. Das kann als default Fall der switch Anweisung angesehen werden.

Die Eingangsdaten können Events und Segmente haben. Sind sie vom Typ XY, wird die Bereichskodierung auf die Y-Komponente angewendet.

In einer zu imc Inline FAMOS kompatiblen Weise kann ein einziger steuernder Datensatz gefolgt von Nullen angegeben werden, bei dem Tripel aus Untergrenze, Obergrenze und Code nacheinander enthalten sind, siehe Beispiel.

Beispiele:

Gangerkennung: rpm_in, rpm_out sind die Eingangs- und Ausgangsdrehzahlen am Getriebe. Ungültige Bereiche werden etwas verbreitert.

```
Lo = [ 0.4, 0.7, 0.9, 1.1 ]
Hi = [ 0.5, 0.8, 0.93, 1.17 ]
Co = [ 1, 2, 3, 4 ]
Ratio = CodeRange ( rpm_out / rpm_in, Lo, Hi, Co )
Change1 = ( Ratio = 0 ) OR (rpm_in < 20 )
Change2 = Monoflop ( Change1, 0.5, "1 retrig", "0-1" )
Change3 = Monoflop ( Change2, 0.5, "1 retrig", "0-1", "reverse" ) ; =1 when gears are changed
Gear = Ratio * (1-Change3)
```

Ein Datensatz steuert, Anwendung wie 1. Beispiel

Kompatibel zu imc Inline FAMOS und imc Online FAMOS:

Der Parameter Control enthält die Wertefolge low[1], high[1], code[1], low[2], high[2], code[2], ...

```
Control = [ 0.4, 0.5, 1, 0.7, 0.8, 2, 0.9, 0.93, 3, 1.1, 1.17, 4 ] ; 4 triples of low/high/code
Ratio = CodeRange ( rpm_out / rpm_in, Control, 0, 0 )
```

Siehe auch:

switch, RangeSet

Coherence

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Die Kohärenz wird ermittelt durch lineare Mittelung von Leistungsspektren.

Deklaration:

```
Coherence ( Eingangskanal, Ausgangskanal, Fensterbreite, Fenstertyp, Überlappung [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangskanal	Der Bezugskanal, Eingangskanal. Die Anregung eines Systems. In Sekunden skaliert.
Ausgangskanal	Der (verzögerte) Ausgangskanal. Die Antwort eines Systems. Ein- und Ausgangskanal haben dieselbe Zeitbasis und sind in Sekunden skaliert.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ergebnis

Beschreibung:

Wenn x Eingang und y Ausgang und G ein Leistungsspektrum, dann wird | Gxy | ^ 2 / (Gxx * Gyy) bestimmt. Wertebereich: 0.0 ... 1.0

Die Berechnung der Mittelung erfolgt über Real- und Imaginärteil getrennt.

Die Berechnung wird erst sinnvoll, wenn eine merkliche Mittelung stattfindet.

Beispiele:

```
Coh = Coherence ( Kraft, Bewegung, 1000, 0, 50, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Leistungsspektren, die sich um je 50% überlappen. Eine Kraft wirkt auf ein mechanisches Teil. Die Bewegung des Bauteils am anderen Ende wird gemessen. Aus den gemittelten Leistungsspektren wird das Resultat bestimmt.

Siehe auch:

FrequencyResponse, CrossPowerDS

Color?

Abfragen der Farbe für die Kurvendarstellung eines Datensatzes
Alternativer Name: **Farbe?**

Deklaration:

```
Color? ( Daten ) -> EwFarbWert
```

Parameter:

Daten	Datensatz, dessen Farbattribut abgefragt werden soll.
EwFarbWert	Farbwert, -1 bedeutet automatische Farbgebung.

Beschreibung:

Normalerweise werden Datensätze im Kurvenfenster in der Farbe angezeigt, die am Kurvenfenster für die Darstellung eingestellt wurde. Sie können aber auch eine feste Farbe zuweisen, mit der der Datensatz dann in jedem Kurvenfenster unabhängig von aktuellen Einstellungen angezeigt wird.

Die Funktion liefert den Farbwert zurück, mit dem der Datensatz angezeigt wird oder -1, falls keine feste Farbe zugewiesen ist.

Der Farbwert ist ein so genannter RGB-Wert, in dem die Anteile der 3 Grundfarben Rot, Grün und Blau vermerkt sind.

Zum Bilden eines Farbwertes aus den Farbanteilen können Sie die Funktion RGB() benutzen.

Beispiele:

```
green = RGB(0, 255, 0)
clr = Color?(data)
IF clr = -1
   SetColor(data, green)
END
```

Falls dem Datensatz bisher keine feste Farbe zugewiesen worden ist, wird er in Zukunft immer Grün dargestellt.

Siehe auch:

SetColor, RGB

Comm?

Abfrage des Kommentars zu einem Datensatz, Text oder Datengruppe
Alternativer Name: **Komm?**

Deklaration:

```
Comm? ( Datenobjekt ) -> TxKommentar
```

Parameter:

Datenobjekt	Datensatz, Text oder Datengruppe, dessen Kommentar ermittelt werden soll.
TxKommentar	Kommentar

Beschreibung:

Der Kommentar zu einem Datensatz, Text oder einer Gruppe wird abgefragt. Jedem dieser Datentypen kann ein Kommentar zugeordnet sein, der per Dialog oder durch die Funktion SetKomm() gesetzt werden kann.

Beispiele:

Der Kommentar eines Datensatzes wird abgefragt. Falls er nicht gesetzt ist (Länge 0), wird der Nutzer aufgefordert, einen Kommentar einzugeben. Dieser wird dem Datensatz dann zugewiesen.

```
txComment = Comm?(data)
IF TLeng(txComment) = 0
   txComment = BoxText?("Kommentar eingeben:", "",0)
   SetComm(data, txComment)
END
```

Siehe auch:

SetComm, Name?

COMMENT

Kommentar einleiten
Alternativer Name: **KOMMENTAR**

Der Befehl ist veraltet, wegen der besseren Lesbarkeit sollte ein Semikolon benutzt werden, um Kommentarzeilen einzuleiten.

Deklaration:

```
COMMENT Kommentar
```

Parameter:

Kommentar	Beliebiger Text für den Kommentar

Beschreibung:

Dieser Befehl leitet einen Kommentar ein. Alle Zeichen, die hinter diesem Befehl stehen, werden als Kommentar verstanden. Dieser Befehl ist geeignet, um Sequenzen zu kommentieren.

Mit Semikolon eingeleitete Kommentare können auch nach einem Befehl / einer Formel in der gleichen Zeile stehen.

Beispiele:

```
COMMENT -- Das ist eine kurze Sequenz,
COMMENT -- die eine Datei lädt und anzeigt.
; -- Es ist ein Parameter zu übergeben.
LOAD PA1 ; Laden des Parameters
SHOW PA1 ; Anzeigen des Parameters
```

Ein Datensatz wird geladen und angezeigt. Die ersten drei Zeilen sind Kommentar.

Siehe auch:

SEQUENCE

Compl

Fasst zwei reelle Datensätze zu einem komplexen zusammen.
Alternativer Name: **Kompl**

Deklaration:

```
Compl ( Komponente1, Komponente2 ) -> KomplexeDaten
```

Parameter:

Komponente1	Realteil oder Betrag des zu bildenden komplexen Datensatzes. [ND]
Komponente2	Imaginärteil oder Phase des zu bildenden komplexen Datensatzes. [ND]
KomplexeDaten	Resultierender komplexer Datensatz. [BP],[DP] oder [RI]

Beschreibung:

Zwei reelle Datensätze (Typ ND) werden zu einem komplexen Datensatz zusammengefasst. Einer der beiden übergebenen Parameter muss ein normaler Datensatz, der andere aber darf auch ein Einzelwert sein. Der Einzelwert wird dann automatisch zu einem normalen Datensatz von passender Dimension expandiert.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), müssen dann aber exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen). Die Kombination eines strukturierten Parameters mit einem Einzelwert ist ebenfalls erlaubt.

Der erzeugte komplexe Datentyp richtet sich nach den Einheiten der beiden übergebenen Datensätze. Der komplexe Datentyp wird nach folgenden Kriterien bestimmt:

Sind die Einheiten gleich oder haben beide Datensätze keine Einheit, wird der Typ RI erzeugt.

Wenn der Typ nicht bereits RI ist, wird der Typ DP erzeugt, wenn die Einheit des ersten übergebenen Datensatzes dB ist.

Wenn der Typ nicht bereits RI oder DP ist, wird der Typ BP erzeugt.

- Geben Sie einer Phase stets eine eindeutige Einheit, also z.B. "Rad" oder "Grad".
- Beide übergebenen reellen Datensätze sollten exakt dieselbe x-Skalierung und Länge haben. Andernfalls wird eine Warnung erzeugt und gegebenenfalls eine Anpassung vorgenommen.
- Wenn Sie nur eine Komponente eines komplexen Datensatzes ändern möchten, benutzen Sie die Erweiterung ".M", ".P" usw.

Beispiele:

Die 3 nachfolgenden Formeln erzeugen jeweils einen imaginären Datensatz, d. h. einen komplexen Datensatz, dessen Realteil 0 ist. Beachten Sie, dass im Falle eines übergebenen Einzelwertes dessen Einheit mit der des übergebenen Datensatzes übereinstimmen muss, um einen komplexen Datensatz vom Typ RI zu erzeugen:

```
RIdata = Compl(RIdata.R * 0, RIdata.I)
RIdata = Compl(0 'V', RIdata.I)
RIdata = Compl(0 'V', NDdata)
```

Es wird ein komplexer Datensatz erzeugt, dessen Betrag von 0 Hz nach 511 Hz wächst und dessen Phase konstant 90° ist. Dieser komplexe Datensatz ist also rein imaginär. Er kann benutzt werden, um die oft benötigte Kombination j (Kreisfrequenz) darzustellen:

```
MPjomega = Compl(Ramp(0, 1, 512) * 1'Hz', 90'°'* (1 - 0 * Ramp(0, 1, 512)))
```

Siehe auch:

Cmp1, Cmp2, IsCplx, Pol, Rect

ComplexSpectrum

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Komplexes Spektrum (Harmonische als Effektivwerte bestimmt) mit gleitendem Fenster und linearer Mittelung. Berechnung mittels FFT.

Deklaration:

```
ComplexSpectrum ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	0 : Keine Mittelung
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung von Real- und Imaginärteil). Der Mittelwert wird über so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	2 : Peak Hold Max ab Beginn, Maximalwerte, berechnet von allen bislang berechneten Spektren
	3 : Peak Hold Max pro Intervall, Maximalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	4 : Peak Hold Min ab Beginn, Minimalwerte, berechnet von allen bislang berechneten Spektren
	5 : Peak Hold Min pro Intervall, Minimalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase. Die Phase wird in Grad ermittelt. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Beschreibung:

Die Berechnung der linearen Mittelung erfolgt über Real- und Imaginärteil getrennt. Für die Berechnung eines gemittelten Betragsspektrums wird die Funktion AmpSpectrumRMS() empfohlen.

Bei Peak Hold Berechnungen wird das Peak Hold auf den Betrag angewendet. Die Phase wird gemittelt, ist aber ohnehin dann bedeutungslos und sollte ignoriert werden.

Beispiele:

```
Spektren = ComplexSpectrum ( Kanal, 1000, 0, 50, 1, 0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen.

```
Spektren = ComplexSpectrum ( Kanal, 2048, 1, 0, 10, 1, 0 )
```

Das ist die Berechnung einer Folge von 2048 Punkte-Spektren mit Hamming-Fenster. Über je 10 Spektren wird der Mittelwert bestimmt und im Ergebnis vermerkt.

Siehe auch:

ComplexSpectrum_exp, ComplexSpectrum_1, AmpSpectrumRMS

ComplexSpectrum_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein gemitteltes komplexes Spektrum (Harmonische als Effektivwerte bestimmt) wird bestimmt. Die Mittelung erfolgt über viele Spektren, die aus einem gleitenden Fenster ermittelt wurden. Berechnung mittels FFT.

Deklaration:

```
ComplexSpectrum_1 ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung von Real- und Imaginärteil). Der Mittelwert wird über alle berechneten Spektren gebildet.
	2 : Peak Hold Max, Maximalwerte, berechnet von allen berechneten Spektren
	4 : Peak Hold Min, Minimalwerte, berechnet von allen berechneten Spektren
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ein gemitteltes Spektrum als Ergebnis. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase. Die Phase wird in Grad ermittelt.

Beschreibung:

Die Berechnung der linearen Mittelung erfolgt über Real- und Imaginärteil getrennt. Für die Berechnung eines gemittelten Betragsspektrums wird die Funktion AmpSpectrumRMS_1() empfohlen.

Bei Peak Hold Berechnungen wird das Peak Hold auf den Betrag angewendet. Die Phase wird gemittelt, ist aber ohnehin dann bedeutungslos und sollte ignoriert werden.

Beispiele:

```
Spektrum = ComplexSpectrum_1 ( Kanal, 1000, 0, 50, 1, 0 )
```

Das ist die Berechnung eines gemittelten Spektrums. Die Mittelung geschieht über eine Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Eingangskanal hat etwa 20000 Messwerte.

Siehe auch:

ComplexSpectrum, ComplexSpectrum_exp, AmpSpectrumRMS_1

ComplexSpectrum_exp

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Komplexes Spektrum (Harmonische als Effektivwerte bestimmt) mit gleitendem Fenster und exponentieller Mittelung. Berechnung mittels FFT.

Deklaration:

```
ComplexSpectrum_exp ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Zeitkonstante [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Zeitkonstante	Die Zeitkonstante für die exponentielle Mittelung. In Sekunden angegeben.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase. Die Phase wird in Grad ermittelt. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Beschreibung:

Die Berechnung der Mittelung erfolgt über Real- und Imaginärteil getrennt. Für die Berechnung eines gemittelten Betragsspektrums wird die Funktion AmpSpectrumRMS_exp() empfohlen.

Beispiele:

```
Spektren = ComplexSpectrum_exp ( Kanal, 1000, 0, 50, 2, 40.0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Kanal hat eine Abtastzeit von 10ms. Also wird alle 5s ein Spektrum gebildet. Diese werden mit einer Zeitkonstante von 40.0s geglättet. Jedes 2. Spektrum wird ausgegeben.

Siehe auch:

ComplexSpectrum, ComplexSpectrum_1, AmpSpectrumRMS_exp

CONTINUE

Der Befehl beendet den aktuellen Schleifendurchlauf. Die Schleife wird, wenn die Schleifen-Bedingung noch erfüllt ist, mit dem nächsten Durchlauf fortgesetzt.

Deklaration:

```
CONTINUE
```

Beschreibung:

Der Befehl kann innerhalb einer WHILE-, FOR- oder FOREACH-Schleife verwendet werden.

Beispiele:

Es werden alle Dateien mit der Erweiterung "*.dat" in einem vorgegebenen Verzeichnis ermittelt und in einer Schleife aufgezählt. Wenn die Dateizeit nach einem festen Stichtag liegt, wird die Datei geladen und verarbeitet.

```
list  = FsFileListNew("c:\imc\dat", "*.dat", 0, 0, 0)
count = FsFileListGetCount(list)
deadline = TimeJoin(1, 1, 2012, 0, 0, 0)
FOR i = 1 TO count
   time = FsFileListGetTime(list, i)
   IF time < deadline
      CONTINUE
   END
   ; load and process file
   TxName = FsFileListGetName(list, i)
   fh = FileOpenDSF(TxName, 0)
   ; ...
END
FsFileListClose(list)
```

Siehe auch:

WHILE, FOR, FOREACH, BREAK

CONTROL

Verfügbar ab: Professional Edition

Befehl über DDE an eine andere Applikation senden
Alternativer Name: **ANWEISEN**

Der Befehl ist veraltet, statt dessen sollten die leistungsfähigere Funktion DDESend() verwendet werden.

Deklaration:

```
CONTROL Applikation Thema Kommando Variablenname
```

Parameter:

Applikation	Name der anzusprechenden DDE-Applikation (engl.:Application).
Thema	Bezeichnung des DDE-Themas (engl.: Topic).
Kommando	DDE-Kommando entsprechend den Format-Spezifikationen der Applikation (engl.: Command).
Variablenname	Name der FAMOS-Variable zum Empfang des Rückgabewertes

Beschreibung:

imc FAMOS arbeitet als DDE-Client, die angesprochene Applikation als DDE-Server. Über die Einträge [Applikation] und [Thema] wird eine andere DDE-fähige Applikation von imc FAMOS ausgewählt und eine Konversation angefordert. Wird der Anforderung entsprochen, überträgt imc FAMOS das gewünschte Kommando. imc FAMOS wartet dann, bis der Server die Übernahme der Daten bestätigt oder negativ antwortet. Danach beendet imc FAMOS die Konversation.

Die optional unter [Variablenname] angegebene Variable enthält den Rückgabewert der angesprochenen DDE-Applikation.

- Die einzelnen Parameter des CONTROL-Befehls dürfen keine Leerzeichen enthalten.
- FAMOS überprüft die Syntax des "Kommando"-Eintrages nicht. Bitte überprüfen Sie das Format des Eintrages entsprechend der DDE-Spezifikation der anzusprechenden Applikation.
- FAMOS erzeugt eine Fehlermeldung, wenn die angesprochene DDE-Applikation (Server) nicht antwortet oder das Kommando nicht akzeptiert.
- Ist die angesprochene Applikation (Server) beschäftigt (engl.: busy), wartet imc FAMOS, bis sie frei ist.
- FAMOS-Befehle können wahlweise groß oder klein geschrieben werden (engl.: Case insensitive), wogegen einige DDE-Applikationen Unterscheidungen treffen (engl.: Case sensitive). Achten Sie also auf richtige Schreibweise.

Beispiele:

```
CONTROL Trans Zeitbasis [dt=1.5e-3]
```

"Trans" ist der Name der empfangenden DDE-Applikation, "Zeitbasis" der des Themas. Das DDE-Kommando ist in diesem Fall "dt=1.5e-3". Es wird kein Rückgabewert gewünscht.

```
CONTROL Trans Zeitbasis [trigger][arm] RückWert
```

"Trans" ist der Name der empfangenden DDE-Applikation, "Zeitbasis" das Thema. Angewiesen werden hier zwei Kommandos gleichzeitig, die auch zur selben Zeit geschrieben werden. Den Rückgabewert dieser DDE-Konversation enthält die Variable "RückWert". Damit kann er überprüft werden.

Siehe auch:

DDESend, DDEInq, DDESet, REQUEST

ConvertUnit

Konvertiert eine Einheit und verändert entsprechend die Zahlenwerte bzw. Kennwerte des Datensatzes.

Deklaration:

```
ConvertUnit ( Daten, TxEinheit, EwAuswahl ) -> Ergebnis
```

Parameter:

Daten	Zu konvertierender Datensatz.
TxEinheit	Einheit, in die konvertiert werden soll. Falls Sonderwerte wie "SI+", dann wird ausgehend von der aktuellen Einheit in eine SI-Einheit ohne Vorsatz konvertiert.
EwAuswahl	Auswahl der Einheit
	0 : X-Einheit bei 1-komponentigen Daten. Einheit der X-Komponente bei XY-Daten. Einheit der Phase bzw. des Imaginärteils bei komplexen Daten.
	1 : Y-Einheit bei 1-komponentigen Daten. Einheit der Y-Komponente bei XY-Daten. Einheit des Betrages bzw. des Realteils bei komplexen Daten.
	2 : Z-Einheit
	3 : Einheit des Parameters bei 2-komponentigen Daten.
	-1 : Nur für Normierung auf SI-Einheiten erlaubt (2. Parameter hat einen Sonderwert wie "SI+"). Es werden dann alle vorhandenen Einheiten im Datensatz normiert.
Ergebnis	Konvertierter Datensatz.

Beschreibung:

Die Funktion konvertiert eine Einheit des Datensatzes und verändert entsprechend die Zahlenwerte.

Typische Anwendungen:

- Konvertieren von Einheiten-Vorsätzen, z.B. 'mV' in 'V' => alle Zahlenwerte werden durch 1000 geteilt.
- Konvertieren verwandter Einheiten, z.B. '°C' in 'K' (Kelvin) => auf alle Zahlenwerte wird 273.15 addiert.

Die Funktion ist insbesondere nützlich, um bei mathematischen Operationen die Parameter vor der eigentlichen Berechnung anzupassen. Dies betrifft insbesondere die Grundrechenarten, aber auch z.B. Integration (siehe Beispiel), Differentiation und FFT (Zeitachse in s).

Aktuelle Einheit und Ziel-Einheit müssen sinnvoll ineinander überführbar sein. Beispielsweise ist eine Konvertierung von 'mA' in 'V' nicht sinnvoll und liefert einen Fehler.

Umrechnung von Temperatureinheiten:

Temperaturen werden mit ihren Offsets umgerechnet, also z.B.

```
T [K] = T [°C] + 273.15
T [°F] = T [°C] * 1.8 + 32
```

Temperaturdifferenzen werden ohne Offsets ineinander umgerechnet:

```
DeltaT [K] = DeltaT [°C]
DeltaT [°F] = DeltaT [°C] * 1.8
```

Die Funktion erkennt von allein Temperaturdifferenzen nach durchgeführter Kürzung an folgenden Merkmalen: Die Temperatureinheit taucht im Nenner auf, etwa bei 1/°C oder mV//°C. Die Temperatureinheit taucht in Kombination mit anderen Einheiten auf, etwa bei °C/W. Es wird empfohlen, bei der Berechnung von Temperaturdifferenzen gleich die Einheit K zu vergeben, obwohl °C dabei auch erlaubt und üblich ist.

Sonderwerte des 2. Parameters

Bei allen Sonderwerten "SI..." findet eine Normierung auf die passende SI-Einheit statt. Vorsätze wie milli, kilo etc. werden entfernt und die Zahlenwerte entsprechend umgerechnet. Die SI-Einheiten bestehen aus den Basis-SI-Einheiten A, cd, K, kg, m, mol, s und den abgeleiteten SI-Einheiten Bq, °C, C, F, Gy, H, Hz, J, kat, lm, lx, Ohm, N, Pa, rad, sr, S, Sv, T, V, W, Wb.

Bei "SI+" und "SI+D" bleiben außerdem Einheiten wie Bark, Bft, dB, Grad (Degree), phon, Scoville, sone erhalten.

Bei den Sonderwerten wird eine Sonderbehandlung der Temperatur wie folgt durchgeführt:

"SI0"	Die erzeugte Temperatureinheit ist K. Wird eine Temperatur erkannt, wird z.B. 0°C = 273,15K angewendet.
"SI0D"	Alle erkannten Temperaturen werden als Temperaturdifferenz gedeutet, womit 1K=1°C angewendet und stets K erzeugt wird.
"SI+"	Die erzeugte Temperatureinheit ist °C, aber K bleibt erhalten. Wird eine Temperatur erkannt, wird z.B. 32°F = 0°C angewendet.
"SI+D"	Alle erkannten Temperaturen werden als Temperaturdifferenz gedeutet, womit 1.8°F=1°C angewendet und °C erzeugt wird, aber K erhalten bleibt.

Die Erkennung und Umrechnung von Einheiten wird durch eine Reihe von Voreinstellungen gesteuert, die in FAMOS im Dialog 'Extras'/'Optionen'/'Einheiten' oder mittels der Funktion SetOption() konfiguriert werden können.

Die Funktion kann auf normale (gleichmäßig abgetastete) Datensätze sowie auf XY- und komplexe Daten angewendet werden.

Beispiele:

```
; Signal hat die Y-Einheit 'm/s'
```

```
Signal = ConvertUnit(Signal, "km/h", 1)
; => alle Y-Werte werden mit 3.6 multipliziert

; Signal hat die X-Einheit 's'
Signal = ConvertUnit(Signal, "min", 0)
; => Normaler Datensatz (1-komponentig): Kennwerte x-Offset (Pretrigger) und x-Delta (Abtastzeit) werden durch 60 dividiert
; => XY-Datensatz: alle Werte der X-Komponente werden durch 60 dividiert

Signal = ConvertUnit(Signal, "SI0", 1)
; Y-Einheit 'kV' => Ergebnis: 'V', alle Y-Werte mit 1000 multipliziert
; Y-Einheit 'min' => Ergebnis: 's', alle Y-Werte mit 60 multipliziert
; Y-Einheit '°C' => Ergebnis: 'K', auf alle Y-Werte wird 273.15 addiert
; Y-Einheit 'km/h' => Ergebnis: 'm/s', alle Y-Werte werden durch 3.6 dividiert
; Y-Einheit 'V/A' => Ergebnis: 'Ohm', Y-Werte unverändert
```

Eine elektrische Leistung wird berechnet. Die Spannung ist in mV, der Strom in mA gemessen. Für eine korrekte Berechnung werden vor der Multiplikation beide Parameter in die Basiseinheit konvertiert, das Ergebnis bekommt automatisch die Einheit 'W'.

```
P = ConvertUnit(U, "SI0", 1) * ConvertUnit(I, "SI0", 1)
```

Ein Beschleunigungssignal hat die Y-Einheit 'g' (Erdbeschleunigung), die Zeitachse ist in Sekunden angegeben. Für die Berechnung der Geschwindigkeit mittels Integration muss das Signal vorher in 'm/s^2' konvertiert werden.

```
SetOption("Units.Read.g", "gravity") ; "g" als Beschleunigung interpretieren (statt als Masse [gramm])
v = Int(ConvertUnit(a, "SI0", 1))
```

Umrechnung eines Drehzahlsignals (in Umdrehungen pro Minute) in Winkelgeschwindigkeit und Frequenz:

```
tr = ConvertUnit(60'RPM', "Hz", 1) ; => 1Hz
tr = ConvertUnit(60'RPM', "rad/s", 1) ; => 6.2832 rad/s
tr = ConvertUnit(60'RPM', "Degr/s", 1) ; => 360 Degr/s
```

Siehe auch:

SetUnit, Unit?

CorrCoeff

Verfügbar ab: Professional Edition

Korrelationskoeffizient, auch gleitend

Deklaration:

```
CorrCoeff ( Pattern, Testdata [, Berechnung] [, Parameter] ) -> Ergebnis
```

Parameter:

Pattern	Datensatz x, der erste der beiden zu korrelierenden Datensätze. Bei gleitender Berechnung das (kurze) Muster bzw. Referenzdatensatz, dessen Vorkommen im Testdatensatz gesucht wird.
Testdata	Datensatz y, der zweite der beiden zu korrelierenden Datensätze. Bei gleitender Berechnung der (lange) Datensatz, in dem das Muster (Pattern) gesucht wird.
Berechnung	Berechnung (optional , Standardwert: 0)
	0 : Ein Korrelationskoeffizient wird beide Datenreihen berechnet. Es erfolgt keine gleitende Berechnung.
	1 : Eine Folge von Korrelationskoeffizienten wird gleitend über Pattern und über ein fortschreitendes Fenster von Testdata bestimmt.
	2 : Wie "Korrelationskoeffizient gleitend", aber zusätzlich wird das Ergebnis mit einem Faktor SF_rms gewichtet.
	3 : Wie "Korrelationskoeffizient gleitend", aber zusätzlich wird das Ergebnis mit einem Faktor SF_span gewichtet.
	4 : Wie "Korrelationskoeffizient gleitend", aber zusätzlich wird das Ergebnis mit einem Faktor SF_fred gewichtet.
Parameter	Abhängig von der Art der Berechnung der benötigte Wert. Bei "Korrelationskoeffizient F_rms" der minimale Effektivwert (>=0) eines Fensters von Testdata. Bei "Korrelationskoeffizient F_span" die minimale Spanne (>=0) eines Fensters von Testdata. Bei "Korrelationskoeffizient F_fred" die minimale relative Spanne von 0 bis 1 für 0 bis 100%. Sonst 0. (optional , Standardwert: 0)
Ergebnis	Ergebnis

Beschreibung:

Korrelationskoeffizient

Bei der Berechnung des Korrelationskoeffizienten wird die Funktion nur mit den ersten beiden Argumenten aufgerufen.

Die Berechnung des Korrelationskoeffizient r erfolgt nach folgender Formel, wobei n die Anzahl der Samples, x Pattern und y Testdata bedeutet.

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{\sqrt{(n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2)(n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2)}}$$

Bei nicht gleitender Berechnung bestimmt der kürzere der beiden Eingangsdaten Pattern und Testdata, über wie viele Punkte berechnet wird.

Liegen z.B. 2 normale Datensätze für Pattern und Testdata mit je 1000 Messwerten vor, wird der Korrelationskoeffizient über diese 1000 Werte ermittelt und als 1 Wert zurückgegeben.

Die Berechnung des Korrelationskoeffizienten beinhaltet stets eine Subtraktion des Mittelwertes. Hat z.B. Pattern die Werte [0, 2, 2, 0], ist das gleichbedeutend mit [-1, 1, 1, -1].

Die Berechnung erfolgt Punkt für Punkt. Die Abtastzeit von Pattern wird nicht berücksichtigt.

Der Wertebereich des Korrelationskoeffizienten reicht von -1 bis +1. Bei +1 (-1) besteht ein vollständig positiver (negativer) linearer Zusammenhang. Bei 0 hängen beide Datensätze überhaupt nicht voneinander ab, sie sind nicht korreliert.

Der Korrelationskoeffizient wird auch Korrelationswert oder Produkt-Moment-Korrelation oder Bravais-Pearson-Korrelation oder Pearson-Korrelation genannt.

Korrelationskoeffizient gleitend

Bei der gleitenden Berechnung wird ein Fenster über den Datensatz Testdata geschoben. Das Fenster hat die Länge von Pattern. Die erste Position ist der linke Rand von Testdata. Das Fenster wird dann Punkt für Punkt vorgerückt. Für jeden Punkt wird der Korrelationskoeffizient ermittelt. Das Fenster wird nur soweit vorgeschoben, bis es am rechten Rand von Testdata angekommen ist.

Typisch wird die Funktion für diesen Zweck mit 3 Parametern aufgerufen, wobei Berechnung = 1 gesetzt ist.

Die Berechnung des Korrelationskoeffizient r erfolgt nach folgender Formel, wobei k die Verschiebung bedeutet. n ist die Länge von Pattern (x).

$$r_k = \frac{n\sum_{i=1}^{n} x_i y_{i+k} - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_{i+k})}{\sqrt{(n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2)(n\sum_{i=1}^{n} y_{i+k}^2 - (\sum_{i=1}^{n} y_{i+k})^2)}}$$

Bei gleitenden Berechnungen ergibt sich die Anzahl der Ergebniswerte aus [Länge(Testdata) - Länge(Pattern) + 1].

Das Ergebnis kann ein leerer Datensatz sein, falls aufgrund zu kurzer Eingangsdaten kein Rechenergebnis ermittelt werden kann.

Bei gleitender Berechnung hat das Ergebnis die Zeitbasis von Testdata, ist aber kürzer.

Bei gleitender Berechnung mit langem Pattern und noch viel längerem Testdata kann eine erhebliche Rechenzeit entstehen.

Formate

Die Eingangsdatensätze Pattern und Testdata müssen äquidistant sein.

Pattern und Testdata dürfen Events und Segmente haben. Falls sie Events und Segmente haben, müssen Pattern und Testdata die gleiche Struktur haben. Dann wird bei der Berechnung zu jedem Event/Segment von Testdata das zugehörige Event/Segment von Pattern zur Berechnung benutzt.

Alternativ darf Testdata Events und Segmente haben, während Pattern ein ganz normaler Datensatz ohne Events/Segmente ist. Dann wird bei der Berechnung jedes Event/Segment von Testdata mit ein und demselben Pattern korreliert.

Spezielle Berechnungen

Optional kann die Berechnung eines zusätzlichen Faktors F erfolgen. Der gleitende Korrelationskoeffizient wird damit gewichtet (multipliziert). Das Resultat ist r', der mit F gewichtete Korrelationskoeffizient:

$$r' = r * F \qquad 0 \leq F \leq 1$$

Der Sinn des zusätzlichen Faktors ist das Reduzieren der Korrelation bei kleinen Werten. Als Beispiel diene ein Muster, dessen Auftreten in einem langen Signal gesucht werden soll. An all den Stellen, an denen der Korrelationskoeffizient so etwa 1 ist, tritt dieses Muster auf. Ist nun das Signal ein gemessenes Signal, kann es Rauschen, auch in der Form des Klapperns von LSBs des Analog-Digital-Wandlers, aufweisen. Aber genau dieses Rauschen kann bei ansonsten etwa konstantem Signal dieselbe Form haben wie das gesuchte Muster. Da die Definition des Korrelationskoeffizienten das Ergebnis unabhängig von der Größe macht, tritt dann ein sehr hoher Korrelationskoeffizient auf.

Hier kommt der zusätzliche Faktor ins Spiel. Unterhalb einer festlegbaren Grenze verringert der Faktor das Ergebnis, oberhalb erhält er unverändert den errechneten Korrelationskoeffizienten.

Damit wird erreicht, dass bei ganz kleinen Signalbereichen kein hoher Wert für den Korrelationskoeffizienten entstehen kann.

Was als "klein" anzusehen ist, wird bei den verschiedenen Berechnungsarten des Faktors festgelegt.

Wenn eine der folgenden Berechnungen einen Faktor > 1 ergibt, wird der Faktor auf 1 begrenzt.

Korrelationskoeffizient F_rms

Innerhalb des betrachteten Fenster wird der Mittelwert von Testdata bestimmt, anschließend von Testdata abgezogen, davon der Effektivwert (RMS) berechnet.

$$F_{rms} = \frac{RMS\big(window - Average(window)\big)}{Parameter}$$

Der Parameter wird als Effektivwert gedeutet. Ist die mittlere quadratische Abweichung im Fenster kleiner als der Vergleichswert, erfolgt eine Reduktion (F < 1) des Ergebnisses.

Falls der Nenner = 0 ist, ist der resultierende Faktor 1.

Korrelationskoeffizient F_span

Innerhalb des betrachteten Fenster wird die Amplitudenspanne von Testdata bestimmt.

$$F_{span} = \frac{Max_{window} - Min_{window}}{Parameter}$$

Der Parameter wird als Spanne gedeutet. Ist die Spanne im Fenster kleiner als der Vergleichswert, erfolgt eine Reduktion (F < 1) des Ergebnisses.

Falls der Nenner = 0 ist, ist der resultierende Faktor 1.

Korrelationskoeffizient F_fred

Algorithmus, der über Spannen und relative Vorgabe ein F zur Reduktion von r ermittelt.

Innerhalb des betrachteten Fenster wird die Amplitudenspanne von Testdata bestimmt. Das ergibt Max_window und Min_window

Minimum und Maximum werden über den gesamten übergebenen Datensatz gebildet. Bei Segmenten und Events nicht über jedes einzelne Segment bzw. Event, sondern über einmal über alle Segmente bzw. Events. Das ergibt Max_total und Min_total.

$$F_{fred} = \frac{\frac{Max_{window} - Min_{window}}{Max_{total} - Min_{total}} \_ \frac{Parameter}{10}}{0.9 \cdot Paramter}$$

Der Parameter wird relative Spanne gedeutet. Die Berechnung erfolgt nach der Formel. Falls einer der Nenner = 0 ist, ist der resultierende Faktor 1.

Beispiele:

Berechnung des Korrelationskoeffizienten zweiter Zeitreihen und Prüfung auf gute Übereinstimmung

```
CoCo = CorrCoeff ( data1, data2 )
if CoCo > 0.9
end
```

Ein Muster soll in einem langen Datensatz gesucht werden.

```
pattern = [0,0,1,2,3,2,1,0,0]
mvc = CorrCoeff ( pattern, data, 1 )
interesting = xmaxi ( mvc, 0.9 )
```

Ein Pattern soll in einem langen Datensatz gesucht werden. Wenn es aber LSB-Klappern der Messdaten ist, soll es nicht gefunden werden. Das

LSB-Klappern hat einen Effektivwert von etwa Bereich 0.02

```
pattern = [0,0,1,2,3,2,1,0,0]
mvc = CorrCoeff ( pattern, data, 2, 0.05 )
```

Ein Pattern soll in einem langen Datensatz gesucht werden. Wenn ein Abschnitt des Datensatzes eine Spanne von 5% der Gesamtspanne des Datensatzes unterschreitet, soll geringer gewichtet werden.

```
pattern = [0,0,1,2,3,2,1,0,0]
mvc = CorrCoeff ( pattern, data, 0, 4, 0.05 )
```

Siehe auch:

KKF, AKF

cos

Kosinus, trigometrische Funktion

Deklaration:

```
cos ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Eingangsdaten (Winkel). Erlaubte Typen: [ND],[XY].
Ergebnis	Kosinus des Parameters

Beschreibung:

Es wird die trigonometrische Funktion cos berechnet, wobei im Bogen- oder Gradmaß entsprechend der Einheit des übergebenen Parameters gearbeitet wird.

Die x-Koordinate(n) von Ergebnis und Parameter sind gleich.

Anmerkungen

- Der Parameter der Funktion sollte keine Einheit oder die Einheiten 'Rad', 'Grad', '° ' haben. Bei anderen Einheiten wird eine Warnung erzeugt und die Einheit ins Ergebnis übernommen.
- Der Parameter darf strukturiert sein (Events/ Segmente).
- Die zugehörige Umkehrfunktion heißt acos().

Beispiele:

cos(0)= 1, cos(PI/2)= 0, cos(PI)= -1

```
one = cos(0.0)
```

Erzeugt einen cos-förmigen Datensatz:

```
NDcos = cos(Ramp(0, 0.2, 100))
```

Bei entsprechender Einheit wird der Parameter im Gradmaß verstanden:

```
zero = cos(90 '°')
zero = cos(90 'Grad')
```

Siehe auch:

sin, tan, acos

CrossPowerDS

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Kreuzleistungsdichtespektrum (Cross Power Density) mit gleitendem Fenster und linearer Mittelung. Berechnung mittels FFT.

Deklaration:

```
CrossPowerDS ( Eingangskanal, Ausgangskanal, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangskanal	Der Bezugskanal, Eingangskanal. Die Anregung eines Systems. In Sekunden skaliert.
Ausgangskanal	Der (verzögerte) Ausgangskanal. Die Antwort eines Systems. Ein- und Ausgangskanal haben dieselbe Zeitbasis und sind in Sekunden skaliert.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	0 : Keine Mittelung
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung von Real- und Imaginärteil). Der Mittelwert wird über so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	2 : Peak Hold Max ab Beginn, Maximalwerte, berechnet von allen bislang berechneten Spektren
	3 : Peak Hold Max pro Intervall, Maximalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	4 : Peak Hold Min ab Beginn, Minimalwerte, berechnet von allen bislang berechneten Spektren
	5 : Peak Hold Min pro Intervall, Minimalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase.

Beschreibung:

Konjugiert komplexes RMS-Spektrum des Eingangskanals, multipliziert mit komplexem RMS-Spektrum des Ausgangskanals, dividiert durch Frequenzlinienabstand.

Die Berechnung der linearen Mittelung erfolgt über Real- und Imaginärteil getrennt.

Bei Peak Hold Berechnungen wird das Peak Hold auf den Betrag angewendet. Die Phase wird gemittelt, ist aber ohnehin dann bedeutungslos und sollte ignoriert werden.

Das Ergebnis wird durch die ENBW (Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch 1.5 im Fall des Hanning-Fensters.

Beispiele:

```
Kreuzleistungsdichte = CrossPowerDS ( Kraft, Bewegung, 1000, 0, 50, 1, 0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Eine Kraft wirkt auf ein mechanisches Teil. Die Bewegung des Bauteils am anderen Ende wird gemessen.

```
Kreuzleistungsdichte = CrossPowerDS ( Kraft, Bewegung, 2048, 1, 0, 10, 1, 0 )
```

Das ist die Berechnung einer Folge von 2048 Punkte-Spektren mit Hamming-Fenster. Über je 10 Spektren wird der Mittelwert bestimmt und im Ergebnis vermerkt.

Siehe auch:

CrossPowerDS_exp, CrossPowerDS_1, CrossPowerNorm

CrossPowerDS_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein gemitteltes Kreuzleistungsdichtespektrum (Cross Power Density) wird bestimmt. Die Mittelung erfolgt über viele Spektren, die aus einem gleitenden Fenster ermittelt wurden.

Deklaration:

```
CrossPowerDS_1 ( Eingangskanal, Ausgangskanal, Fensterbreite, Fenstertyp, Überlappung, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangskanal	Der Bezugskanal, Eingangskanal. Die Anregung eines Systems. In Sekunden skaliert.
Ausgangskanal	Der (verzögerte) Ausgangskanal. Die Antwort eines Systems. Ein- und Ausgangskanal haben dieselbe Zeitbasis und sind in Sekunden skaliert.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung von Real- und Imaginärteil). Der Mittelwert wird über alle berechneten Spektren gebildet.
	2 : Peak Hold Max, Maximalwerte, berechnet von allen berechneten Spektren
	4 : Peak Hold Min, Minimalwerte, berechnet von allen berechneten Spektren
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ein gemitteltes Spektrum als Ergebnis. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase.

Beschreibung:

Konjugiert komplexes RMS-Spektrum des Eingangskanals, multipliziert mit komplexem RMS-Spektrum des Ausgangskanals, dividiert durch Frequenzlinienabstand

.Die Berechnung der Mittelung erfolgt über Real- und Imaginärteil getrennt.

Bei Peak Hold Berechnungen wird das Peak Hold auf den Betrag angewendet. Die Phase wird gemittelt, ist aber ohnehin dann bedeutungslos und sollte ignoriert werden.

Das Ergebnis wird durch die ENBW (Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch 1.5 im Fall des Hanning-Fensters.

Beispiele:

```
Kreuzleistungsdichte = CrossPowerDS_1 ( Kraft, Bewegung, 1000, 0, 50, 1, 0 )
```

Das ist die Berechnung eines gemittelten Spektrums. Die Mittelung geschieht über eine Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Die Kanäle enthalten etwa 20000 Messwerte. Eine Kraft wirkt auf ein mechanisches Teil. Die Bewegung des Bauteils am anderen

Ende wird gemessen.

Siehe auch:

CrossPowerDS, CrossPowerDS_exp, CrossPowerNorm_1

CrossPowerDS_exp

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Kreuzleistungsdichtespektrum (Cross Power Density) mit gleitendem Fenster und exponentieller Mittelung. Berechnung mittels FFT.

Deklaration:

```
CrossPowerDS_exp ( Eingangskanal, Ausgangskanal, Fensterbreite, Fenstertyp, Überlappung, Reduktion,
Zeitkonstante [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangskanal	Der Bezugskanal, Eingangskanal. Die Anregung eines Systems. In Sekunden skaliert.
Ausgangskanal	Der (verzögerte) Ausgangskanal. Die Antwort eines Systems. Ein- und Ausgangskanal haben dieselbe Zeitbasis und sind in Sekunden skaliert.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Zeitkonstante	Die Zeitkonstante für die exponentielle Mittelung. In Sekunden angegeben.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von Spektren, segmentierter Datensatz. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase.

Beschreibung:

Konjugiert komplexes RMS-Spektrum des Eingangskanals, multipliziert mit komplexem RMS-Spektrum des Ausgangskanals, dividiert durch Frequenzlinienabstand.

Das Ergebnis wird durch die ENBW (Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch 1.5 im Fall des Hanning-Fensters.

Beispiele:

```
Kreuzleistungsdichte = CrossPowerDS_exp ( Kraft, Bewegung, 1000, 0, 50, 2, 40.0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Eine Kraft wirkt auf ein mechanisches Teil. Die Bewegung des Bauteils am anderen Ende wird gemessen. Beide Kanäle haben eine Abtastzeit von 10ms. Also wird alle 5s ein Spektrum gebildet. Diese werden mit einer Zeitkonstante von 40.0s geglättet. Jedes 2. Spektrum wird ausgegeben.

Siehe auch:

CrossPowerDS, CrossPowerDS_1

CrossPowerNorm

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Normiertes Kreuzleistungsdichtespektrum (Normalized Cross Power Spectrum) mit gleitendem Fenster und linearer Mittelung. Berechnung mittels FFT.

Deklaration:

```
CrossPowerNorm ( Eingangskanal, Ausgangskanal, Fensterbreite, Fenstertyp, Überlappung, Reduktion [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangskanal	Der Bezugskanal, Eingangskanal. Die Anregung eines Systems. In Sekunden skaliert.
Ausgangskanal	Der (verzögerte) Ausgangskanal. Die Antwort eines Systems. Ein- und Ausgangskanal haben dieselbe Zeitbasis und sind in Sekunden skaliert.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase.

Beschreibung:

Konjugiert komplexes RMS-Spektrum des Eingangskanals, multipliziert mit komplexem RMS-Spektrum des Ausgangskanals.

Das Ergebnis ist auf das Produkt der Effektivwerte der Eingangssignale normiert. Bereich des Resultats: -1 .. +1

Die Effektivwerte der Kanäle werden nach der Fensterung berechnet.

Die Berechnung der Mittelung erfolgt über Real- und Imaginärteil getrennt.

Die Art der Mittelung ist das arithmetisches Mittel (lineare Mittelung).

Beispiele:

```
NormKreuzleistung = CrossPowerNorm ( Kraft, Bewegung, 1000, 0, 50, 1, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Eine Kraft wirkt auf ein mechanisches Teil. Die Bewegung des Bauteils am anderen Ende wird gemessen.

```
NormKreuzleistung = CrossPowerNorm ( Kraft, Bewegung, 2048, 1, 0, 10, 0 )
```

Das ist die Berechnung einer Folge von 2048 Punkte-Spektren mit Hamming-Fenster. Über je 10 Spektren wird der Mittelwert bestimmt und im Ergebnis vermerkt.

Siehe auch:

CrossPowerNorm_1, CrossPowerDS

CrossPowerNorm_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein gemitteltes normiertes Kreuzleistungsdichtespektrum (Normalized Cross Power Spectrum) mit gleitendem Fenster und linearer Mittelung. Berechnung mittels FFT.

Deklaration:

```
CrossPowerNorm_1 ( Eingangskanal, Ausgangskanal, Fensterbreite, Fenstertyp, Überlappung [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangskanal	Der Bezugskanal, Eingangskanal. Die Anregung eines Systems. In Sekunden skaliert.
Ausgangskanal	Der (verzögerte) Ausgangskanal. Die Antwort eines Systems. Ein- und Ausgangskanal haben dieselbe Zeitbasis und sind in Sekunden skaliert.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ein gemitteltes Spektrum als Ergebnis. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase.

Beschreibung:

Konjugiert komplexes RMS-Spektrum des Eingangskanals, multipliziert mit komplexem RMS-Spektrum des Ausgangskanals.

Das Ergebnis ist auf das Produkt der Effektivwerte der Eingangssignale normiert. Bereich des Resultats: -1 .. +1

Die Effektivwerte der Kanäle werden nach der Fensterung berechnet.

Die Berechnung der Mittelung erfolgt über Real- und Imaginärteil getrennt.

Die Art der Mittelung ist das arithmetisches Mittel (lineare Mittelung).

Die Mittelung erfolgt über viele Spektren, die aus einem gleitenden Fenster ermittelt wurden.

Beispiele:

```
NormKreuzleistung = CrossPowerNorm_1 ( Kraft, Bewegung, 1000, 0, 50, 0 )
```

Das ist die Berechnung eines gemittelten Spektrums. Die Mittelung geschieht über eine Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Die Kanäle haben etwa 20000 Messwerte. Eine Kraft wirkt auf ein mechanisches Teil. Die Bewegung des Bauteils am anderen Ende wird gemessen.

Siehe auch:

CrossPowerNorm, CrossPowerDS_1

CURVESETUP

Datensatz mit einer gegebenen Kurvenfenster-Konfiguration darstellen
Alternativer Name: **KURVELADEN**

Der Befehl ist veraltet, statt dessen sollte die leistungsfähigere Funktion CwLoadCCV() verwendet werden.

Deklaration:

```
CURVESETUP Variable Dateiname
```

Parameter:

Variable	Name der darzustellenden Variable
Dateiname	Name der zu ladenden Kurvenkonfigurations-Datei.

Beschreibung:

Ein Datensatz wird als Kurvenfenster in einer vorgegebenen Kurvenfenster-Konfiguration dargestellt.

Die Datei mit der Kurven-Konfiguration (*.ccv) muss mit einem Kurvenfenster über den Menüpunkt "Sichern unter" (siehe Dokumentation zum Kurvenfenster) erstellt worden sein.

- Wenn sich die Datei nicht im Standardverzeichnis befindet, kann auch ein vollständiger Pfad angegeben werden.
- Das Standardverzeichnis für Kurvenkonfigurationen stellen Sie im Dialog "Optionen"/"Verzeichnisse" ein.

Beispiele:

```
CURVESETUP Anstieg Konfig1
CURVESETUP Anstieg Konfig1.ccv
CURVESETUP Anstieg c:\imc\ccv\Konfig1.ccv
```

Drei Möglichkeiten, eine Kurvenfenster-Konfiguration zu laden und in der Kurvenfenster-Konfiguration den Datensatz "Anstieg" darzustellen.

Siehe auch:

CwLoadCCV

Cut

Herausschneiden eines Abschnitts eines Datensatzes in gegebenen x-Grenzen
Alternativer Name: **Gren**

Deklaration:

```
Cut ( Daten, EwIntervallVorn, EwIntervallHinten [, IntervallDeutung] ) -> Stück
```

Parameter:

Daten	Zu schneidender Datensatz. Erlaubte Typen: [ND],[XY].
EwIntervallVorn	Koordinate des ersten herauszuschneidenden Wertes.
EwIntervallHinten	Koordinate des letzten herauszuschneidenden Wertes.
IntervallDeutung	Gibt an, wie die angegebenen Intervallgrenzen zu interpretieren sind (als x-Koordinaten oder als absolute Zeit-Angabe). (optional , Standardwert: 0)
	0 : Als x-Koordinate
	1 : Als absolute Zeit
Stück	Teilstück des Parameter-Datensatzes

Beschreibung:

Aus einem Datensatz kann ein Stück herauskopiert werden, um gesondert weiterverarbeitet zu werden. Zum Kennzeichnen dieses Abschnittes werden sein Anfang und Ende als zweiter und dritter Parameter übergeben. Diese beiden Grenzen sind als x-Koordinaten anzugeben.

Normaler Datensatz:

Liegen Anfang oder Ende des Abschnittes nicht mehr innerhalb des Ausdehnungsbereichs des übergebenen Datensatzes, so wird der Datensatz entsprechend verlängert. Dabei wird mit Nullen aufgefüllt.

Trifft die vordere Intervallgrenze nicht genau einen Abtastwert des Datensatzes, wird auf den vorhergehenden Abtastwert abgerundet. Dabei wird mit einer leichten Unschärfe gearbeitet. Ist der gesuchte Wert also nur geringfügig kleiner als ein Abtastwert, wird trotzdem der Wert an dieser Position zurückgegeben. Dieses Verhalten dient dazu, um numerische Abweichungen bei zuvor berechneten Intervallgrenzen auszugleichen.

Für IntervallDeutung == 0 (Angabe in x-Koordinaten): Ein angegebener x-Wert gilt auch noch als genauer Treffer eines Abtastwertes, wenn er weniger als 1/10000 eines Abtastschrittes vor diesem Abtastwert liegt.

Für IntervallDeutung == 1 (Angabe in absoluter Zeit): Um die Differenz zu bestimmen, die ein Startwert von einem Abtastpunkt entfernt liegen darf, um noch als genauer Treffer zu gelten, wird sowohl 1/10000 eines Abtastschrittes sowie das Produkt der vorderen Intervallgrenze mit 1e-14 berechnet und der größere der beiden Werte verwendet. Der zweite Ausdruck wird u.U. bei Werten von hohem Betrag wirksam, z.B. wenn eine aktuelle Zeit angegeben wird. LSB-Fehler bewirken hier Abweichungen im Mikrosekunden-Bereich, so dass ein 1/10000 der Abtastzeit als Unschärfe schon zu knapp sein könnte.

Trifft die hintere Intervallgrenze nicht genau einen Abtastwert des Datensatzes, wird der nächstliegende Abtastwert verwendet.

XY-Datensatz:

Liegen Anfang oder Ende des Abschnittes nicht mehr innerhalb des Ausdehnungsbereichs des übergebenen Datensatzes, so wird das Resultat auf diesem Anfangs- bzw. Endwert begrenzt. Wenn eine x-Grenze nicht genau an einer Stützstelle des Datensatzes liegt, wird der y-Wert an dieser Stelle linear interpoliert.

- Die Einheiten des zweiten und dritten Parameters sollten sinnvollerweise der x-Einheit des Datensatzes entsprechen.
- Die Einheit des Datensatzes wird nicht verändert.
- Der x-Offset des erzeugten Datensatzes wird angepasst.
- Der zweite Parameter darf mit dem dritten vertauscht werden. Der kleinere Wert wird als untere Grenze verstanden.
- Sie können alternativ mit der Funktion CutIndex() arbeiten, wenn Sie die Grenzen über den Index der Punkte im Datensatz angeben wollen.
- Eine weitere Alternative ist die Funktion CutDt(). Der wesentliche funktionale Unterschied zur Funktion Cut() ist, dass ein am Ende des Schnitt-Bereichs liegender Abtastpunkt nicht im Ergebnis enthalten ist. Die Funktion CutDt() ist somit oft einfacher anzuwenden, wenn in einer Schleife nacheinander aufeinanderfolgende Abschnitte eines Datensatzes extrahiert werden sollen, da keine Überlappungseffekte an den Fenstergrenzen auftreten.
- Sie können einen Ausschnitt eines Datensatzes auch erzeugen, indem Sie in einem Kurvenfenster ein Messwertfenster erzeugen und dort den Bereich zwischen den Messcursoren exportieren. Dieses hat den Vorteil, dass Sie grafisch arbeiten können. Deshalb ist diese Vorgehensweise oft bequemer, aber für eine automatische Berechnung in einer Sequenz ungeeignet. Siehe Handbuch zum Kurvenmanager, Kapitel 'Kurvenfenster/Messen'.

Beispiele:

Aus einem Datensatz mit der Ausdehnung 0s ... 100s wird der Bereich 10s ... 20.5s herausgeschnitten:

```
NDpart = Cut(NDdata, 10 's', 20.5 's')
```

Ein Datensatz mit der Ausdehnung 0s ... 100s wird nach vorn und hinten mit Nullen verlängert, so dass er sich von -10s bis 200s ausdehnt.

```
NDlonger = Cut(NDdata, -10 's', 200 's')
```

Aus einem äquidistanten Datensatz, der sich zeitlich über mehrere Tage erstreckt, wird der Auschnitt für einen Tag gebildet:

```
time1 = TimeJoin(18, 7, 2018, 0, 0, 0)
time2 = TimeJoin(19, 7, 2018, 0, 0, 0)
Day_18_07_2018 = Cut(datagroup, time1, time2, 1)
```

Achtung: Im vorliegenden Beispiel wird in den meisten Fällen der erste Messwert vom 20.7. mit im Ergebnis enthalten sein, da der nächstliegende Wert verwendet wird, wenn die hintere Grenze nicht genau einen Abtastwert trifft.

Siehe auch:

CutIndex, CutDt, Value2, ValueIndex, Repl, ReplIndex

CutDt

Herausschneiden eines Abschnitts eines Datensatzes unter Angabe von Startpunkt und Breite.

Deklaration:

```
CutDt ( Daten, Start, Breite [, StartDeutung] [, StartRundung] [, Extrapolation] ) -> Stück
```

Parameter:

Daten	Zu schneidender Datensatz. Äquidistant abgetastet. Keine Events, keine Segmente.
Start	Beginn des Ausschnitts in x-Koordinaten oder absoluter Zeit.
Breite	Breite des Ausschnitts in x-Koordinaten. Es wird auf ein Vielfaches der Abtastzeit bzw. des Inkrements in x-Richtung gerundet.
StartDeutung	Gibt an, wie der Start-Parameter zu interpretieren ist (als x-Koordinate oder als absolute Zeit-Angabe). (optional , Standardwert: 0)
	0 : Als x-Koordinate
	1 : Als absolute Zeit
StartRundung	Steuert die Bestimmung des ersten auszuschneidenden Wertes, wenn der angegebene Startwert nicht genau einen Abtastpunkt des Datensatzes trifft (also zwischen Abtastpunkten des Signals liegt). (optional , Standardwert: 0)
	0 : Es wird der näher am Startwert gelegene Abtastpunkt verwendet.
	1 : Es wird der auf den Startwert direkt folgende Abtastpunkt verwendet.
	2 : Es wird der direkt vor dem Startwert liegende Abtastpunkt verwendet.
Extrapolation	Steuert das Verhalten der Funktion, wenn sich Start oder Ende des auszuschneidenden Bereiches außerhalb der Ausdehnung des Datensatzes befinden. (optional , Standardwert: 0)
	0 : Es wird nicht extrapoliert. Es werden nur die tatsächlich im Bereich liegenden realen Werte geliefert. Das Ergebnis ist ggf. kürzer als erwartet.
	1 : Die fehlenden Stützwerte am Beginn oder Ende werden mit 0 initialisiert.
	2 : Der letzte bzw. erste Wert des Datensatz wird nach hinten bzw. vorn verlängert.
Stück	Teilstück des Parameter-Datensatzes

Beschreibung:

Aus einem Datensatz kann ein Stück herauskopiert werden, um gesondert weiterverarbeitet zu werden. Zum Kennzeichnen dieses Abschnittes wird sein Start (in x-Koordinaten oder absoluter Zeit) und seine Länge (in x-Koordinaten) übergeben.

Die Funktion kann nur auf gleichmäßig abgetastete Daten angewendet werden.

Die angegebene Länge wird auf ein Vielfaches der Abtastzeit des Datensatzes gerundet.

Die Länge des Ergebnisses berechnet sich aus dem Quotienten aus der (auf Vielfaches der Abtastzeit gerundeten) Breite und der Abtastzeit. Der genau am Ende des Bereichs liegende Abtastwert ist also nicht enthalten.

Beispiel: Datensatz [data], Abtastzeit 1s, Offset 0s

```
result = CutDt(data, 0, 10, 0, 0)
```

Das Ergebnis hat 10 Werte (zu den x-Koordinaten 0..9s). Der Abtastwert an der Stelle 10s ist nicht mehr enthalten.

- Der x-Offset des erzeugten Datensatzes wird angepasst, die Triggerzeit bleibt unverändert.
- Die ähnlichen Funktionen Cut() und CutIndex() bieten alternative Möglichkeiten zur Angabe des auszuschneidenden Bereichs (mittels Start/Ende-Koordinate bzw. Start-Index/Wert-Anzahl).
- Ein wesentlicher funktionaler Unterschied zur Funktion Cut() ist, dass ein am Ende des Schnitt-Bereichs liegender Abtastpunkt nicht im Ergebnis enthalten ist. Die Funktion CutDt() ist somit oft einfacher anzuwenden, wenn in einer Schleife nacheinander aufeinanderfolgende Abschnitte eines Datensatzes extrahiert werden sollen, da keine Überlappungseffekte an den Fenstergrenzen auftreten.
- Im Vergleich zur Funktion CutIndex() ist CutDt() überlegen, wenn Datensätze mit unterschiedlichen Triggerzeiten/Abtastraten bzw. unterschiedlichem x-Offset/x-Inkrement gleichzeitig geschnitten werden sollen.
- Sie können einen Ausschnitt eines Datensatzes auch erzeugen, indem Sie in einem Kurvenfenster ein Messwertfenster erzeugen und dort den Bereich zwischen den Messcursoren exportieren. Dieses hat den Vorteil, dass Sie grafisch arbeiten können. Deshalb ist diese Vorgehensweise oft bequemer, aber für eine automatische Berechnung in einer Sequenz ungeeignet. Siehe Handbuch zum Kurvenmanager, Kapitel 'Kurvenfenster/Messen'.

Wenn der Parameter [StartRundung] ungleich 0 ist, also ggf. der vorhergehende oder nachfolgende Abtastwert verwendet werden soll, und der Startwert nicht genau einen Abtastwert des Datensatzes trifft, so wird mit einer leichten Unschärfe gearbeitet. Ist also z.B. bei [StartRundung] = 2 der gesuchte Wert nur geringfügig kleiner als ein Abtastwert, wird trotzdem der Wert an dieser Position zurückgegeben. Dieses Verhalten dient dazu, um numerische Abweichungen bei zuvor berechneten X-Koordinaten auszugleichen.

Um die Differenz zu bestimmen, die ein Startwert von einem Abtastpunkt entfernt liegen darf, um noch als genauer Treffer zu gelten, wird sowohl 1/10000 eines Abtastschrittes sowie das Produkt des Startwertes mit 1e-14 berechnet und der größere der beiden Werte verwendet. Der zweite Ausdruck wird u.U. bei Startwerten von hohem Betrag wirksam, z.B. wenn eine absolute, aktuelle Zeit angegeben wird. LSB-Fehler bewirken hier Abweichungen im Mikrosekunden-Bereich, so dass ein 1/10000 der Abtastzeit als Unschärfe schon zu knapp sein könnte.

Beispiele:

Die Datengruppe [datagroup] im folgenden Beispiel enthält mehrere Kanäle einer mehrtägigen Langzeitaufnahme. Die Abtastzeiten der Kanäle sind verschieden. Es werden die Messwerte für 3 aufeinanderfolgende Tage herausgeschnitten:

```
time = TimeJoin(18, 7, 2018, 0, 0, 0)
secondsPerDay = 60*60*24
Day_18_07_2018 = CutDt(datagroup, time, secondsPerDay, 1)
Day_19_07_2018 = CutDt(datagroup, time+ secondsPerDay, secondsPerDay, 1, 1)
Day_20_07_2018 = CutDt(datagroup, time+ 2*secondsPerDay, secondsPerDay, 1, 1)
```

Der Datensatz [data] im folgenden Beispiel habe die Offset =0s, Abtastzeit =1s.

```
x5_to_14 = CutDt(data, 5, 10)         ; cut 10 samples from x=5 to 14
x5_to_15 =   Cut(data, 5, 15)         ; cut 11 samples from x=5 to 15

x5_to_14 = CutDt(data, 5.2, 10)       ; cut 10 samples from x=5 to 14
x6_to_15 = CutDt(data, 5.2, 10, 0, 1) ; cut 10 samples from x=6 to 15
x5_to_14 = CutDt(data, 5.2, 10, 0, 2) ; cut 10 samples from x=5 to 14
x5_to_15 =   Cut(data, 5.2, 15.2)     ; cut 11 samples from x=5 to 15

x5_to_14 = CutDt(data, 4.7, 10)       ; cut 10 samples from x=5 to 14
x5_to_14 = CutDt(data, 4.7, 10, 0, 1) ; cut 10 samples from x=5 to 14
x4_to_13 = CutDt(data, 4.7, 10, 0, 2) ; cut 10 samples from x=4 to 13
x4_to_15 =   Cut(data, 4.7, 14.7)     ; cut 12 samples from x=4 to 15

x0_to_3  = CutDt(data, -1, 5, 0, 0, 0) ; cut 4 samples from x=0 to 3
xm1_to_3 = CutDt(data, -1, 5, 0, 0, 1); cut 5 samples from x=-1 to 3. xm1_to_3[1] = 0
xm1_to_3 = CutDt(data, -1, 5, 0, 0, 2); cut 5 samples from x=-1 to 3. xm1_to_3[1] = data[1]
```

Siehe auch:

CutIndex, Cut, Value2, ValueIndex, Repl, ReplIndex, MatrixPart

CutIndex

Herausschneiden eines Abschnitts eines Datensatzes, Angabe der Grenzen durch die Indizes der Punkte im Datensatz.
Alternativer Name: **GrenIndex**

Deklaration:

```
CutIndex ( Daten, EwStartIndex, EwEndeIndex ) -> Stück
```

Parameter:

Daten	Zu schneidender Datensatz. Erlaubte Typen: [ND],[XY].
EwStartIndex	Index des ersten herauszuschneidenden Wertes.
EwEndeIndex	Index des letzten herauszuschneidenden Wertes.
Stück	Teilstück des Parameter-Datensatzes

Beschreibung:

Aus einem Datensatz wird ein Teilstück kopiert. Die Festlegung des Teilstücks erfolgt durch die Angabe der Indizes (Positionen) des ersten und des letzten zu kopierenden Wertes. Der Originaldatensatz wird nicht verändert.

Die Position (der Index) [EwStartIndex] muss zwischen 1 und der Datensatzlänge von [Daten] liegen. Es wird bis einschließlich [EwEndeIndex] ausgeschnitten. Wenn die beiden angegebenen Indizes gleich sind, wird also genau 1 Wert kopiert. Wenn [EwEndeIndex] größer als die Datensatzlänge ist, wird bis zum Ende von [Daten] ausgeschnitten.

Der Datentyp des ersten Parameters und des Ergebnisses sind gleich. Der x-Offset des erzeugten Datensatzes wird angepasst.

Der zweite Parameter darf mit dem dritten vertauscht werden. Der kleinere Wert wird als untere Grenze verstanden.

Alternativ können die Funktionen Cut() oder CutDt() verwendet werden, bei der die Grenzen durch x-Koordinaten vorgegeben werden.

Um einen einzelnen Wert über dessen Index im Datensatz abzufragen, können Sie den Datensatz in Formeln auch indizieren:

```
singleValue = Data[ Index ]
```

Sie können einen Ausschnitt eines Datensatzes auch erzeugen, indem Sie in einem Kurvenfenster ein Messwertfenster erzeugen und dort den Bereich zwischen den Messcursoren exportieren. Dieses hat den Vorteil, dass Sie grafisch arbeiten können. Deshalb ist diese Vorgehensweise oft bequemer, aber für eine automatische Berechnung in einer Sequenz ungeeignet. Siehe Handbuch zum Kurvenmanager, Kapitel 'Kurvenfenster/Messen'.

Beispiele:

Aus einem Datensatz wird das Stück vom 10. bis zum letzten Wert kopiert. Das Stück wird geglättet und in den Datensatz zurückkopiert:

```
DataPart = CutIndex(Data, 10, Leng?(Data))
DataPart = Smo5(DataPart)
Data = ReplIndex(Data, DataPart, 10)
```

Die folgenden Formeln sind äquivalent:

```
Sample2 = CutIndex(Data, 2, 2)
Sample2 = ValueIndex(Data, 2)
Sample2 = Data[2]
```

Siehe auch:

Cut, CutDt, Value2, ValueIndex, Repl, ReplIndex, MatrixPart, SamplesGate

CvUpdate

Aktualisierung der Kurvenfenster sperren
Alternativer Name: **KvUpdate**

Die Funktion ist veraltet, statt dessen sollte die Funktion CwUpdateEnable() verwendet werden.

Deklaration:

```
CvUpdate ( EwUpdate )
```

Parameter:

EwUpdate	Aktualisierung an/aus
	0 : Aktualisieren der Kurvenfenster sperren
	1 : Aktualisieren der Kurvenfenster (wieder) erlauben

Beschreibung:

Wenn [EWUpdate] ungleich 0 gesetzt ist, werden während des Ablaufs einer Sequenz WM_PAINT- und andere Nachrichten erlaubt und imc FAMOS ist bedienbar.

Sonst nicht! So kann z. B. mehrmaliges Updaten eines Kurvenfensters bei Neugestaltung verhindert werden.

Achtung:

[EwUpdate] sollte nur vor einer Gruppe von Funktionen zur Konfigurierung eines Kurvenfensters auf 0 gesetzt werden und gleich danach wieder auf 1. Schweres Fehlverhalten ist bei unsachgemäßer Anwendung der Funktion möglich!!!

Beispiele:

```
CvUpdate(0)
CvYAxis(...)
CvYAxis(...)
CvUpdate(1)
```

Siehe auch:

CwUpdateEnable

CwAction

Anwendungsbereich: Kurvenfenster

Führt eine Aktion am selektierten Kurvenfenster aus.

Deklaration:

```
CwAction ( Aktion )
```

Parameter:

Aktion	Welche Aktion soll ausgeführt werden?
	"axes.fix" : Achsen fixieren
	"clipboard.copy" : Kopieren in Ablage
	"cosys.height.auto" : Alle Höhen der Koordinatensysteme werden auf automatisch gestellt.
	"delete.harmonic" : Alle Marker, die den harmonischen Cursor bilden, löschen
	"delete.lines" : Alle Linien samt Daten aus dem Fenster entfernen
	"delete.markers" : Alle Marker löschen
	"dialog.display" : Dialog starten: Darstellung
	"dialog.lines" : Dialog starten: Linien
	"dialog.more channels" : Dialog starten: Weitere Datensätze
	"dialog.x-axis" : Dialog starten: x-Achse und andere Achsen
	"history.reset" : Historie löschen
	"link.remove" : x-Link zu anderen Kurvenfenstern entfernen
	"map.fit.axes" : Bei Landkarte aus Internet Anpassen der Achsen an die Landkarte für eine gute Lesbarkeit
	"map.load" : Veraltet, bitte CwActionP benutzen! Laden einer Datei mit Hintergrundbild. Der Dateiname ist durch CwDisplaySet ("map.filename" ...) gerade vorher gesetzt worden.
	"measure.close" : Messwertfenster schließen
	"measure.invisible" : Messwertfenster unsichtbar öffnen, wobei die Messcursoren aber angezeigt werden
	"measure.show" : Messwertfenster anzeigen
	"optimize" : Optimieren (Leere Kanäle entfernen)
	"overview.close" : Übersichtsfenster schließen
	"overview.show" : Übersichtsfenster anzeigen
	"print" : Drucken
	"reset" : Reset
	"slavepointer.reset" : Schleppzeiger reset
	"start.link select" : Start des Modus: Verbinden (Link) mit anderem Kurvenfenster
	"start.modify values" : Start des Modus: Messpunkte verändern
	"start.zoom" : Start des Modus: Zoom
	"win.close" : Ein frei fliegendes Kurvenfenster wird geschlossen.
	"win.disable" : Disable des Kurvenfesters: Es ist mit Maus (und Tastatur) nicht mehr bedienbar.
	"win.enable" : Enable des Kurvenfesters: Es ist mit Maus (und Tastatur) bedienbar.
	"win.hide" : Ein frei fliegendes Kurvenfenster wird versteckt.
	"win.icon" : Ein frei fliegendes Kurvenfenster wird als Icon dargestellt.
	"win.maximize" : Ein frei fliegendes Kurvenfenster wird maximiert dargestellt.
	"win.show" : Ein frei fliegendes Kurvenfenster wird gezeigt, falls es vorher versteckt war.
	"win.sizenormal" : Ein frei fliegendes Kurvenfenster, das maximiert oder als Icon dargestellt ist, wird in normaler Größe dargestellt.
	"win.twin" : Ein frei fliegendes Zwillingsfenster erzeugen

	"unzoom" : Rezoom

Beschreibung:

Beispiele:

Kurvenfenster schließen

```
CwSelectWindow("curve1")
CwAction("win.close")
```

Siehe auch:

CwActionP

CwActionP

Anwendungsbereich: Kurvenfenster

Führt eine Aktion (mit Zusatzparameter) am selektierten Kurvenfenster aus.

Deklaration:

```
CwActionP ( Aktion, Parameter1, Parameter2 )
```

Parameter:

Aktion	Welche Aktion soll ausgeführt werden?
	"export.graphics" : Exportieren der Grafik in eine Datei, z.B. .bmp, .png, .jpg oder .pdf. Der Dateiname wird in Parameter1 angeben.
	"export.pdf.append" : Anhängen der Grafik an eine bestehende pdf-Datei. Der Dateiname wird in Parameter1 angeben. Bei nicht vorhandener Datei wird sie neu angelegt.
	"link.set" : x-Link zu einem anderen Kurvenfenster herstellen
	"map.load" : Laden einer Datei mit Hintergrundbild. Der Dateiname wird in Parameter1 angeben.
Parameter1	Parameter1. Bedeutung abhängig von Aktion
Parameter2	Parameter2. = 0, wenn nicht benutzt

Beschreibung:

Wenn eine Grafik exportiert wird, muss in einigen Fällen das Kurvenfenster sichtbar sein. Wird z.B. ein neu erzeugtes Kurvenfenster versteckt dargestellt und eine exakte Bildschirm-Kopie verlangt, so fehlt die definierte Größe.

Beispiele:

Link zwischen 2 Kurvenfenstern herstellen

```
CwSelectWindow("curve1")
CwActionP("link.set", "curve2", 0)
```

PNG-Datei erstellen

```
CwActionP("export.graphics", "c:\1.png", 0)
```

PDF-Datei erstellen und 2. Seite anhängen

```
CwActionP("export.graphics",   "c:\1.pdf", 0)
CwActionP("export.pdf.append", "c:\1.pdf", 0)
```

Siehe auch:

CwAction

CwAxisGet

Anwendungsbereich: Kurvenfenster

Eigenschaft einer Achse abfragen

Deklaration:

```
CwAxisGet ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"range" : Bereich (1 auto, 2 auto mit null, 3 runden, 4 fest, 5 wie vorherige)
	"scale" : Skala (1 Standard linear, 2 dB, 3 logarithmisch, 4 absolute Zeit, 5 relative Zeit, 6 Terzen)
	"max" : Tatsächlicher aktueller Wert des Maximums.
	"min" : Tatsächlicher aktueller Wert des Minimums.
	"count.data" : Anzahl Datenelemente, die in dieser Achse enthalten sind.
	"count.line" : Anzahl Linien, die in dieser y-Achse enthalten sind.
	"count.userticks" : Anzahl eigene Ticks, die zu dieser Achse vorhanden sind.
	"description.color" : Farbe des Zusatztextes, Format siehe rgb(), (-1 automatisch)
	"description.distance" : Wie weit entfernt von der Achse soll der Zusatztext platziert werden. In mm, auch < 0.
	"description.font.size" : Größe der Schrift für den Zusatztext, in pt, z.B. 8, (0 auto)
	"description.option" : Soll ein zusätzlicher Text an der Achse angezeigt werden? (0 nein, 1 fester Text, 2 Einheit, 3 [Einheit], 4 Name Einheit, 5 definierbar mit Platzhaltern)
	"description.orientation" : In welche Richtung soll der Zusatztext verlaufen (0 auto, 1 parallel zur Achse, 2 quer zur Achse)
	"description.pos" : Wo genau soll der Zusatztext angebracht werden? (0 auto, 1 zentriert, 2 bündig mit Achsenbeginn, 3 bündig mit Achsenende)
	"description.symbol" : Soll das Symbol der Linie wie in einer Legende dargestellt werden? (0 nein, 1 ja davor)
	"description.text" : Der darzustellende Text, ggf. mit Platzhaltern
	"direction" : Sollen die Zahlenwerte an der Achse in umgekehrter Richtung wachsen? Nur in besonderen Darstellungsarten. (0 auto, 1 umgekehrt)
	"exponent" : Zehnerpotenz für die Beschriftung der Skala (-12, -9, ...12), 1000 für auto
	"font.color" : Farbe der Schrift (Zahlenwerte an den Ticks), Format siehe rgb(), (-1 automatisch, -3 Farbe der 1. Linie zur Achse)
	"font.size" : Größe der Schrift, in pt, z.B. 8 (0 auto)
	"format.option" : Gilt das Format? (0 auto mit dreireihigem Zeitformat, 1 auto Zeitformat eine Reihe, 2 auto Zeitformat zwei Reihen, 3 frei definiertes Format mit einer Reihe, 4 frei definiertes Format mit zwei Reihen)
	"labels.end" : Sollen die Beschriftungen an den beiden Enden der Achse z.B. verschoben werden, sodass sie nicht über das Ende der Achse hinausragen? (0 auto, 1 stets unverschoben, 2 verschieben falls nötig)
	"line.color" : Farbe der Achsenlinie, Format siehe rgb(), (-1 automatisch)
	"line.show" : Soll die Achsenlinie angezeigt werden? (0 auto, 1 ja, 2 nein)
	"line.width" : Liniendicke der Achsenlinien mm (0 auto)
	"max.nominal" : Nominalwert des Maximums.
	"min.nominal" : Nominalwert des Minimums.
	"orientation.logical" : Logische Orientierung, Richtung (1 x, 2 y, 3 z oder Farbachse bei Standard und Farbkarte oder Winkel bei Polar, 4 Farbachse bei 3D)
	"places.right" : Nachkommastellen: 0..14, -1 automatisch
	"position.bot" : Relative Position des unteren Endes der Achse. 0% ist ganz unten, 50% in der Mitte.
	"position.place" : Soll die Achse links oder rechts vom Koordinatensystem positioniert werden? (0 auto, 1 links, 2 rechts)
	"position.top" : Relative Position des oberen Endes der Achse. 100% ist ganz oben, 50% in der Mitte.
	"resolution" : Auflösung einer Achse, i.a. für y-Achse (0 auto, 1 gleiche Auflösung wie x-Achse)

	"small ticks.count" : Anzahl kleine Ticks, >= 0 fest; -2 auto
	"ticks.count" : Anzahl große Ticks, > 1. Nur gültig, wenn die Tickoption auf "am Achsenende mit fester Anzahl" gesetzt ist.
	"ticks.large.length" : Gesamte Länge der großen Ticks in mm (-1 auto)
	"ticks.large.width" : Linienstärke der großen Ticks (0 auto, 1 wie Achsenlinie, 2 75% Achsenlinie, 3 50% Achsenlinie, 4 25% Achsenlinie)
	"ticks.option" : Tickoption (1 Anzahl auto am Achsenende, 2 Anzahl fest am Achsenende, 3 Abstand auto, 4 Abstand fest)
	"ticks.orientation" : In welche Richtung sollen die Ticks gezeichnet werden. Nach außen in Richtung der Schrift oder nach innen zum Koordinatensystem hin? (0 auto, 1 außen, 2 innen, 3 außen und innen)
	"ticks.small.length" : Gesamte Länge der kleinen Ticks in mm (-1 auto)
	"ticks.small.width" : Linienstärke der kleinen Ticks (0 auto, 1 wie Achsenlinie, 2 75% Achsenlinie, 3 50% Achsenlinie, 4 25% Achsenlinie)
	"ticks.spacing" : Der feste Abstand zwischen den Ticks. Nur gültig, wenn die Tickoption auf festen Abstand gesetzt ist.
	"unit.visible" : Einheit Sichtbarkeit (0 nein, 1 auto=ja)
	"userticks" : Eigene Ticks; gewünscht (0 nein, 1 zusätzlich, 2 ausschließlich)
	"userticks.alignX" : Eigene Ticks; Ausrichtung Bezugspunkt Text horizontal (0 auto, 1 linksbündig, 2 zentriert, 3 rechtsbündig, 4 linksbündig erweitert, 5 zentriert erweitert, 6 rechtsbündig erweitert, 7 andere Seite). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.alignY" : Eigene Ticks; Ausrichtung Bezugspunkt Text vertikal (0 auto, 1 unten bündig, 2 zentriert, 3 oben bündig, 4 unten bündig erweitert, 5 zentriert erweitert, 6 oben bündig erweitert, 7 andere Seite). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.clip" : Eigene Ticks; Begrenzung Text. Wird auf die Beschriftung verzichtet, wenn ein außerhalb liegender Tick selbst nicht gezeichnet wird? (0 auto, 1 nein). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.exclusive" : Eigene Ticks; exklusiv. Soll ausschließlich der eigene Tick an der Pixelposition der Achse gezeichnet werden? (0 nein, ein regulärer Tick könnte gezeichnet werden; 1 ja, ein regulärer Tick wird dort nicht gezeichnet). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.grid" : Eigene Ticks; Gitterlinie sichtbar (0 auto, 1 bei Gitter am Fenster, 2 ja, 3 nein). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.Label for raw data" : Eigene Ticks Ableitung von der Anwender-definierten Eigenschaft des Kanals "Label for raw data". (0 nein, 1 ja, 2 Vorlage). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.position" : Eigene Ticks; Position. An welcher Stelle (Koordinate) soll der Tick angebracht werden. Vorgabe als Zahlenwert. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.position.type" : Eigene Ticks; was bedeutet die Position (0 auto = Koordinate in physikalischen Einheiten, 1 prozentual von 0 bis 100 entlang der Achse in Richtung steigender Achsenwerte). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.shiftx" : Eigene Ticks; Verschiebung des Bezugspunktes des Textes in x-Richtung, in mm, positiv nach rechts, negativ nach links. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.shifty" : Eigene Ticks; Verschiebung des Bezugspunktes des Textes in y-Richtung, in mm, positiv nach oben, negativ nach unten. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text.angle" : Eigene Ticks; Neigung des Textes in Grad (-90 .. +90). Wie stark ist der Text gegen die Horizontale geneigt? Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text.color" : Eigene Ticks; Textfarbe, Format siehe rgb(), -1 automatisch. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text.size" : Eigene Ticks; Größe der Schrift, in pt, z.B. 8 (0 auto). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text.style" : Eigene Ticks; Schriftstil (0 auto, 1 Standard, 2 fett, 3 kursiv, 4 fett und kursiv, 5 unterstrichen, 6 fett und unterstrichen, 7 kursiv und unterstrichen, 8 fett und kursiv und unterstrichen). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.tick" : Eigene Ticks; Art des Ticks (0 auto, 1 großer Tick, 2 kleiner Tick, 3 kein großer Tick, 4 kein kleiner Tick). Zugriff auf den aktuell selektierten eigenen Tick
	"visible" : Sichtbarkeit der Achse. Nur in besonderen Situationen, z.B. Polardiagramm oder z-Achse bei Farbpalette (0 auto, 1 sichtbar, 2 unsichtbar)
	"width" : Breite der Achse in mm (0 auto)
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Den Wertebereich der x-Achse abfragen

```
CwSelectByIndex("x-axis", 1)
xmin = CwAxisGet("min")
xmax = CwAxisGet("max")
```

Siehe auch:

CwAxisSet, CwAxisGetText

CwAxisGetText

Anwendungsbereich: Kurvenfenster

Text-Eigenschaft einer Achse abfragen

Deklaration:

```
CwAxisGetText ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"description.text" : Der darzustellende Text, ggf. mit Platzhaltern
	"format.text" : Format der Beschriftung der Ticks, z.B. bei absoluter oder relativer Zeit: <hh:mm:ss.s> Gültig bei passender Option
	"format.text2" : Format der Beschriftung der zweiten Reihe im Zeitformat, z.B. bei absoluter Zeit: <DD.MM.YYYY>
	"userticks.text" : Eigene Ticks; angezeigter Text. Zugriff auf den aktuell selektierten eigenen Tick
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Den Zusatztext der x-Achse abfragen

```
CwSelectByIndex("x-axis", 1)
txt = CwAxisGetTxt("description.text")
```

Siehe auch:

CwAxisSet, CwAxisGet

CwAxisSet

Anwendungsbereich: Kurvenfenster

Eigenschaft einer Achse setzen

Deklaration:

```
CwAxisSet ( Eigenschaft, Wert )
```

Parameter:

Eigenschaft	Welche Eigenschaft soll gesetzt werden?
	"range" : Bereich (1 auto, 2 auto mit null, 3 runden, 4 fest, 5 wie vorherige)
	"scale" : Skala (1 Standard linear, 2 dB, 3 logarithmisch, 4 absolute Zeit, 5 relative Zeit, 6 Terzen)
	"max" : Sollwert des Maximums. Nur wirksam, wenn der Bereich auch fest ist, siehe "range"
	"min" : Sollwert des Minimums. Nur wirksam, wenn der Bereich auch fest ist, siehe "range"
	"count.userticks" : Setzen der Anzahl eigene Ticks, so viele sollen zu dieser Achse vorhanden sein.
	"description.color" : Farbe des Zusatztextes, Format siehe rgb(), (-1 automatisch)
	"description.distance" : Wie weit entfernt von der Achse soll der Zusatztext platziert werden. In mm, auch < 0.
	"description.font.size" : Größe der Schrift für den Zusatztext, in pt, z.B. 8, (0 auto)
	"description.option" : Soll ein zusätzlicher Text an der Achse angezeigt werden? (0 nein, 1 fester Text, 2 Einheit, 3 [Einheit], 4 Name Einheit, 5 definierbar mit Platzhaltern)
	"description.orientation" : In welche Richtung soll der Zusatztext verlaufen (0 auto, 1 parallel zur Achse, 2 quer zur Achse)
	"description.pos" : Wo genau soll der Zusatztext angebracht werden? (0 auto, 1 zentriert, 2 bündig mit Achsenbeginn, 3 bündig mit Achsenende)
	"description.symbol" : Soll das Symbol der Linie wie in einer Legende dargestellt werden? (0 nein, 1 ja davor)
	"description.text" : Der darzustellende Text, ggf. mit Platzhaltern
	"direction" : Sollen die Zahlenwerte an der Achse in umgekehrter Richtung wachsen? Nur in besonderen Darstellungsarten. (0 auto, 1 umgekehrt)
	"exponent" : Zehnerpotenz für die Beschriftung der Skala (-12, -9, ...12), 1000 für auto
	"font.color" : Farbe der Schrift (Zahlenwerte an den Ticks), Format siehe rgb(), (-1 automatisch, -3 Farbe der 1. Linie zur Achse)
	"font.size" : Größe der Schrift, in pt, z.B. 8 (0 auto)
	"format.option" : Gilt das Format? (0 auto mit dreireihigem Zeitformat, 1 auto Zeitformat eine Reihe, 2 auto Zeitformat zwei Reihen, 3 frei definiertes Format mit einer Reihe, 4 frei definiertes Format mit zwei Reihen)
	"format.text" : Format der Beschriftung der Ticks, z.B. bei absoluter oder relativer Zeit: <hh:mm:ss.s> Gültig bei passender Option
	"format.text2" : Format der Beschriftung der zweiten Reihe im Zeitformat, z.B. bei absoluter Zeit: <DD.MM.YYYY>
	"labels.end" : Sollen die Beschriftungen an den beiden Enden der Achse z.B. verschoben werden, sodass sie nicht über das Ende der Achse hinausragen? (0 auto, 1 stets unverschoben, 2 verschieben falls nötig)
	"line.color" : Farbe der Achsenlinie, Format siehe rgb(), (-1 automatisch)
	"line.show" : Soll die Achsenlinie angezeigt werden? (0 auto, 1 ja, 2 nein)
	"line.width" : Liniendicke der Achsenlinien mm (0 auto)
	"places.right" : Nachkommastellen: 0..14, -1 automatisch
	"position.bot" : Relative Position des unteren Endes der Achse. 0% ist ganz unten, 50% in der Mitte.
	"position.place" : Soll die Achse links oder rechts vom Koordinatensystem positioniert werden? (0 auto, 1 links, 2 rechts)
	"position.top" : Relative Position des oberen Endes der Achse. 100% ist ganz oben, 50% in der Mitte.
	"resolution" : Auflösung einer Achse, i.a. für y-Achse (0 auto, 1 gleiche Auflösung wie x-Achse)
	"small ticks.count" : Anzahl kleine Ticks, >= 0 fest; -2 auto
	"ticks.count" : Anzahl große Ticks, > 1. Nur gültig, wenn die Tickoption auf "am Achsenende mit fester Anzahl" gesetzt ist.
	"ticks.large.length" : Gesamte Länge der großen Ticks in mm (-1 auto)

	"ticks.large.width" : Linienstärke der großen Ticks (0 auto, 1 wie Achsenlinie, 2 75% Achsenlinie, 3 50% Achsenlinie, 4 25% Achsenlinie)
	"ticks.option" : Tickoption (1 Anzahl auto am Achsenende, 2 Anzahl fest am Achsenende, 3 Abstand auto, 4 Abstand fest)
	"ticks.orientation" : In welche Richtung sollen die Ticks gezeichnet werden. Nach außen in Richtung der Schrift oder nach innen zum Koordinatensystem hin? (0 auto, 1 außen, 2 innen, 3 außen und innen)
	"ticks.small.length" : Gesamte Länge der kleinen Ticks in mm (-1 auto)
	"ticks.small.width" : Linienstärke der kleinen Ticks (0 auto, 1 wie Achsenlinie, 2 75% Achsenlinie, 3 50% Achsenlinie, 4 25% Achsenlinie)
	"ticks.spacing" : Der feste Abstand zwischen den Ticks. Nur gültig, wenn die Tickoption auf festen Abstand gesetzt ist.
	"unit.visible" : Einheit Sichtbarkeit (0 nein, 1 auto=ja)
	"userticks" : Eigene Ticks; gewünscht (0 nein, 1 zusätzlich, 2 ausschließlich)
	"userticks.alignX" : Eigene Ticks; Ausrichtung Bezugspunkt Text horizontal (0 auto, 1 linksbündig, 2 zentriert, 3 rechtsbündig, 4 linksbündig erweitert, 5 zentriert erweitert, 6 rechtsbündig erweitert, 7 andere Seite). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.alignY" : Eigene Ticks; Ausrichtung Bezugspunkt Text vertikal (0 auto, 1 unten bündig, 2 zentriert, 3 oben bündig, 4 unten bündig erweitert, 5 zentriert erweitert, 6 oben bündig erweitert, 7 andere Seite). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.clip" : Eigene Ticks; Begrenzung Text. Wird auf die Beschriftung verzichtet, wenn ein außerhalb liegender Tick selbst nicht gezeichnet wird? (0 auto, 1 nein). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.exclusive" : Eigene Ticks; exklusiv. Soll ausschließlich der eigene Tick an der Pixelposition der Achse gezeichnet werden? (0 nein, ein regulärer Tick könnte gezeichnet werden; 1 ja, ein regulärer Tick wird dort nicht gezeichnet). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.grid" : Eigene Ticks; Gitterlinie sichtbar (0 auto, 1 bei Gitter am Fenster, 2 ja, 3 nein). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.Label for raw data" : Eigene Ticks Ableitung von der Anwender-definierten Eigenschaft des Kanals "Label for raw data". (0 nein, 1 ja, 2 Vorlage). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.position" : Eigene Ticks; Position. An welcher Stelle (Koordinate) soll der Tick angebracht werden. Vorgabe als Zahlenwert. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.position.type" : Eigene Ticks; was bedeutet die Position (0 auto = Koordinate in physikalischen Einheiten, 1 prozentual von 0 bis 100 entlang der Achse in Richtung steigender Achsenwerte). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.shiftx" : Eigene Ticks; Verschiebung des Bezugspunktes des Textes in x-Richtung, in mm, positiv nach rechts, negativ nach links. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.shifty" : Eigene Ticks; Verschiebung des Bezugspunktes des Textes in y-Richtung, in mm, positiv nach oben, negativ nach unten. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text" : Eigene Ticks; angezeigter Text. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text.angle" : Eigene Ticks; Neigung des Textes in Grad (-90 .. +90). Wie stark ist der Text gegen die Horizontale geneigt? Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text.color" : Eigene Ticks; Textfarbe, Format siehe rgb(), -1 automatisch. Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text.size" : Eigene Ticks; Größe der Schrift, in pt, z.B. 8 (0 auto). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.text.style" : Eigene Ticks; Schriftstil (0 auto, 1 Standard, 2 fett, 3 kursiv, 4 fett und kursiv, 5 unterstrichen, 6 fett und unterstrichen, 7 kursiv und unterstrichen, 8 fett und kursiv und unterstrichen). Zugriff auf den aktuell selektierten eigenen Tick
	"userticks.tick" : Eigene Ticks; Art des Ticks (0 auto, 1 großer Tick, 2 kleiner Tick, 3 kein großer Tick, 4 kein kleiner Tick). Zugriff auf den aktuell selektierten eigenen Tick
	"visible" : Sichtbarkeit der Achse. Nur in besonderen Situationen, z.B. Polardiagramm oder z-Achse bei Farbpalette (0 auto, 1 sichtbar, 2 unsichtbar)
	"width" : Breite der Achse in mm (0 auto)
Wert	Auf welchen Wert soll diese Eigenschaft gesetzt werden?

Beschreibung:

Beispiele:

Eine Achse selektieren und parametrieren

```
CwSelectWindow("curve1")
```

```
CwSelectByIndex("y-axis", 1)
CwAxisSet("range", 4)
CwAxisSet("min", -10)
CwAxisSet("max", 10)
```

Die x-Achse selektieren und parametrieren

```
CwSelectByIndex("x-axis", 1)
CwAxisSet("scale", 4)
CwAxisSet("range", 1)
```

Eigene Ticks an der x-Achse: Zuerst den Modus wählen. Dann die Anzahl der Ticks wählen. Dann einen Tick selektieren und einstellen.

```
CwSelectByIndex("x-axis", 1)
CwAxisSet("userticks", 1 )
CwAxisSet("count.userticks", 2)
CwSelectByIndex("usertick", 1)
CwAxisSet("userticks.position", 3.0)
CwAxisSet("userticks.text", "!")
CwSelectByIndex("usertick", 2)
CwAxisSet("userticks.position", 4.0)
CwAxisSet("userticks.text", "?")
```

Siehe auch:

CwAxisGet, CwAxisGetText

CwColorGet

Anwendungsbereich: Kurvenfenster

Farben des Kurvenfensters abfragen

Deklaration:

```
CwColorGet ( Farbe, Individuell, Bildschirm ) -> Farbe
```

Parameter:

Farbe	Welche Farbe soll abgefragt werden?
	"back" : Hintergrund. Farbe = -2 für transparenten Hintergrund bei Drucker
	"border.bot" : Koordinatensystem unten rechts
	"border.top" : Koordinatensystem oben links
	"cos.back" : Koordinatensystem Hintergrund
	"grid.1" : Hauptgitter
	"grid.2" : Nebengitter
	"leg.back" : Legende Hintergrund
	"leg.bot" : Legende Rahmen unten rechts
	"leg.text" : Legende Text
	"leg.top" : Legende Rahmen oben links
	"num" : Zahlen: Vordergrund
	"num.back" : Zahlen: Hintergrund
	"text" : Allgemeiner Text
	"trigger" : Trigger-Linie, Hilfslinien
	1 .. 24 : Index der Kurve
Individuell	Farbe individuell für das selektierte Kurvenfenster oder global für alle?
	"global" : global
	"individual" : individuell
Bildschirm	Bildschirm oder Drucker
	"printer" : Drucker
	"screen" : Bildschirm
Farbe	Farbe, Format siehe rgb()

Beschreibung:

Individuelle Farben können nur abgefragt werden, wenn das Kurvenfenster auf individuelle Farben eingestellt ist.

Wenn individuelle Farben abgefragt werden, muss das betreffende Kurvenfenster selektiert sein.

Für das Abfragen einer globalen Farbe braucht kein Kurvenfenster selektiert zu sein.

Beispiele:

```
rgb = CwColorGet("back", "individual", "screen")
```

Siehe auch:

CwColorSet

CwColorSet

Anwendungsbereich: Kurvenfenster

Farben des Kurvenfensters setzen

Deklaration:

```
CwColorSet ( Farbe, Individuell, Bildschirm, Farbe )
```

Parameter:

Farbe	Welche Farbe soll gesetzt werden?
	"back" : Hintergrund. Farbe = -2 für transparenten Hintergrund bei Drucker
	"border.bot" : Koordinatensystem unten rechts
	"border.top" : Koordinatensystem oben links
	"cos.back" : Koordinatensystem Hintergrund
	"grid.1" : Hauptgitter
	"grid.2" : Nebengitter
	"leg.back" : Legende Hintergrund
	"leg.bot" : Legende Rahmen unten rechts
	"leg.text" : Legende Text
	"leg.top" : Legende Rahmen oben links
	"num" : Zahlen: Vordergrund
	"num.back" : Zahlen: Hintergrund
	"text" : Allgemeiner Text
	"trigger" : Trigger-Linie, Hilfslinien
	1 .. 24 : Index der Kurve
Individuell	Farbe individuell für das selektierte Kurvenfenster oder global für alle?
	"global" : global
	"individual" : individuell
Bildschirm	Bildschirm oder Drucker
	"printer" : Drucker
	"screen" : Bildschirm
	"screen+printer" : Bildschirm und Drucker
Farbe	Auf welchen Wert soll diese Farbe gesetzt werden? Format siehe rgb()

Beschreibung:

Individuelle Farben werden erst wirksam, wenn das Kurvenfenster auf individuelle Farben eingestellt ist. Das kann z.B. mit CwDisplaySet() und "colors.screen.indiv" oder "colors.printer.indiv" erreicht werden.

Wenn individuelle Farben gesetzt werden, muss das betreffende Kurvenfenster selektiert sein.

Globale Farben beeinflussen alle Kurvenfenster. Für das Setzen braucht kein Kurvenfenster selektiert zu sein.

Beispiele:

Individuelle Farben setzen

```
CwDisplaySet("colors.screen.indiv", 1)
CwColorSet("back", "individual", "screen", rgb(255,0,0))
```

Globale Farben setzen zum Drucken

```
CwColorSet(1, "global", "printer", rgb(0,0,0))
CwColorSet(2, "global", "printer", rgb(0,0,0))
```

Siehe auch:

CwColorGet

CwCosysGet

Anwendungsbereich: Kurvenfenster

Eigenschaft eines Koordinatensystems abfragen

Deklaration:

```
CwCosysGet ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"count.axis" : Anzahl Achsen, die in diesem Koordinatensystem enthalten sind.
	"count.data" : Anzahl Datenelemente, die in diesem Koordinatensystem enthalten sind.
	"count.line" : Anzahl Linien, die in diesem Koordinatensystem enthalten sind.
	"height.relative" : Die konfigurierte relative Höhe des einzelnen Koordinatensystems bezogen auf die Gesamthöhe aller Koordinatensysteme. Zwischen 0 und 1. Bei automatischer Höhe -1.
	"pos.dx" : Breite des Koordinatensystems auf dem Bildschirm, in Pixeln
	"pos.dy" : Höhe des Koordinatensystems auf dem Bildschirm, in Pixeln
	"pos.x" : Abstand des Koordinatensystems vom linken Rand der Client Area des Kurvenfensters auf dem Bildschirm, in Pixeln
	"pos.y" : Abstand des Koordinatensystems vom oberen Rand der Client Area des Kurvenfensters auf dem Bildschirm, in Pixeln
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Anzahl der Achsen im Koodinatensystem abfragen

```
CwSelectByIndex("cosys", 1)
count = CwCosysGet("count.axis")
```

Siehe auch:

CwCosysSet

CwCosysSet

Anwendungsbereich: Kurvenfenster

Eigenschaft eines Koordinatensystems setzen

Deklaration:

```
CwCosysSet ( Eigenschaft, Wert )
```

Parameter:

Eigenschaft	Welche Eigenschaft soll gesetzt werden?
	"height.relative" : Die relative Höhe des einzelnen Koordinatensystems bezogen auf die Gesamthöhe aller Koordinatensysteme. >= 0.001 und <= 1.0. Nicht alle Vorgaben sind aufgrund von Einschränkungen der Darstellungsmöglichkeiten realisierbar. I. Allg. nur das, was auch mit der Maus durch Ziehen eingestellt werden kann. Wenn kein Koordinatensystem eine individuelle Höhe hat, werden alle Höhen automatisch bestimmt.
Wert	Auf welchen Wert soll diese Eigenschaft gesetzt werden?

Beschreibung:

Beispiele:

```
CwSelectByIndex("cosys", 1)
CwCosysSet("height.relative", 0.1)
```

Siehe auch:

CwCosysGet

CwDataGet

Anwendungsbereich: Kurvenfenster

Eigenschaft eines Datenelementes abfragen

Deklaration:

```
CwDataGet ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"component" : Komponente des Kanals: 0 gesamter Kanal, 1 erste Komponte (.Y, .R, .B), 2 zweite Komponente (.X, .I, .P)
	"event.add" : Events jedes N-te, >= 1 (bei Selektionsmodus N)
	"event.align" : Events zeitliche Anordnung (0 auto, 1 jedes zeitrichtig, 2 Triggerzeit erstes, 3 Triggerzeit letztes 4 Triggerzeit erstes dargestelltes, 5 Triggerzeit letztes dargestelltes, 6 Triggerzeitunterschied erstes, 7 Triggerzeitunterschied letztes, 8 Triggerzeitunterschied erstes dargestelltes, 9 Triggerzeitunterschied letztes dargestelltes)
	"event.count" : Events Anzahl (bei Selektionsmodus N oder N letzte)
	"event.index" : Events Startindex, >= 1 (bei Selektionsmodus eins, N, N-te von hinten)
	"event.select" : Events Selektionsmodus (0 auto, 1 alle, 2 letztes, 3 N letzte, 4 eins, 5 N, 6 N-te von hinten). Beim Setzen des Selektionsmodus werden die anderen Eigenschaften zur Eventauswahl zurück gesetzt. Der Selektionsmodus muss gesetzt werden, bevor weitere Eigenschaften die Eventauswahl näher beschreiben.
	"function" : Bedeutung des Datenelementes (0 Standard, 1 Farbinformation zum Vorgänger (Farbpalette), 2 Boxplot Quartil, 3 Boxplot Whisker)
	"instrument.fill" : Instrument: Füllung (0 auto, 1 von unten, 2 von oben, 3 von der Mitte, 4 von y=0, 5 Min/Max Spanne prozentual)
	"instrument.color.base" : Instrument: Grundfarbe Balken, Format siehe rgb(), -1 auto
	"instrument.color.scheme" : Instrument: Farbgestaltung (0 auto, 1 einfarbig, 2 drei Farben, 3 drei Farben gleichzeitig, 4 zwei Farben, Übertretung halten)
	"instrument.color.lower" : Instrument: Farbe ab unterer Grenze, Format siehe rgb()
	"instrument.color.number" : Instrument: Farbe des Zahlenwertes, Format siehe rgb(), -1 auto
	"instrument.color.text" : Instrument: Farbe des Textes, Format siehe rgb(), -1 auto
	"instrument.color.upper" : Instrument: Farbe ab oberer Grenze, Format siehe rgb()
	"instrument.limit.lower" : Instrument: Untere Grenze, Minimum
	"instrument.limit.upper" : Instrument: Obere Grenze, Maximum
	"instrument.slavepointer.color" : Instrument: Schleppzeiger Farbe, Format siehe rgb()
	"instrument.slavepointer.show" : Instrument: Schleppzeiger anzeigen (0 nein, 1 ja)
	"instrument.title.auto" : Instrument: Titel automatisch (0 nein 1 ja)
	"instrument.unit.show" : Instrument: Einheit anzeigen (0 nein, 1 ja)
	"instrument.width" : Instrument: Breite in Prozent (0 bis 100)
	"instrument.100.color" : Instrument: 100% Werte, Linienfarbe, Format siehe rgb()
	"instrument.100.line" : Instrument: 100% Werte, Linie (0 nein, 1 ja)
	"instrument.100.+" : Instrument: 100% Werte, physikalischer Wert bei +100%
	"instrument.100.-" : Instrument: 100% Werte, physikalischer Wert bei -100%
	"numerical.format" : Zahlenwertdarstellung: Format Option (1 Festkomma, 2 Gleitkomma, 3 hex ein Byte, 4 hex zwei Bytes, 5 hex vier Bytes, 6 absolute Zeit auto, 7 abs. Zeit mit frei definiertem Format, 8 relative Zeit auto, 9 rel. Zeit mit frei definiertem Format). Bei Instrumenten 1 und 2.
	"numerical.places.left" : Zahlenwertdarstellung: Vorkommastellen: 1 bis 15
	"numerical.places.right" : Zahlenwertdarstellung: Nachkommastellen: 0 bis 15. Auch bei Instrumenten
	"numerical.prefix" : Zahlenwertdarstellung: Vorsilbe Einheit (100 auto, -12, -9, -6, -3, 0, 3, 6, 9). Auch bei Instrumenten. auto nicht immer verfügbar
	"period.add" : Perioden jedes N-te, >= 1 (bei Selektionsmodus N)

	"period.align" : Perioden zeitliche Anordnung (0 auto, 1 x0 auf null, 2 x0 der ersten Periode, 3 x0 der letzten Periode, 4 jede Periode behält individuelles x0)
	"period.count" : Perioden Anzahl (bei Selektionsmodus N oder N letzte)
	"period.index" : Perioden Startindex, >= 1 (bei Selektionsmodus eins, N, N-te von hinten)
	"period.length" : Länge einer Periode, >= 1. Nur wirksam, wenn ein gültiger Selektionsmodus zum Vergleich von Perioden gewählt ist.
	"period.select" : Perioden Selektionsmodus (7 kein Periodenvergleich, 0 auto, 1 alle, 2 letztes, 3 N letzte, 4 eins, 5 N, 6 N-te von hinten). Beim Setzen des Selektionsmodus werden Startindex, Anzahl und N-te Periode zurück gesetzt. Der Selektionsmodus muss also gesetzt werden, bevor weitere Eigenschaften die Periodenauswahl näher beschreiben.
	"segment.add" : Segmente jedes N-te, >= 1 (bei Selektionsmodus N)
	"segment.align" : Segmente zeitliche Anordnung (0 auto, 1 x-Koordinate bleibt erhalten, 2 zur x-Koordinate wird z addiert)
	"segment.count" : Segmente Anzahl (bei Selektionsmodus N oder N letzte)
	"segment.index" : Segmente Startindex, >= 1 (bei Selektionsmodus eins, N, N-te von hinten)
	"segment.select" : Segmente Selektionsmodus (0 auto, 1 alle, 2 letztes, 3 N letzte, 4 eins, 5 N, 6 N-te von hinten). Beim Setzen des Selektionsmodus werden die Eigenschaften Startindex, Anzahl und N-tes Segment zurück gesetzt. Der Selektionsmodus muss also gesetzt werden, bevor weitere Eigenschaften die Segmentauswahl näher beschreiben.
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Die dargestellte Komponente abfragen

```
CwSelectByIndex("data", 1)
cmp = CwDataGet("component")
```

Siehe auch:

CwDataSet, CwDataGetText

CwDataGetText

Anwendungsbereich: Kurvenfenster

Text-Eigenschaft eines Datenelementes abfragen

Deklaration:

```
CwDataGetText ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"channelname" : Name des Kanals, z.B. "channel" oder bei Kanal in Gruppe "group:channel"
	"instrument.title" : Instrument: Titel
	"numerical.format.text" : Zahlenwertdarstellung: Formattext bei passender Format Option, z.B. <hh:mm:ss.s> bei absoluter Zeit
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Den Namen des dargestellten Kanals abfragen

```
CwSelectByIndex("data", 1)
name = CwDataGetText("channelname")
```

Siehe auch:

CwDataSet, CwDataGet

CwDataSet

Anwendungsbereich: Kurvenfenster

Eigenschaft eines Datenelementes setzen

Deklaration:

```
CwDataSet ( Eigenschaft, Wert )
```

Parameter:

Eigenschaft	Welche Eigenschaft soll gesetzt werden?
	"component" : Komponente des Kanals: 0 gesamter Kanal, 1 erste Komponte (.Y, .R, .B), 2 zweite Komponente (.X, .I, .P)
	"event.add" : Events jedes N-te, >= 1 (bei Selektionsmodus N)
	"event.align" : Events zeitliche Anordnung (0 auto, 1 jedes zeitrichtig, 2 Triggerzeit erstes, 3 Triggerzeit letztes 4 Triggerzeit erstes dargestelltes, 5 Triggerzeit letztes dargestelltes, 6 Triggerzeitunterschied erstes, 7 Triggerzeitunterschied letztes, 8 Triggerzeitunterschied erstes dargestelltes, 9 Triggerzeitunterschied letztes dargestelltes)
	"event.count" : Events Anzahl (bei Selektionsmodus N oder N letzte)
	"event.index" : Events Startindex, >= 1 (bei Selektionsmodus eins, N, N-te von hinten)
	"event.select" : Events Selektionsmodus (0 auto, 1 alle, 2 letztes, 3 N letzte, 4 eins, 5 N, 6 N-te von hinten). Beim Setzen des Selektionsmodus werden Startindex, Anzahl und N-tes Event zurück gesetzt. Der Selektionsmodus muss also gesetzt werden, bevor weitere Eigenschaften die Eventauswahl näher beschreiben.
	"function" : Bedeutung des Datenelementes (0 Standard, 1 Farbinformation zum Vorgänger (Farbpalette), 2 Boxplot Quartil, 3 Boxplot Whisker, 4 Bubbles)
	"instrument.fill" : Instrument: Füllung (0 auto, 1 von unten, 2 von oben, 3 von der Mitte, 4 von y=0, 5 Min/Max Spanne prozentual)
	"instrument.color.base" : Instrument: Grundfarbe Balken, Format siehe rgb(), -1 auto
	"instrument.color.scheme" : Instrument: Farbgestaltung (0 auto, 1 einfarbig, 2 drei Farben, 3 drei Farben gleichzeitig, 4 zwei Farben, Übertretung halten)
	"instrument.color.lower" : Instrument: Farbe ab unterer Grenze, Format siehe rgb()
	"instrument.color.number" : Instrument: Farbe des Zahlenwertes, Format siehe rgb(), -1 auto
	"instrument.color.text" : Instrument: Farbe des Textes, Format siehe rgb(), -1 auto
	"instrument.color.upper" : Instrument: Farbe ab oberer Grenze, Format siehe rgb()
	"instrument.limit.lower" : Instrument: Untere Grenze, Minimum
	"instrument.limit.upper" : Instrument: Obere Grenze, Maximum
	"instrument.slavepointer.color" : Instrument: Schleppzeiger Farbe, Format siehe rgb()
	"instrument.slavepointer.show" : Instrument: Schleppzeiger anzeigen (0 nein, 1 ja)
	"instrument.title" : Instrument: Titel
	"instrument.title.auto" : Instrument: Titel automatisch (0 nein 1 ja)
	"instrument.unit.show" : Instrument: Einheit anzeigen (0 nein, 1 ja)
	"instrument.width" : Instrument: Breite in Prozent (0 bis 100)
	"instrument.100.color" : Instrument: 100% Werte, Linienfarbe, Format siehe rgb()
	"instrument.100.line" : Instrument: 100% Werte, Linie (0 nein, 1 ja)
	"instrument.100.+" : Instrument: 100% Werte, physikalischer Wert bei +100%
	"instrument.100.-" : Instrument: 100% Werte, physikalischer Wert bei -100%
	"numerical.calc.type" : Zahlenwertdarstellung: Berechnung (1 Max, 2 Min, 3 Mittel). Auch bei Instrumenten
	"numerical.calc.samples" : Zahlenwertdarstellung: Berechnung über diese Anzahl von Werten. Auch bei Instrumenten
	"numerical.format" : Zahlenwertdarstellung: Format Option (1 Festkomma, 2 Gleitkomma, 3 hex ein Byte, 4 hex zwei Bytes, 5 hex vier Bytes, 6 absolute Zeit auto, 7 abs. Zeit mit frei definiertem Format, 8 relative Zeit auto, 9 rel. Zeit mit frei definiertem Format). Bei Instrumenten 1 und 2.
	"numerical.format.text" : Zahlenwertdarstellung: Formattext bei passender Format Option, z.B. <hh:mm:ss.s> bei absoluter Zeit
	"numerical.places.left" : Zahlenwertdarstellung: Vorkommastellen: 1 bis 15

	"numerical.places.right" : Zahlenwertdarstellung: Nachkommastellen: 0 bis 15. Auch bei Instrumenten
	"numerical.prefix" : Zahlenwertdarstellung: Vorsilbe Einheit (100 auto, -12, -9, -6, -3, 0, 3, 6, 9). Auch bei Instrumenten. auto nicht immer verfügbar
	"period.add" : Perioden jedes N-te, >= 1 (bei Selektionsmodus N)
	"period.align" : Perioden zeitliche Anordnung (0 auto, 1 x0 auf null, 2 x0 der ersten Periode, 3 x0 der letzten Periode, 4 jede Periode behält individuelles x0)
	"period.count" : Perioden Anzahl (bei Selektionsmodus N oder N letzte)
	"period.index" : Perioden Startindex, >= 1 (bei Selektionsmodus eins, N, N-te von hinten)
	"period.length" : Länge einer Periode, >= 1. Nur wirksam, wenn ein gültiger Selektionsmodus zum Vergleich von Perioden gewählt ist.
	"period.select" : Perioden Selektionsmodus (7 kein Periodenvergleich, 0 auto, 1 alle, 2 letztes, 3 N letzte, 4 eins, 5 N, 6 N-te von hinten). Beim Setzen des Selektionsmodus werden Startindex, Anzahl und N-te Periode zurück gesetzt. Der Selektionsmodus muss also gesetzt werden, bevor weitere Eigenschaften die Periodenauswahl näher beschreiben.
	"segment.add" : Segmente jedes N-te, >= 1 (bei Selektionsmodus N)
	"segment.align" : Segmente zeitliche Anordnung (0 auto, 1 x-Koordinate bleibt erhalten, 2 zur x-Koordinate wird z addiert)
	"segment.count" : Segmente Anzahl (bei Selektionsmodus N oder N letzte)
	"segment.index" : Segmente Startindex, >= 1 (bei Selektionsmodus eins, N, N-te von hinten)
	"segment.select" : Segmente Selektionsmodus (0 auto, 1 alle, 2 letztes, 3 N letzte, 4 eins, 5 N, 6 N-te von hinten). Beim Setzen des Selektionsmodus werden die Eigenschaften Startindex, Anzahl und N-tes Segment zurück gesetzt. Der Selektionsmodus muss also gesetzt werden, bevor weitere Eigenschaften die Segmentauswahl näher beschreiben.
Wert	Auf welchen Wert soll diese Eigenschaft gesetzt werden?

Beschreibung:

Beispiele:

Die Komponente des angezeigten Kanals ändern

```
CwSelectByIndex("data", 1)
CwDataSet("component", 1)
```

Siehe auch:

CwDataGet, CwDataGetText

CwDeleteElement

Anwendungsbereich: Kurvenfenster

Löscht das selektierte Element.

Deklaration:

```
CwDeleteElement ( Elementsorte )
```

Parameter:

Elementsorte	Welche Sorte von Element soll gelöscht werden?
	"axis" : y-Achse
	"cosys" : Koordinatensystem
	"line" : Linie
	"marker" : Marker

Beschreibung:

Wenn andere Elemente enthalten sind, werden diese ebenfalls gelöscht. So werden z.B. beim Löschen einer Achse alle in ihr enthaltenen Linien und die wiederum darin enthaltenen Datenelemente gelöscht.

Die Kanäle (bzw. Variablen oder Daten) werden nicht gelöscht.

Einige Elemente wie z.B. ein letztes Koordinatensystem oder auch die x-Achse lassen sich nicht löschen.

Beispiele:

Eine Achse selektieren und löschen

```
CwSelectByIndex("y-axis", 2)
CwDeleteElement("axis")
```

Siehe auch:

CwSelectByIndex, CwNewElement

CwDisplayGet

Anwendungsbereich: Kurvenfenster

Allgemeine Eigenschaft am Kurvenfenster abfragen

Deklaration:

```
CwDisplayGet ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"displaymode" : Darstellungsart: 1 Standard, 2 Übereinander, 3 Wasserfall, 4 Farbkarte, 5 Letzter Wert als Zahl, 6 Balkeninstrument, 7 Tabelle, 8 3D, 9 Polardiagramm
	"child" : Ist das Kurvenfenster ein eingebettetes Fenster? (0 nein, frei fliegend, 1 ja, eingebettet im Dialog oder Panel)
	"color palette" : Farbpalette bei Standarddarstellung (0 nein, 1 ja)
	"colors.printer.indiv" : Individuelle Farben für dieses Fenster für den Drucker (0 nein, globale Farben nutzen, 1 ja)
	"colors.printer.pattern" : Kurven in Struktur (0 nein, 1 ja). Nur gültig bei individuellen Farben für dieses Fenster für den Drucker und wenn Linienstruktur auf auto gestellt
	"colors.screen.indiv" : Individuelle Farben für dieses Fenster für den Bildschirm (0 nein, globale Farben nutzen, 1 ja)
	"colors.screen.pattern" : Kurven in Struktur (0 nein, 1 ja). Nur gültig bei individuellen Farben für dieses Fenster für den Bildschirm und wenn Linienstruktur auf auto gestellt
	"cosys.count" : Anzahl der Koordinatensysteme
	"cosys.max" : Koordinatensystem maximiert (0 nein, 1 ja)
	"data.count" : Gesamte Anzahl der Datenelemente
	"db.reference" : Bezugswert für dB-Berechnungen
	"grid" : Gitter (0 nein, 1 ja)
	"header.coordinate.system" : Kopf- oder Fußzeile, Nummer des Koordinatensystems ab 1 (0 auto). Zugriff auf die aktuell selektierte Kopfzeile
	"header.count" : Anzahl der Texte, die als Kopf- oder Fußzeile oder Überschrift angezeigt werden sollen
	"header.position" : Kopf- oder Fußzeile, Überschrift; Lage (0 auto, 1 oben links, 2 oben Mitte, 3 oben rechts, 4 unten links, 5 unten Mitte, 6 unten rechts, 7 links Mitte, 8 Mitte, 9 rechts Mitte, 10 oben links Koordinatensystem, 11 oben Mitte Koordinatensystem, 12 oben rechts Koordinatensystem, 13 unten links Koordinatensystem, 14 unten Mitte Koordinatensystem, 15 unten rechts Koordinatensystem, 16 links Mitte Koordinatensystem, 17 Mitte Koordinatensystem, 18 rechts Mitte Koordinatensystem). Zugriff auf die aktuell selektierte Kopfzeile
	"header.shiftx" : Kopf- oder Fußzeile, Überschrift; Verschiebung des Bezugspunktes des Textes in x-Richtung, in mm, positiv nach rechts, negativ nach links. Zugriff auf die aktuell selektierte Kopfzeile
	"header.shifty" : Kopf- oder Fußzeile, Überschrift; Verschiebung des Bezugspunktes des Textes in y-Richtung, in mm, positiv nach oben, negativ nach unten. Zugriff auf die aktuell selektierte Kopfzeile
	"header.text.angle" : Kopf- oder Fußzeile, Überschrift; Neigung des Textes in Grad (-90 .. +90). Wie stark ist der Text gegen die Horizontale geneigt? Zugriff auf die aktuell selektierte Kopfzeile
	"header.text.color" : Kopf- oder Fußzeile, Überschrift; Textfarbe, Format siehe rgb(), -1 automatisch. Zugriff auf die aktuell selektierte Kopfzeile
	"header.text.size" : Kopf- oder Fußzeile, Überschrift; Größe der Schrift, in pt, z.B. 8 (0 auto). Zugriff auf die aktuell selektierte Kopfzeile
	"header.text.style" : Kopf- oder Fußzeile, Überschrift; Schriftstil (0 auto, 1 Standard, 2 fett, 3 kursiv, 4 fett und kursiv, 5 unterstrichen, 6 fett und unterstrichen, 7 kursiv und unterstrichen, 8 fett und kursiv und unterstrichen). Zugriff auf die aktuell selektierte Kopfzeile
	"instrument.numval.show" : Instrument: Numerischen Wert anzeigen (0 nein, 1 linksbündig, 2 zentriert, 3 rechtsbündig)
	"instrument.title.show" : Instrument: Titel anzeigen (0 nein, 1 linksbündig, 2 zentriert, 3 rechtsbündig)
	"instrument.axis.show" : Instrument: Achse anzeigen (0 nein, 1 ja)
	"instrument.tooltip" : Instrument: Tooltip anzeigen (0 nein, 1 Name)
	"instrument.slavepointer" : Instrument: Schleppzeiger (1 auto, 2 Linie)
	"instrument.frame" : Instrument: Rahmen (1 auto, 2 gemeinsam)

	"labels" : Beschriftung (0 nein, 1 ja)
	"lastvalue.columns" : Letzter Wert als Zahl: Spalten (0 nein, 1 ja)
	"lastvalue.font.size" : Letzter Wert als Zahl: Schriftgröße, in pt, z.B. 8 (0 auto)
	"lastvalue.names" : Letzter Wert als Zahl: Namen (0 nein, 1 ja, 2 Kommentar, 3 Name und Kommentar)
	"lastvalue.right" : Letzter Wert als Zahl: rechtsbündig (0 nein, 1 ja)
	"lastvalue.=" : Letzter Wert als Zahl: Gleichheitszeichen (0 nein, 1 ja)
	"legend.border" : Legende Rand (0 nein, 1 ja)
	"legend.content" : Legende Textinhalt (0 Kanalname, 1 Kanalname (ohne Gruppenname), 2 Kommentar des Kanals, 3 Kanalname und Kommentar, 4 Kanalname ohne Gruppenname und Kommentar, 5 Kanalname ohne Messung, 6 Kanalname mit Nummer der selekt. Messung)
	"legend.curvecol" : Legende Text in Kurvenfarbe (0 nein, 1 ja)
	"legend.display" : Legende Anwesenheit (0 auto, 1 immer, 2 nie, 3 bei mehr als 1 Kurve)
	"legend.distx" : Legende Abstand zur Kante horizontal [mm] bei beweglicher Legende
	"legend.disty" : Legende Abstand zur Kante vertikal [mm] bei beweglicher Legende
	"legend.font.size" : Legende Schriftgröße, in pt, z.B. 8 (0 auto)
	"legend.font.style" : Legende Schriftstil (0 auto, 1 Standard, 2 fett, 3 kursiv, 4 fett und kursiv, 5 unterstrichen, 6 fett und unterstrichen, 7 kursiv und unterstrichen, 8 fett und kursiv und unterstrichen)
	"legend.lines" : Legende Linienprobe (0 nein, 1 ja)
	"legend.location" : Legende Lage (0 oben, 1 oben über jedem Ko'system, 2 links, 3 links neben jedem Ko'system, 4 relativ oben links, 5 relativ oben rechts, 6 relativ unten links, 7 relativ unten rechts, 8 relativ rechts, 9 relativ links, 10 relativ oben, 11 relativ unten, 12 relativ Mitte, 13 relativ oben links mehrfach, 14 relativ oben rechts mehrfach, 15 relativ unten links mehrfach, 16 relativ unten rechts mehrfach, 17 relativ rechts mehrfach, 18 relativ links mehrfach, 19 relativ oben mehrfach, 20 relativ unten mehrfach, 21 relativ Mitte mehrfach)
	"legend.numerical.display" : Legende mit Zahlenwert (0 nein, 1 letzter Wert als Zahl)
	"legend.numerical.maxdigits" : Legende Zahlenwert max. Anzahl Ziffern (0..15)
	"legend.numerical.sep" : Legende Zahlenwert mit Trennzeichen (0 nein, 1 :, 2 =)
	"legend.numerical.unit" : Legende Zahlenwert mit Einheit (0 nein, 1 ja)
	"legend.rowcol" : Legende Zeilen/Spalten (0 Zeilen, Spalten automatisch, 1 Feste Zeilenanzahl, 2 Feste Spaltenanzahl)
	"legend.rows" : Legende Anzahl der Zeilen bzw. Spalten bei der Option Feste Zeilenanzahl bzw. Feste Spaltenanzahl
	"legend.space.bottom" : Legende zusätzlicher Platz am Rand unten [mm]
	"legend.space.left" : Legende zusätzlicher Platz am Rand links [mm]
	"legend.space.right" : Legende zusätzlicher Platz am Rand rechts [mm]
	"legend.space.top" : Legende zusätzlicher Platz am Rand oben [mm]
	"legend.transparent" : Legende transparent (0 nein, 1 ja)
	"line.count" : Gesamte Anzahl der Linien
	"link.color" : Link: Farbe der Markierungslinie, Format siehe rgb(), -1 für automatisch
	"link.coordinate" : Link: Steuerung über andere Koordinate (0 auto, 1 x und y vertauschen, 2 original beibehalten)
	"link.edge.mouse" : Link: Anpassen der Achsen bei Mausbewegung am Fensterrand (0 auto, 1 nein, 2 Vergrößern, 3 Verschieben)
	"link.edge.position" : Link: Anpassen der Achsen, wenn die Markierung beim Folgen an den Rand kommt (0 auto, 1 nein, 2 Vergrößern, 3 Verschieben)
	"link.follow" : Link: Dieses Fenster folgt (0 auto, 1 Achse folgt, 2 Linie folgt)
	"link.influence" : Link: Was wird vom Link beeinflusst? (0 auto, 1 x-Achse, 2 Parameter einer XY-Darstellung, 3 x-, y-Achsen (Farbkarte), 4 y-Achse, 5 Schnitt)
	"link.linestyle" : Link: Linienart der Markierungslinie (0 auto, 1 durchgezogen, 2 gepunktet, 3 gestrichelt, 4..13 andere Kombinationen)
	"link.scale" : Link: Bei Änderung der Skalierung (0 auto, 1 Linie folgt, 2 Linie bleibt an Bildschirmposition)
	"link.shape" : Link: Grafische Form der Markierung (0 auto, 1 Linie)
	"link.t" : An der Link-Position der Parameter t der XY-Darstellung bei passender Verlinkung des Kurvenfensters

	"link.width" : Link: Dicke der Markierung der Linie in mm (0 auto)
	"link.x" : An der Link-Position die x-Koordinate bei passender Verlinkung des Kurvenfensters
	"link.y" : An der Link-Position die y-Koordinate bei passender Verlinkung des Kurvenfensters
	"map.backgnd" : Landkarte: Hintergrund (0 auto, 1 Landkarte, 2 Hintergrundbild, 3 Landkarte aus Internet)
	"map.draw" : Landkarte: Hintergrundbild zeichnen (0 nur auf Bildschirm, 1 auch auf Drucker)
	"map.filled" : Landkarte: Welchen Teil des Fensters soll das Bild überdecken? (0 Nur Koordinatensytem, 1 gesamtes Fenster)
	"map.fit" : Landkarte: Hintergrundbild Streckung (0 auto, 1 horizontal anpassen, 2 vertikal anpassen, 3 Größe beibehalten)
	"map.lat1" : Landkarte: Breitengrad des 1. Punktes in Grad
	"map.lat2" : Landkarte: Breitengrad des 2. Punktes in Grad
	"map.lon1" : Landkarte: Längengrad des 1. Punktes in Grad
	"map.lon2" : Landkarte: Längengrad des 2. Punktes in Grad
	"map.scale.adjust" : Landkarte: Maßstab anpassen (0 nein, 1 x, y-Achse passend zu Internetkarten skalieren)
	"map.x1" : Landkarte: Relative x-Koordinate des 1. Punktes, 0.0 für ganz links, 1.0 für ganz rechts im Bild.
	"map.x2" : Landkarte: Relative x-Koordinate des 2. Punktes, 0.0 für ganz links, 1.0 für ganz rechts im Bild.
	"map.y1" : Landkarte: Relative y-Koordinate des 1. Punktes, 0.0 für ganz unten, 1.0 für ganz oben im Bild.
	"map.y2" : Landkarte: Relative y-Koordinate des 2. Punktes, 0.0 für ganz unten, 1.0 für ganz oben im Bild.
	"marker.count" : Anzahl der Marker
	"measure.free" : Bei offenem Messwertfenster den Messcursor frei bewegen. (0 nein, 1 ja)
	"measure.exist" : Existiert das Messwertfenster mit Messcursoren (2 ja, 1 ja aber versteckt, 0 nein)?
	"measure.p.left" : Bei offenem Messwertfenster der Parameter des Cursors (bei XY-Darstellung) zur linken Maustaste
	"measure.p.right" : Bei offenem Messwertfenster der Parameter des Cursors (bei XY-Darstellung) zur rechten Maustaste
	"measure.x.left" : Bei offenem Messwertfenster die x-Position des Cursors zur linken Maustaste
	"measure.x.max" : Bei offenem Messwertfenster die x-Position des aktuell rechten Cursors (größeres x, größerer Parameter).
	"measure.x.min" : Bei offenem Messwertfenster die x-Position des aktuell linken Cursors (kleineres x, kleinerer Parameter).
	"measure.x.right" : Bei offenem Messwertfenster die x-Position des Cursors zur rechten Maustaste
	"measure.y.left" : Bei offenem Messwertfenster der y-Wert des Cursors zur linken Maustaste
	"measure.y.right" : Bei offenem Messwertfenster der y-Wert des Cursors zur rechten Maustaste
	"measure.z.left" : Bei offenem Messwertfenster der z-Wert des Cursors zur linken Maustaste
	"measure.z.right" : Bei offenem Messwertfenster der z-Wert des Cursors zur rechten Maustaste
	"number.trimm" : Abgeschnittene Zahlen (0 nein, 1 ja)
	"opt.on.delete" : Optimieren beim Löschen von Datensätzen (0 nein, 1 ja)
	"overview.exist" : Existiert das Übersichtsfenster (1 ja, 0 nein)?
	"scroll" : Rollmodus (0 nein, 1 rollen, 2 wachsen, 3 Oszilloskop, 4 füllen)
	"showtrigger" : Triggerlinie zeigen (0 nein, 1 ja)
	"splitmode.active" : Aktive Ansicht im Splitmodus (1 bis Anzahl der Ansichten). 0, falls kein Splitmodus. Z.B. Änderungen des Wertebereichs der x-Achse wirken auf der aktiven Ansicht.
	"splitmode.count" : Anzahl der Ansichten im Splitmodus. 1 für eine einzige Ansicht ohne Splitmodus
	"splitmode.width" : Splitmodus Breite der aktiven Ansicht in Prozent (0 bis 100) der Gesamtbreite
	"suppress.lines.opt" : Optimierung bei ausgeblendeten Linien (0 nein: Achsen und Koordinatensysteme bleiben sichtbar; 1 ja: Achsen und Koordinatensysteme werden möglichst verkleinert oder gar unsichtbar, wenn alle zugehörigen Linien ausgeblendet sind.)
	"win.client.dx" : Breite der Client Area des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"win.client.dy" : Höhe der Client Area des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"win.client.x" : Abstand der linken Kante der Client Area des Kurvenfensters vom linken Fensterrand für frei fliegende Kurvenfenster, in Pixeln

	"win.client.y" : Abstand der oberen Kante der Client Area des Kurvenfensters vom oberen Fensterrand für frei fliegende Kurvenfenster, in Pixeln
	"win.dx" : Breite des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"win.dy" : Höhe des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"win.x" : Position der linken Kante des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"win.y" : Position der oberen Kante des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"y-axis.count" : Gesamte Anzahl der y-Achsen
	"3D.angle1" : 3D: Erste Winkelangabe zur Definition der Perspektive, in Grad (Länge oder Winkel z-Achse)
	"3D.angle2" : 3D: Zweite Winkelangabe zur Definition der Perspektive, in Grad (Breite)
	"3D.angle3" : 3D: Dritte Winkelangabe zur Definition der Perspektive, in Grad (Rotation)
	"3D.color.fillsymbol" : Farbkarte: Farbe Symbolfüllung, Format siehe rgb()
	"3D.color.fix1" : Farbpalette: 1. feste Farbe, Format siehe rgb()
	"3D.color.fix2" : Farbpalette: 2. feste Farbe, Format siehe rgb()
	"3D.color.fix3" : Farbpalette: 3. feste Farbe, Format siehe rgb()
	"3D.color.fix4" : Farbpalette: 4. feste Farbe, Format siehe rgb()
	"3D.color.isoback" : Farbkarte: Farbe ISO-Linien Hintergrund, Format siehe rgb(), -1 für automatisch, -2 für transparent
	"3D.color.isoborder" : Farbkarte: Farbe ISO-Linien Rand, Format siehe rgb(), -1 für automatisch, -2 für transparent
	"3D.color.isolines" : Farbkarte und 3D: Farbe ISO-Linien, Format siehe rgb()
	"3D.color.isotext" : Farbkarte: Farbe ISO-Linientext, Format siehe rgb()
	"3D.color.symbolborder" : Farbkarte: FarbeSymbolrand, Format siehe rgb(), -1 für automatisch, -2 für transparent
	"3D.color bar.width" : Farbpalette: Farbprobe Breite in mm, 0 für automatisch
	"3D.color legend" : 3D: Farblegende Anwesenheit (0 auto, 1 nein, 2 Farblegende an vertikaler y-Achse, 3 separate Farblegende rechts, 4 separate Farblegende links)
	"3D.color pattern" : Farbpalette: Farbschema (0 auto, 1 zwei feste Farben, 2 drei feste Farben, 3 vier feste Farben, 4 spektral, 5 schwarz weiss kompatibel, 6 schwarz weiss, 7 black blue green yellow red, 8 blue cyan green yellow red, 9 blue green yellow red, 10 black blue pink red yellow white, 11 green cyan blue pink red, 12 black blue cyan green yellow red, 13 black blue red yellow white, 14 black blue cyan white, 15 white yellow red pink blue black, 16 white cyan blue black, 17 white yellow red blue black, 18 vier feste Farben + erste weiß, 19 vier feste Farben + weiß in der Mitte, 20 vier feste Farben + letzte weiß, 21 vier feste Farben + weiß vorn, schwarz hinten, 22 vier feste Farben + schwarz vorn, weiß hinten)
	"3D.colors.number" : Farbpalette: Anzahl Farben bei abgestuften Farben (2..1000)
	"3D.font.iso.size" : Farbkarte: Größe der Schrift für die Beschriftung von ISO-Linien, in pt, z.B. 8.
	"3D.grid" : 3D: Optionen zum Gitter (0 auto, 1 Volumen und Randflächen, 2 Volumen, 3 Volumen und Randflächen, nur Hauptgitter, 4 Volumen, nur Hauptgitter, 5 Auch an Messpunkten, 6 Nur an Messpunkten, 7 Auch an Messpunkten, hinten, 8 Hinten gepunktet, 9 Hinten gepunktet, mit Randflächen, 10 Randflächen
	"3D.interpolation" : Farbkarte: Interpolation (0 auto, 1 linear, 2 konstant)
	"3D.iso.exponent" : Farbkarte: Zehnerpotenz für die Beschriftung der ISO-Linien (-12, -9, ...9)
	"3D.iso.format" : Farbkarte: ISO-Text Format (0 auto, 1 Festkomma, 2 Gleitkomma)
	"3D.iso.text" : Farbkarte: Beschriftung der ISO-Linien (0 auto, 1 nein, 2 ja, 3 möglichst horizontal, 4 möglichst horizontal und frei liegend, 5 möglichst frei liegend, 6 an diskreten x-Positionen, 7 an diskreten x-Positionen eng, 8 an diskreten y-Positionen, 9 an diskreten y-Positionen eng, 10 mehrfach an langen Linien, 11 häufig
	"3D.isolines" : Farbkarte und 3D: ISO-Linien gewünscht (0 nein, 1 ja)
	"3D.lowerlimit" : Farbkarte: Untergrenze. Nur Werte oberhalb werden beachtet.
	"3D.lowerlimit.use" : Farbkarte: Untergrenze beachten (0 nein, 1 ja)
	"3D.perspective" : 3D: Wie wird die Perspektive definniert(0 auto, 1 Scherung mit Winkel der z-Achse, 2 Längen- und Breitengrad, 3 mit Rotation)
	"3D.places.right" : Farbkarte: Nachkommastellen: 0..15
	"3D.represent" : Farbkarte und 3D: Repräsentation (0 auto, 1 Farbkarte kontinuierliche Farbübergänge, 2 Farbkarte abgestufte Farben, 3 Symbole mit Größe nach Amplitude, 4 Symbole mit Füllung wie Amplitude
	"3D.sym.mult" : Farbkarte: Multiplikator maximale Symbolgröße. Faktor, um den die Symbolgröße größer als die Schriftgröße werden darf.

	"3D.symbol.fixed" : Farbkarte: Festes Symbol (0 Quadrat von der Mitte, 1 Quadrat von unten, 2 Qudrat von unten links, 3 Quadrat von aussen, 4 Kreis)
	"3D.symbol.var" : Farbkarte: Variables Symbol (0 Kreis, 1 Quadrat, 2 Raute, 3 gefüllter Kreis, 4 gefülltes Quadrat, 5 gefüllte Raute)
	"3D.z.coordinate.dz" : Farbkarte, 3D: dz bei fest vorgegebener z-Koordinate der Daten.
	"3D.z.coordinate.mode" : Farbkarte, 3D: Bildung der z-Koordinate der Daten. Segmentierte Daten bringen selbst eine z-Koordinate mit. Für andere Daten muss die z-Koordinate ggf. ergänzt werden. (1: Feste Vorgabe 0, 1, 2, 3, ..; 2: Feste Vorgabe z0, dz und z-Einheit, 3: auto, z-Koordinate der Daten wird benutzt)
	"3D.z.coordinate.unit" : Farbkarte, 3D: Einheit bei fest vorgegebener z-Koordinate der Daten.
	"3D.z.coordinate.z0" : Farbkarte, 3D: z0 bei fest vorgegebener z-Koordinate der Daten.
	"polar.skydirection" : Polardiagramm, Anzeige der Himmelsrichtungen festlegen. 0 keine, 4, 8, 16 Himmelsrichtungen
	"polar.displayangles" : Polardiagramm, Anzeige von Winkeln am Kreis, 0 keine, 1: Winkel in Grad [°]
	"polar.spin" : Polardiagramm, Drehrichtung des Winkels, 0 links drehend (mathematisch positiv, 1 rechts drehend (mathematisch negativ)
	"polar.anglezeropos" : Polardiagramm, 0°-Winkelposition, 0 rechts (0°), 1 oben (90°), 2 links (180°), 3 unten (270°), 4 frei definiert
	"polar.angleoffset" : Polardiagramm, Winkeloffset der 0° Position, nur wenn 0°-Winkelposition frei definiert (4) wurde
	"polar.axispos" : Polardiagramm, Position der Y-Achse, 0 innen oben, 1 links oben, 2 rechts oben, 3 innen, 4 links, 5 rechts
	"polar.xasangle" : Polardiagramm, Interpretation der X-Daten als Winkel, 0 kein Winkel, 1 Winkel in °, 2 Winkel in Radiant
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Die x-Koordinate des Messcursors abfragen

```
x = CwDisplayGet("measure.x.max")
```

Siehe auch:

CwDisplaySet, CwDisplayGetText

CwDisplayGetText

Anwendungsbereich: Kurvenfenster

Allgemeine Text-Eigenschaft am Kurvenfenster abfragen

Deklaration:

```
CwDisplayGetText ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"header.text" : Kopf- oder Fußzeile, Überschrift; angezeigter Text. Zugriff auf die aktuell selektierte Kopfzeile
	"title" : Titelzeile bei frei fliegendem Kurvenfenster, "<auto>" für automatische Bestimmung
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Die Titelzeile des Kurvenfensters abfragen

```
caption = CwDisplayGetText("title")
```

Siehe auch:

CwDisplaySet, CwDisplayGet

CwDisplaySet

Anwendungsbereich: Kurvenfenster

Allgemeine Eigenschaft am Kurvenfenster setzen

Deklaration:

```
CwDisplaySet ( Eigenschaft, Wert )
```

Parameter:

Eigenschaft	Welche Eigenschaft soll gesetzt werden?
	"displaymode" : Darstellungsart: 1 Standard, 2 Übereinander, 3 Wasserfall, 4 Farbkarte, 5 Letzter Wert als Zahl, 6 Balkeninstrument, 7 Tabelle, 8 3D, 9 Polardiagramm
	"color palette" : Farbpalette bei Standarddarstellung (0 nein, 1 ja)
	"colors.printer.indiv" : Individuelle Farben für dieses Fenster für den Drucker (0 nein, globale Farben nutzen, 1 ja)
	"colors.printer.pattern" : Kurven in Struktur (0 nein, 1 ja). Nur gültig bei individuellen Farben für dieses Fenster für den Drucker und wenn Linienstruktur auf auto gestellt
	"colors.screen.indiv" : Individuelle Farben für dieses Fenster für den Bildschirm (0 nein, globale Farben nutzen, 1 ja)
	"colors.screen.pattern" : Kurven in Struktur (0 nein, 1 ja). Nur gültig bei individuellen Farben für dieses Fenster für den Bildschirm und wenn Linienstruktur auf auto gestellt
	"cosys.max" : Koordinatensystem maximiert (0 nein, 1 ja)
	"db.reference" : Bezugswert für dB-Berechnungen
	"grid" : Gitter (0 nein, 1 ja)
	"header.coordinate.system" : Kopf- oder Fußzeile, Nummer des Koordinatensystems ab 1 (0 auto). Zugriff auf die aktuell selektierte Kopfzeile
	"header.count" : Anzahl der Texte, die als Kopf- oder Fußzeile oder Überschrift angezeigt werden sollen
	"header.position" : Kopf- oder Fußzeile, Überschrift; Lage (0 auto, 1 oben links, 2 oben Mitte, 3 oben rechts, 4 unten links, 5 unten Mitte, 6 unten rechts, 7 links Mitte, 8 Mitte, 9 rechts Mitte, 10 oben links Koordinatensystem, 11 oben Mitte Koordinatensystem, 12 oben rechts Koordinatensystem, 13 unten links Koordinatensystem, 14 unten Mitte Koordinatensystem, 15 unten rechts Koordinatensystem, 16 links Mitte Koordinatensystem, 17 Mitte Koordinatensystem, 18 rechts Mitte Koordinatensystem). Zugriff auf die aktuell selektierte Kopfzeile
	"header.shiftx" : Kopf- oder Fußzeile, Überschrift; Verschiebung des Bezugspunktes des Textes in x-Richtung, in mm, positiv nach rechts, negativ nach links. Zugriff auf die aktuell selektierte Kopfzeile
	"header.shifty" : Kopf- oder Fußzeile, Überschrift; Verschiebung des Bezugspunktes des Textes in y-Richtung, in mm, positiv nach oben, negativ nach unten. Zugriff auf die aktuell selektierte Kopfzeile
	"header.text" : Kopf- oder Fußzeile, Überschrift; angezeigter Text. Zugriff auf die aktuell selektierte Kopfzeile
	"header.text.angle" : Kopf- oder Fußzeile, Überschrift; Neigung des Textes in Grad (-90 .. +90). Wie stark ist der Text gegen die Horizontale geneigt? Zugriff auf die aktuell selektierte Kopfzeile
	"header.text.color" : Kopf- oder Fußzeile, Überschrift; Textfarbe, Format siehe rgb(), -1 automatisch. Zugriff auf die aktuell selektierte Kopfzeile
	"header.text.size" : Kopf- oder Fußzeile, Überschrift; Größe der Schrift, in pt, z.B. 8 (0 auto). Zugriff auf die aktuell selektierte Kopfzeile
	"header.text.style" : Kopf- oder Fußzeile, Überschrift; Schriftstil (0 auto, 1 Standard, 2 fett, 3 kursiv, 4 fett und kursiv, 5 unterstrichen, 6 fett und unterstrichen, 7 kursiv und unterstrichen, 8 fett und kursiv und unterstrichen). Zugriff auf die aktuell selektierte Kopfzeile
	"history.size" : Max. Speicher für die Historie in MByte. -1 für auto
	"instrument.numval.show" : Instrument: Numerischen Wert anzeigen (0 nein, 1 linksbündig, 2 zentriert, 3 rechtsbündig)
	"instrument.title.show" : Instrument: Titel anzeigen (0 nein, 1 linksbündig, 2 zentriert, 3 rechtsbündig)
	"instrument.axis.show" : Instrument: Achse anzeigen (0 nein, 1 ja)
	"instrument.tooltip" : Instrument: Tooltip anzeigen (0 nein, 1 Name)
	"instrument.slavepointer" : Instrument: Schleppzeiger (1 auto, 2 Linie)
	"instrument.frame" : Instrument: Rahmen (1 auto, 2 gemeinsam)
	"labels" : Beschriftung (0 nein, 1 ja)

	"lastvalue.columns" : Letzter Wert als Zahl: Spalten (0 nein, 1 ja)
	"lastvalue.font.size" : Letzter Wert als Zahl: Schriftgröße, in pt, z.B. 8 (0 auto)
	"lastvalue.names" : Letzter Wert als Zahl: Namen (0 nein, 1 ja, 2 Kommentar, 3 Name und Kommentar)
	"lastvalue.right" : Letzter Wert als Zahl: rechtsbündig (0 nein, 1 ja)
	"lastvalue.=" : Letzter Wert als Zahl: Gleichheitszeichen (0 nein, 1 ja)
	"legend.border" : Legende Rand (0 nein, 1 ja)
	"legend.content" : Legende Textinhalt (0 Kanalname, 1 Kanalname (ohne Gruppenname), 2 Kommentar des Kanals, 3 Kanalname und Kommentar, 4 Kanalname ohne Gruppenname und Kommentar, 5 Kanalname ohne Messung, 6 Kanalname mit Nummer der selekt. Messung, 7 Kanalname ohne Gruppe und Messung, 8 Kanalanme mit Kommentar ohne Gruppe und Messung)
	"legend.curvecol" : Legende Text in Kurvenfarbe (0 nein, 1 ja)
	"legend.display" : Legende Anwesenheit (0 auto, 1 immer, 2 nie, 3 bei mehr als 1 Kurve)
	"legend.distx" : Legende Abstand zur Kante horizontal [mm] bei beweglicher Legende
	"legend.disty" : Legende Abstand zur Kante vertikal [mm] bei beweglicher Legende
	"legend.font.size" : Legende Schriftgröße, in pt, z.B. 8 (0 auto)
	"legend.font.style" : Legende Schriftstil (0 auto, 1 Standard, 2 fett, 3 kursiv, 4 fett und kursiv, 5 unterstrichen, 6 fett und unterstrichen, 7 kursiv und unterstrichen, 8 fett und kursiv und unterstrichen)
	"legend.lines" : Legende Linienprobe (0 nein, 1 ja)
	"legend.location" : Legende Lage (0 oben, 1 oben über jedem Ko'system, 2 links, 3 links neben jedem Ko'system, 4 relativ oben links, 5 relativ oben rechts, 6 relativ unten links, 7 relativ unten rechts, 8 relativ rechts, 9 relativ links, 10 relativ oben, 11 relativ unten, 12 relativ Mitte, 13 relativ oben links mehrfach, 14 relativ oben rechts mehrfach, 15 relativ unten links mehrfach, 16 relativ unten rechts mehrfach, 17 relativ rechts mehrfach, 18 relativ links mehrfach, 19 relativ oben mehrfach, 20 relativ unten mehrfach, 21 relativ Mitte mehrfach)
	"legend.numerical.display" : Legende mit Zahlenwert (0 nein, 1 letzter Wert als Zahl)
	"legend.numerical.maxdigits" : Legende Zahlenwert max. Anzahl Ziffern (0..15)
	"legend.numerical.sep" : Legende Zahlenwert mit Trennzeichen (0 nein, 1 :, 2 =)
	"legend.numerical.unit" : Legende Zahlenwert mit Einheit (0 nein, 1 ja)
	"legend.rowcol" : Legende Zeilen/Spalten (0 Zeilen, Spalten automatisch, 1 Feste Zeilenanzahl, 2 Feste Spaltenanzahl)
	"legend.rows" : Legende Anzahl der Zeilen bzw. Spalten bei der Option Feste Zeilenanzahl bzw. Feste Spaltenanzahl
	"legend.space.bottom" : Legende zusätzlicher Platz am Rand unten [mm]
	"legend.space.left" : Legende zusätzlicher Platz am Rand links [mm]
	"legend.space.right" : Legende zusätzlicher Platz am Rand rechts [mm]
	"legend.space.top" : Legende zusätzlicher Platz am Rand oben [mm]
	"legend.transparent" : Legende transparent (0 nein, 1 ja)
	"link.color" : Link: Farbe der Markierungslinie, Format siehe rgb(), -1 für automatisch
	"link.coordinate" : Link: Steuerung über andere Koordinate (0 auto, 1 x und y vertauschen, 2 original beibehalten)
	"link.edge.mouse" : Link: Anpassen der Achsen bei Mausbewegung am Fensterrand (0 auto, 1 nein, 2 Vergrößern, 3 Verschieben)
	"link.edge.position" : Link: Anpassen der Achsen, wenn die Markierung beim Folgen an den Rand kommt (0 auto, 1 nein, 2 Vergrößern, 3 Verschieben)
	"link.follow" : Link: Dieses Fenster folgt (0 auto, 1 Achse folgt, 2 Linie folgt)
	"link.influence" : Link: Was wird vom Link beeinflusst? (0 auto, 1 x-Achse, 2 Parameter einer XY-Darstellung, 3 x-, y-Achsen (Farbkarte), 4 y-Achse, 5 Schnitt)
	"link.linestyle" : Link: Linienart der Markierungslinie (0 auto, 1 durchgezogen, 2 gepunktet, 3 gestrichelt, 4..13 andere Kombinationen)
	"link.scale" : Link: Bei Änderung der Skalierung (0 auto, 1 Linie folgt, 2 Linie bleibt an Bildschirmposition)
	"link.shape" : Link: Grafische Form der Markierung (0 auto, 1 Linie)
	"link.width" : Link: Dicke der Markierung der Linie in mm (0 auto)
	"map.backgnd" : Landkarte: Hintergrund (0 auto, 1 Landkarte, 2 Hintergrundbild, 3 Landkarte aus Internet). Beim Setzen auf "auto" gehen alle Einstellungen zur Landkarte verloren.

	"map.draw" : Landkarte: Hintergrundbild zeichnen (0 nur auf Bildschirm, 1 auch auf Drucker)
	"map.filename" : Veraltet, bitte CwActionP benutzen! Landkarte: Name der Datei (komplett mit Pfad und Erweiterung), aus der das Bild gelesen werden soll. Der Dateiname wird nur im direkt danach folgenden Aufruf von CwAction ("map.load") benutzt.
	"map.filled" : Landkarte: Welchen Teil des Fensters soll das Bild überdecken? (0 Nur Koordinatensytem, 1 gesamtes Fenster)
	"map.fit" : Landkarte: Hintergrundbild Streckung (0 auto, 1 horizontal anpassen, 2 vertikal anpassen, 3 Größe beibehalten)
	"map.lat1" : Landkarte: Breitengrad des 1. Punktes in Grad
	"map.lat2" : Landkarte: Breitengrad des 2. Punktes in Grad
	"map.lon1" : Landkarte: Längengrad des 1. Punktes in Grad
	"map.lon2" : Landkarte: Längengrad des 2. Punktes in Grad
	"map.scale.adjust" : Landkarte: Maßstab anpassen (0 nein, 1 x, y-Achse passend zu Internetkarten skalieren)
	"map.x1" : Landkarte: Relative x-Koordinate des 1. Punktes, 0.0 für ganz links, 1.0 für ganz rechts im Bild.
	"map.x2" : Landkarte: Relative x-Koordinate des 2. Punktes, 0.0 für ganz links, 1.0 für ganz rechts im Bild.
	"map.y1" : Landkarte: Relative y-Koordinate des 1. Punktes, 0.0 für ganz unten, 1.0 für ganz oben im Bild.
	"map.y2" : Landkarte: Relative y-Koordinate des 2. Punktes, 0.0 für ganz unten, 1.0 für ganz oben im Bild.
	"measure.free" : Bei offenem Messwertfenster den Messcursor frei bewegen. (0 nein, 1 ja)
	"measure.p.left" : Bei offenem Messwertfenster der Parameter des Cursors (bei XY-Darstellung) zur linken Maustaste
	"measure.p.right" : Bei offenem Messwertfenster der Parameter des Cursors (bei XY-Darstellung) zur rechten Maustaste
	"measure.x.left" : Bei offenem Messwertfenster die x-Position des Cursors zur linken Maustaste. Nicht bei XY-Darstellungen.
	"measure.x.right" : Bei offenem Messwertfenster die x-Position des Cursors zur rechten Maustaste. Nicht bei XY-Darstellungen.
	"measure.y.left" : Bei offenem Messwertfenster der y-Wert des Cursors zur linken Maustaste. Nur für Darstellungen, bei denen die y-Koordinate des Cursors frei bewegt werden kann, z.B. Farbkarte.
	"measure.y.right" : Bei offenem Messwertfenster der y-Wert des Cursors zur rechten Maustaste. Nur für Darstellungen, bei denen die y-Koordinate des Cursors frei bewegt werden kann, z.B. Farbkarte.
	"name" : Titel des Kurvenfensters. Nur für Sonderanwendungen, in denen ein frei fliegendes Kurvenfenster einen Titel zur Identifikation erhalten muss, etwa für RgCurveSet().
	"number.trimm" : Abgeschnittene Zahlen (0 nein, 1 ja)
	"opt.on.delete" : Optimieren beim Löschen von Datensätzen (0 nein, 1 ja)
	"scroll" : Rollmodus (0 nein, 1 rollen, 2 wachsen, 3 Oszilloskop, 4 füllen)
	"shift.in.cfg" : Sollen die Werte für Timeshift und Amplitudenshift in der Konfiguration (ccv) abgelegt werden? (0 nein, 1 ja)
	"showtrigger" : Triggerlinie zeigen (0 nein, 1 ja)
	"splitmode.active" : Aktive Ansicht im Splitmodus (1 bis Anzahl der Ansichten). 0, falls kein Splitmodus. Z.B. Änderungen des Wertebereichs der x-Achse wirken auf der aktiven Ansicht.
	"splitmode.count" : Anzahl der Ansichten im Splitmodus. 1 für eine einzige Ansicht ohne Splitmodus
	"splitmode.width" : Splitmodus Breite der aktiven Ansicht in Prozent (0 bis 100) der Gesamtbreite
	"suppress.lines.opt" : Optimierung bei ausgeblendeten Linien (0 nein: Achsen und Koordinatensysteme bleiben sichtbar; 1 ja: Achsen und Koordinatensysteme werden möglichst verkleinert oder gar unsichtbar, wenn alle zugehörigen Linien ausgeblendet sind.)
	"title" : Titelzeile bei frei fliegendem Kurvenfenster, "<auto>" für automatische Bestimmung
	"toolbar.on" : In Sonderanwendungen der Toolbar bei frei fliegendem Kurvenfenster (0 komplett abgeschaltet, Kontextmenü wie bei eingebettetem Fenster. 1 an, Standard)
	"win.dx" : Breite des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"win.dy" : Höhe des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"win.enable.cursors" : Ist es dem Anwender möglich, an den Cursoren zu ziehen und sie zu bewegen: Messcursoren, Linklinie etc. (0 nein; 1 ja). Ist bei eingebetteten Kurvenfenstern im Design-Modus nicht wirksam.
	"win.x" : Position der linken Kante des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"win.y" : Position der oberen Kante des Kurvenfensters auf dem Bildschirm für frei fliegende Kurvenfenster, in Pixeln
	"3D.angle1" : 3D: Erste Winkelangabe zur Definition der Perspektive, in Grad (Länge oder Winkel z-Achse)
	"3D.angle2" : 3D: Zweite Winkelangabe zur Definition der Perspektive, in Grad (Breite)

	"3D.angle3" : 3D: Dritte Winkelangabe zur Definition der Perspektive, in Grad (Rotation)
	"3D.color.fillsymbol" : Farbkarte: Farbe Symbolfüllung, Format siehe rgb()
	"3D.color.fix1" : Farbpalette: 1. feste Farbe, Format siehe rgb()
	"3D.color.fix2" : Farbpalette: 2. feste Farbe, Format siehe rgb()
	"3D.color.fix3" : Farbpalette: 3. feste Farbe, Format siehe rgb()
	"3D.color.fix4" : Farbpalette: 4. feste Farbe, Format siehe rgb()
	"3D.color.isoback" : Farbkarte: Farbe ISO-Linien Hintergrund, Format siehe rgb(), -1 für automatisch, -2 für transparent
	"3D.color.isoborder" : Farbkarte: Farbe ISO-Linien Rand, Format siehe rgb(), -1 für automatisch, -2 für transparent
	"3D.color.isolines" : Farbkarte und 3D: Farbe ISO-Linien, Format siehe rgb()
	"3D.color.isotext" : Farbkarte: Farbe ISO-Linientext, Format siehe rgb()
	"3D.color.symbolborder" : Farbkarte: FarbeSymbolrand, Format siehe rgb(), -1 für automatisch, -2 für transparent
	"3D.color bar.width" : Farbpalette: Farbprobe Breite in mm, 0 für automatisch
	"3D.color legend" : 3D: Farblegende Anwesenheit (0 auto, 1 nein, 2 Farblegende an vertikaler y-Achse, 3 separate Farblegende rechts, 4 separate Farblegende links)
	"3D.color pattern" : Farbpalette: Farbschema (0 auto, 1 zwei feste Farben, 2 drei feste Farben, 3 vier feste Farben, 4 spektral, 5 schwarz weiss kompatibel, 6 schwarz weiss, 7 black blue green yellow red, 8 blue cyan green yellow red, 9 blue green yellow red, 10 black blue pink red yellow white, 11 green cyan blue pink red, 12 black blue cyan green yellow red, 13 black blue red yellow white, 14 black blue cyan white, 15 white yellow red pink blue black, 16 white cyan blue black, 17 white yellow red blue black, 18 vier feste Farben + erste weiß, 19 vier feste Farben + weiß in der Mitte, 20 vier feste Farben + letzte weiß, 21 vier feste Farben + weiß vorn, schwarz hinten, 22 vier feste Farben + schwarz vorn, weiß hinten). Nur wirksam, wenn "3D.represent" passend auf 1 oder 2 gesetzt ist.
	"3D.colors.number" : Farbpalette: Anzahl Farben bei abgestuften Farben (2..1000)
	"3D.font.iso.size" : Farbkarte: Größe der Schrift für die Beschriftung von ISO-Linien, in pt, z.B. 8.
	"3D.grid" : 3D: Optionen zum Gitter (0 auto, 1 Volumen und Randflächen, 2 Volumen, 3 Volumen und Randflächen, nur Hauptgitter, 4 Volumen, nur Hauptgitter, 5 Auch an Messpunkten, 6 Nur an Messpunkten, 7 Auch an Messpunkten, hinten, 8 Hinten gepunktet, 9 Hinten gepunktet, mit Randflächen, 10 Randflächen
	"3D.interpolation" : Farbkarte: Interpolation (0 auto, 1 linear, 2 konstant)
	"3D.iso.exponent" : Farbkarte: Zehnerpotenz für die Beschriftung der ISO-Linien (-12, -9, ...9)
	"3D.iso.format" : Farbkarte: ISO-Text Format (0 auto, 1 Festkomma, 2 Gleitkomma)
	"3D.iso.text" : Farbkarte: Beschriftung der ISO-Linien (0 auto, 1 nein, 2 ja, 3 möglichst horizontal, 4 möglichst horizontal und frei liegend, 5 möglichst frei liegend, 6 an diskreten x-Positionen, 7 an diskreten x-Positionen eng, 8 an diskreten y-Positionen, 9 an diskreten y-Positionen eng, 10 mehrfach an langen Linien, 11 häufig
	"3D.isolines" : Farbkarte und 3D: ISO-Linien gewünscht (0 nein, 1 ja)
	"3D.lowerlimit" : Farbkarte: Untergrenze. Nur Werte oberhalb werden beachtet.
	"3D.lowerlimit.use" : Farbkarte: Untergrenze beachten (0 nein, 1 ja)
	"3D.perspective" : 3D: Wie wird die Perspektive definniert(0 auto, 1 Scherung mit Winkel der z-Achse, 2 Längen- und Breitengrad, 3 mit Rotation)
	"3D.places.right" : Farbkarte: Nachkommastellen: 0..15
	"3D.represent" : Farbkarte und 3D: Repräsentation (0 auto, 1 Farbkarte kontinuierliche Farbübergänge, 2 Farbkarte abgestufte Farben, 3 Symbole mit Größe nach Amplitude, 4 Symbole mit Füllung wie Amplitude
	"3D.sym.mult" : Farbkarte: Multiplikator maximale Symbolgröße. Faktor, um den die Symbolgröße größer als die Schriftgröße werden darf.
	"3D.symbol.fixed" : Farbkarte: Festes Symbol (0 Quadrat von der Mitte, 1 Quadrat von unten, 2 Qudrat von unten links, 3 Quadrat von aussen, 4 Kreis)
	"3D.symbol.var" : Farbkarte: Variables Symbol (0 Kreis, 1 Quadrat, 2 Raute, 3 gefüllter Kreis, 4 gefülltes Quadrat, 5 gefüllte Raute)
	"3D.z.coordinate.dz" : Farbkarte, 3D: dz bei fest vorgegebener z-Koordinate der Daten. Nur wirksam, falls "3D.z.coordinate.mode" auf 2 gesetzt ist.
	"3D.z.coordinate.mode" : Farbkarte, 3D: Bildung der z-Koordinate der Daten. Segmentierte Daten bringen selbst eine z-Koordinate mit. Für andere Daten muss die z-Koordinate ggf. ergänzt werden. (1: Feste Vorgabe 0, 1, 2, 3, ..; 2: Feste Vorgabe z0, dz und z-Einheit, 3: auto, z-Koordinate der Daten wird benutzt)
	"3D.z.coordinate.unit" : Farbkarte, 3D: Einheit bei fest vorgegebener z-Koordinate der Daten. Nur wirksam, falls "3D.z.coordinate.mode" auf 2 gesetzt ist.

	"3D.z.coordinate.z0" : Farbkarte, 3D: z0 bei fest vorgegebener z-Koordinate der Daten. Nur wirksam, falls "3D.z.coordinate.mode" auf 2 gesetzt ist.
	"polar.skydirection" : Polardiagramm, Anzeige der Himmelsrichtungen festlegen. 0 keine, 4, 8, 16 Himmelsrichtungen
	"polar.displayangles" : Polardiagramm, Anzeige von Winkeln am Kreis, 0 keine, 1: Winkel in Grad [°]
	"polar.spin" : Polardiagramm, Drehrichtung des Winkels, 0 links drehend (mathematisch positiv, 1 rechts drehend (mathematisch negativ)
	"polar.anglezeropos" : Polardiagramm, Winkeloffset, 0 rechts (0°) , 1 oben (90°), 2 links (180°), 3 unten (270°), 4 frei definiert
	"polar.angleoffset" : Polardiagramm, 0° Winkeloffset
	"polar.axispos" : Polardiagramm, Position der Y-Achse, 0 innen oben, 1 links oben, 2 rechts oben, 3 innen, 4 links, 5 rechts
	"polar.xasangle" : Polardiagramm, Interpretation der X-Daten , 0 kein Winkel, 1 Winkel in °, 2 Winkel als Radiant
Wert	Auf welchen Wert soll diese Eigenschaft gesetzt werden?

Beschreibung:

Beispiele:

Die Darstellungsart auf Standard setzen

```
CwSelectWindow("curve1")
CwDisplaySet("displaymode", 1)
```

3D Farbpalette

```
CwSelectWindow("curve1")
CwDisplaySet("displaymode", 8)
CwDisplaySet("3D.color pattern", 1)
CwDisplaySet("3D.color.fix1", rgb(0,255,0))
CwDisplaySet("3D.color.fix1", rgb(0,0,255))
```

Farbpalette in Standarddarstellung

```
CwNewWindow( "1", "show")
CwDisplaySet("displaymode", 1)
CwDisplaySet("color palette", 1)
CwDisplaySet("3D.color pattern",4)
CwDisplaySet("3D.colors.number",10)
CwNewChannel("append last axis", sintest1)
CwNewChannel("append last axis", sintest2)
CwSelectByChannel( "data", sintest2)
CwDataSet("function", 1)
CwSelectByIndex( "z-axis", 1 )
CwAxisSet( "visible", 0)
```

Kopf- oder Fußzeile oder Überschrift: Zuerst die Anzahl setzen. Dann die Kopfzeile selektieren und einstellen.

```
CwDisplaySet("header.count", 1)
CwSelectByIndex("header", 1)
CwDisplaySet("header.text", "Title")
CwDisplaySet("header.text.size", 12)
CwDisplaySet("legend.space.top", 10)
```

Siehe auch:

CwDisplayGet, CwDisplayGetText

CwGlobalGet

Anwendungsbereich: Kurvenfenster

Globale Eigenschaft abfragen

Deklaration:

```
CwGlobalGet ( Eigenschaft, Parameter ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"caption.option" : Wie wird die Titelzeile eines frei fliegenden Kurvenfensters bestimmt. (0 auto, 1 Dateiname einer geladenen ccv-Datei benutzen)
	"colors.printer.pattern" : Kurven in Struktur (0 nein, 1 ja). Gilt für alle Kurvenfenster für den Drucker, solange nicht auf individuelle Farben umgestellt.
	"colors.screen.pattern" : Kurven in Struktur (0 nein, 1 ja). Gilt für alle Kurvenfenster für den Bildschirm, solange nicht auf individuelle Farben umgestellt.
	"double click.empty" : Reaktion bei Doppelklick auf freie Flächen. (0 keine Aktion, 1 Selektionsmodus ein/ausschalten)
	"export.dpi" : Grafikexport Auflösung in dpi (150, 300, 600, 1200)
	"export.orientation" : PDF Export, Ausrichtung (0 auto, 1 hoch, 2 quer)
	"export.pdf.append" : PDF-Export, Anhängen von Seiten bei bereits vorhandener PDF-Datei. (0 nein, 1 ja) Nur wirksam bei manuellem Export.
	"export.pdf.method" : PDF-Export, Verfahren zur Erzeugung (0: automatisch, 1 Bitmap, 2 möglichst Vektorgrafik, beste Qualität, mittels XPS Document Writer)
	"graphics.type" : Grafik Typ beim Kopieren in die Ablage, ggf. Drucken und auch Export. (0 auto, Vektorgrafik, 1 Bitmap Pixelgrafik, 2 exakte Bildschirmdarstellung)
	"measure.cursor.change" : Wie verhält sich der Messcursor bei Achsenänderung? (0 Messcursor bleibt auf Pixelposition, 1 Messcursor bleibt an seiner Koordinate)
	"measure.cursor.hori" : Horizontaler Messcursor? (0 nein, 1 ja)
	"sample.index" : Index des selektierten Messpunktes. Beginnt mit 1 bei erstem Messpunkt. Bei Daten mit Segmenten oder Events wird ab dem allerersten Messpunkt überhaupt gezählt.
	"sample.x" : x-Komponente (2. Komponente bzw. Phase oder Realteil) des selektierten Messpunktes
	"sample.y" : y-Komponente (1. Komponente bzw. Betrag oder Imaginärteil) des selektierten Messpunktes
	"sample.z" : z-Koordinate des selektierten Messpunktes. Z.B. Koordinate des farbgebenden Kanals
	"y-axis.navi.x" : Ändert sich die y-Achse, wenn in x-Richtung navigiert wird. (0 y-Achsen fixieren, 1 y-Achsen bleiben automatisch)
Parameter	Parameter, abhängig von der Eigenschaft, i. Allg. = 0
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Im Kurvenfenster wird ein Messpunkt selektiert und anschließend über eine Kitfunktion der Messwert abgefragt:

```
y = CwGlobalGet("sample.y", 0)
```

Siehe auch:

CwGlobalSet, CwGlobalGetText

CwGlobalGetText

Anwendungsbereich: Kurvenfenster

Globale Text-Eigenschaft abfragen

Deklaration:

```
CwGlobalGetText ( Eigenschaft, Parameter ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"sample.x.name" : bei xy-Darstellung Kanalname der x-Komponente
	"sample.y.name" : Kanalname des selektierten Messpunktes bzw. bei xy-Darstellung Kanalname der y-Komponente
	"sample.z.name" : Kanalname des farbgebenden Kanals des selektierten Messpunktes, bei xyz-Darstellung Kanalname der z-Komponente
Parameter	Parameter, abhängig von der Eigenschaft, i. Allg. = 0
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Im Kurvenfenster wird ein Messpunkt selektiert und anschließend über eine Kitfunktion der Kanalname abgefragt:

```
name = CwGlobalGetText("sample.y.name", 0)
```

Siehe auch:

CwGlobalSet, CwGlobalGet

CwGlobalSet

Anwendungsbereich: Kurvenfenster

Allgemeine globale Eigenschaft für alle Kurvenfenster setzen

Deklaration:

```
CwGlobalSet ( Eigenschaft, Wert )
```

Parameter:

Eigenschaft	Welche Eigenschaft soll gesetzt werden?
	"caption.option" : Wie wird die Titelzeile eines frei fliegenden Kurvenfensters bestimmt. (0 auto, 1 Dateiname einer geladenen ccv-Datei benutzen)
	"close all" : Schließt alle Kurvenfenster, die frei fliegend sind. (Wert = 0 setzen)
	"colors.printer.pattern" : Kurven in Struktur (0 nein, 1 ja). Gilt für alle Kurvenfenster für den Drucker, solange nicht auf individuelle Farben umgestellt.
	"colors.screen.pattern" : Kurven in Struktur (0 nein, 1 ja). Gilt für alle Kurvenfenster für den Bildschirm, solange nicht auf individuelle Farben umgestellt.
	"dir.ccv" : Setzt das Standard-Verzeichnis für ccv-Dateien. Gilt für die aktuelle Sitzung. Wird nicht beim Arbeiten in Projekten benutzt.
	"double click.empty" : Reaktion bei Doppelklick auf freie Flächen. (0 keine Aktion, 1 Selektionsmodus ein/ausschalten)
	"export.dpi" : Grafikexport Auflösung in dpi (150, 300, 600, 1200)
	"export.orientation" : PDF Export, Ausrichtung (0 auto, 1 hoch, 2 quer)
	"export.pdf.append" : PDF-Export, Anhängen von Seiten bei bereits vorhandener PDF-Datei. (0 nein, 1 ja) Nur wirksam bei manuellem Export.
	"export.pdf.method" : PDF-Export, Verfahren zur Erzeugung (0: automatisch, 1 Bitmap, 2 möglichst Vektorgrafik, beste Qualität, mittels XPS Document Writer)
	"font.name" : Name der Standard-Schriftart für Beschriftungen in Kurvenfenstern, z.B. "Arial".
	"font.size" : Größe der Standard-Schrift für Beschriftungen in Kurvenfenstern, in pt, z.B. 8.
	"graphics.type" : Grafik Typ beim Kopieren in die Ablage, ggf. Drucken und auch Export. (0 auto, Vektorgrafik, 1 Bitmap Pixelgrafik, 2 exakte Bildschirmdarstellung)
	"infront.of.main" : Wirkt wie die Famos-Option "Nie vom Hauptfenster verdeckt": 0 Hauptfenster kann auch das Kurvenfenster verdecken, 1 Kurvenfenster nie vom Hauptfenster verdeckt.Wirksam für frei fliegende Kurvenfenster, die durch dieses Kit angelegt werden. Ohne Aufruf der Funktion gilt die 0.
	"load.show" : Wie soll ein Kurvenfenster nach dem Laden einer Konfiguration angezeigt werden (0 immer versteckt, 1 automatisch wie in der CCV-Datei vermerkt,). Wirksam für frei fliegende Kurvenfenster, die durch dieses Kit geladen werden. Ohne Aufruf der Funktion gilt die 1.
	"measure.cursor.change" : Wie verhält sich der Messcursor bei Achsenänderung? (0 Messcursor bleibt auf Pixelposition, 1 Messcursor bleibt an seiner Koordinate)
	"measure.cursor.hori" : Horizontaler Messcursor? (0 nein, 1 ja)
	"y-axis.navi.x" : Ändert sich die y-Achse, wenn in x-Richtung navigiert wird. (0 y-Achsen fixieren, 1 y-Achsen bleiben automatisch)
Wert	Auf welchen Wert soll diese Eigenschaft gesetzt werden?

Beschreibung:

Beispiele:

Die Schriftgröße für Kurvenfenster setzen

```
CwGlobalSet("font.size", 10) ; font size to 12 pt
```

Einstellungen für einen pdf Export

```
CwGlobalSet("graphics.type", 0)
CwGlobalSet("export.dpi", 300)
CwGlobalSet("export.pdf.append", 0)
CwGlobalSet("export.orientation", 1)
CwGlobalSet("export.pdf.method", 2)
CwPrintSet("layout", 0)
```

```
CwPrintSet("width", 180)
CwPrintSet("height", 260)
```

Siehe auch:

CwGlobalGet, CwGlobalGetText, CwPrintSet

CwIsWindow

Anwendungsbereich: Kurvenfenster

Stellt fest, ob das angegebene Kurvenfenster existiert.

Deklaration:

```
CwIsWindow ( Identifikation ) -> Exist
```

Parameter:

Identifikation	Dieser Datensatz identifiziert das Kurvenfenster. Mit CwSelectMode wurde festgelegt, wie die Identifikation erfolgt.
Exist	Exist (=0, falls nicht vorhanden; <> 0, falls vorhanden)

Beschreibung:

Beispiele:

```
data=rampe(0,1,10)
wenn CwIsWindow(data) <> 0
   ; z.B. das Kurvenfenster selektieren und damit arbeiten
ende
```

Siehe auch:

CwSelectMode

CwLineGet

Anwendungsbereich: Kurvenfenster

Eigenschaft einer Linie abfragen

Deklaration:

```
CwLineGet ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"type" : Linientyp: 0 keine, 1 Linien, 2 Treppen, 3 Punkte, 4 vert. Linie, 5 Balken, 7 3D-Balken, 8 Interpolation
	"color.printer" : Farbe der Linie auf dem Drucker, Format siehe rgb() und -1 für automatisch
	"color.screen" : Farbe der Linie auf dem Bildschirm, Format siehe rgb().
	"area.border" : Rand bei Darstellung von Flächen unter Kurven, die selbst nicht Treppen oder Geraden sind (0 auto, 1 Treppen, 2 linear)
	"area.color" : Bei Darstellung von Flächen unter Kurven wird diese Farbe benutzt. Format siehe rgb() und -1 für automatisch
	"area.color2" : Bei Darstellung eines Farbverlaufs ist das die 2. Farbe. Format siehe rgb() und -1 für automatisch
	"area.fill" : Soll die Fläche unter der Kurve gefüllt werden (0 nein, 1 bis y=null, 2 bis unten, 3 innen)
	"area.gradient" : Farbverlauf bei Darstellung von Flächen unter Kurven (0 nein, 1 von oben nach unten, 2 von unten nach oben, 3 von links nach rechts, 4 von rechts nach links)
	"auxiliary" : Linie ist eine Hilfslinie (0 nein, 1 ja)
	"bar.begin.y" : Balken Beginn in y-Richtung (0 auto, 1 Beginn bei 0, 2 Beginn auf Grundfläche, 3 Beginn unter der Grundfläche)
	"bar.color.type" : Balken Farbgebung (0 auto, 1 mit Farbverlauf, 2 einfarbig)
	"bar.place.x" : Balken Anordnung in x-Richtung (0 auto, 1 zwischen einem Messwert und dem nächsten, 2 zentriert um Messwert, 3 bündig am Messwert)
	"bar.place.z" : Balken Anordnung in z-Richtung (0 auto, 1 zwischen einem Messwert und dem nächsten, 2 zentriert um Messwert, 3 bündig am Messwert)
	"bar.width.x" : Balken Breite in x-Richtung, in Prozent angegeben: 1 bis 100, 0 auto
	"bar.width.z" : Balken Breite in z-Richtung, in Prozent angegeben: 1 bis 100, 0 auto
	"color2" : Farbe 2, z.B. bei 3D die Farbe der Oberfläche. Format siehe rgb() und -1 für automatisch.
	"count.data" : Anzahl Datenelemente, die in dieser Linie enthalten sind.
	"cross section.option" : Soll ein Schnitt von den Daten berechnet werden? (0 nein, 1 x = konstant, 2 y = konstant)
	"cross section.value" : Der Wert, an dem der Schnitt erfolgen soll.
	"effect" : Sollen die Daten (meist bei Farbkarte) z.B. als echte Linie dargestellt werden (0 auto, 1 Überlagerung, 2 obere Grenze, 3 untere Grenze, 4 linke Grenze, 5 rechte Grenze, 6 äußere Grenze, 7 innere Grenze, 8 RGB)
	"label.color" : Farbe der Beschriftung, Format siehe rgb(), (-1 auto)
	"label.font.size" : Beschriftung der Linie Schriftgröße, in pt, z.B. 8
	"label.format" : Zahlenformat der Beschriftung (0 auto, 1 Festkomma, 2 Gleitkomma)
	"label.option" : Sollen die Messpunkte mit ihrem Zahlenwert beschriftet werden. (0 nein, 1 ja)
	"label.placement" : Lage der Beschriftung (0 auto, 1 rechts, 2 rechts oben, 3 rechts unten, 4 links, 5 links oben, 6 links unten, 7 Mitte, 8 Mitte oben, 9 Mitte unten)
	"label.precision" : Präzision der Zahlenwerte der Beschriftung. Bei Fest-,Gleitkomma die Nachkommastellen 0..14, sonst Anzahl der gültigen Ziffern 1..14
	"label.selection" : Werteauswahl für die Beschriftung (0 auto, 1 y, 2 x, 3 Parameter, 4 Betrag, 5 Phase)
	"legend.append" : Namenserweiterung in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.channel" : Kanalnamen in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.comment" : Kommentar in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.group" : Gruppennamen in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.measurement" : Messung in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein, 3 Nummer der selektierten Messung)

	"legend.numerical.format" : Format der Zahlenwerte in der Legende zur Linie (0 auto, 1 Festkomma, 2 Gleitkomma)
	"legend.numerical.precision" : Präzision der Zahlenwerte in der Legende zur Linie. Bei Fest-,Gleitkomma die Nachkommastellen 0..14, sonst Anzahl der gültigen Ziffern 1..14
	"legend.numerical.option" : Zahlenwerte in Legende zur Linie zulassen (0 auto, 1 nein)
	"legend.sample" : Linienprobe in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.show" : Legende zur Linie zeigen (0 auto, 1 ja, 2 nein, 3 Text mit Platzhaltern)
	"linestyle.printer" : Linienart auf dem Drucker (0 auto, 1 durchgezogen, 2 gepunktet, 3 gestrichelt, 4..13 andere Kombinationen)
	"linestyle.screen" : Linienart auf dem Bildschirm (0 auto, 1 durchgezogen, 2 gepunktet, 3 gestrichelt, 4..13 andere Kombinationen)
	"scale.type" : Skalierung der Linie (0 auto, 1 von 0 bis 1 relativ, 2 von 0 bis 1 mit Offset, 3 in mm relativ, 4 in mm mit Offset, 5 Bild strecken, 6 Bild horizontal anpassen, 7 Bild vertikal anpassen, 8 Bild komplett darstellen, 9 Bild Größe beibehalten)
	"shift.x" : Timeshift in physikalischen Einheiten. Bei linear ein Offset, bei logarithmisch ein Faktor.
	"shift.y" : Amplitudenshift in physikalischen Einheiten. Bei linear ein Offset, bei logarithmisch ein Faktor.
	"suppress" : Linie ausblenden (0 nein Standard, 1 ja)
	"symbol" : Symbol: 0 keins, 1 Qudrat, 2 Kreis, 3-4 Dreieck, 5 Raute, 6 Puls, 7 X, 8 dicke Punkte, 9 horz. Linie, 10-11 Dreieck, 12-13 X, 14 gef. Raute, 15 Quadrat, 16-19 Dreiecke, 20-21 Quadrat, 22-23 X, 24 Minus, 25 leer
	"symbol.count" : Symbolanzahl, zu deuten entsprechend Symbolanzahl Option
	"symbol.count.opt" : Symbolanzahl Option. 1: an jedem Sample (Symbolanzahl=0); 2 feste Anzahl über die Breite des Koordinatensystems gezeichnet (Symbolanzahl>0).
	"symbolsize.printer" : Symbolgröße auf dem Drucker in mm (-1 auto)
	"symbolsize.screen" : Symbolgröße auf dem Bildschirm in mm (-1 auto)
	"uncertainty.show" : Messunsicherheit darstellen (0 nein, 1 farbige Fläche, 2 Linie). Siehe Eigenschaft Uncertainty
	"uncertainty.color" : Bei Darstellung der Messunsicherheit wird diese Farbe benutzt. Format siehe rgb() und -1 für automatisch
	"uncertainty.selection" : Messunsicherheit auswählen (0 auto, 1 Standardmessunsicherheit, 2 Erweiterte Messunsicherheit). Siehe Eigenschaften Uncertainty und Expanded uncertainty. Bei auto wird die Standardmessunsicherheit bevorzugt.
	"width.printer" : Liniendicke auf dem Drucker in mm (-1 auto)
	"width.screen" : Liniendicke auf dem Bildschirm in mm (-1 auto)
	"3D.surface" : 3D: Auswahl der Oberfläche (0 auto, 1 Drahtgitter gefüllt, 2 gefüllt, 3 Drahtgitter, 4 Punkte, 5 Drahtgitter in Farbpalette, 6 Raumkurve, 7 Raumkurve entsprechend Farbpalette)
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Abfrage, mit welchem Symbol die Linie gerade dargestellt wird.

```
CwSelectByIndex("line", 1)
symbol = CwLineGet("symbol")
```

Siehe auch:

CwAxisLineSet

CwLineSet

Anwendungsbereich: Kurvenfenster

Eigenschaft einer Linie setzen

Deklaration:

```
CwLineSet ( Eigenschaft, Wert )
```

Parameter:

Eigenschaft	Welche Eigenschaft soll gesetzt werden?
	"type" : Linientyp: 0 keine, 1 Linien, 2 Treppen, 3 Punkte, 4 vert. Linie, 5 Balken, 7 3D-Balken, 8 Interpolation
	"color.printer" : Farbe der Linie auf dem Drucker, Format siehe rgb() und -1 für automatisch
	"color.screen" : Farbe der Linie auf dem Bildschirm, Format siehe rgb() und -1 für automatisch.
	"area.border" : Rand bei Darstellung von Flächen unter Kurven, die selbst nicht Treppen oder Geraden sind (0 auto, 1 Treppen, 2 linear)
	"area.color" : Bei Darstellung von Flächen unter Kurven wird diese Farbe benutzt. Format siehe rgb() und -1 für automatisch
	"area.color2" : Bei Darstellung eines Farbverlaufs ist das die 2. Farbe. Format siehe rgb() und -1 für automatisch
	"area.fill" : Soll die Fläche unter der Kurve gefüllt werden (0 nein, 1 bis y=null, 2 bis unten, 3 innen)
	"area.gradient" : Farbverlauf bei Darstellung von Flächen unter Kurven (0 nein, 1 von oben nach unten, 2 von unten nach oben, 3 von links nach rechts, 4 von rechts nach links)
	"auxiliary" : Linie ist eine Hilfslinie (0 nein, 1 ja)
	"bar.begin.y" : Balken Beginn in y-Richtung (0 auto, 1 Beginn bei 0, 2 Beginn auf Grundfläche, 3 Beginn unter der Grundfläche)
	"bar.color.type" : Balken Farbgebung (0 auto, 1 mit Farbverlauf, 2 einfarbig)
	"bar.place.x" : Balken Anordnung in x-Richtung (0 auto, 1 zwischen einem Messwert und dem nächsten, 2 zentriert um Messwert, 3 bündig am Messwert)
	"bar.place.z" : Balken Anordnung in z-Richtung (0 auto, 1 zwischen einem Messwert und dem nächsten, 2 zentriert um Messwert, 3 bündig am Messwert)
	"bar.width.x" : Balken Breite in x-Richtung, in Prozent angegeben: 1 bis 100, 0 auto
	"bar.width.z" : Balken Breite in z-Richtung, in Prozent angegeben: 1 bis 100, 0 auto
	"color2" : Farbe 2, z.B. bei 3D die Farbe der Oberfläche. Format siehe rgb() und -1 für automatisch.
	"cross section.option" : Soll ein Schnitt von den Daten berechnet werden? (0 nein, 1 x = konstant, 2 y = konstant)
	"cross section.value" : Der Wert, an dem der Schnitt erfolgen soll.
	"effect" : Sollen die Daten (meist bei Farbkarte) z.B. als echte Linie dargestellt werden (0 auto, 1 Überlagerung, 2 obere Grenze, 3 untere Grenze, 4 linke Grenze, 5 rechte Grenze, 6 äußere Grenze, 7 innere Grenze, 8 RGB)
	"label.color" : Farbe der Beschriftung, Format siehe rgb(), (-1 auto)
	"label.font.size" : Beschriftung der Linie Schriftgröße, in pt, z.B. 8
	"label.format" : Zahlenformat der Beschriftung (0 auto, 1 Festkomma, 2 Gleitkomma)
	"label.option" : Sollen die Messpunkte mit ihrem Zahlenwert beschriftet werden. (0 nein, 1 ja)
	"label.placement" : Lage der Beschriftung (0 auto, 1 rechts, 2 rechts oben, 3 rechts unten, 4 links, 5 links oben, 6 links unten, 7 Mitte, 8 Mitte oben, 9 Mitte unten)
	"label.precision" : Präzision der Zahlenwerte der Beschriftung. Bei Fest-,Gleitkomma die Nachkommastellen 0..14, sonst Anzahl der gültigen Ziffern 1..14
	"label.selection" : Werteauswahl für die Beschriftung (0 auto, 1 y, 2 x, 3 Parameter, 4 Betrag, 5 Phase)
	"legend.append" : Namenserweiterung in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.channel" : Kanalnamen in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.comment" : Kommentar in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.group" : Gruppennamen in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.measurement" : Messung in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein, 3 Nummer der selektierten Messung)
	"legend.numerical.format" : Format der Zahlenwerte in der Legende zur Linie (0 auto, 1 Festkomma, 2 Gleitkomma)

	"legend.numerical.precision" : Präzision der Zahlenwerte in der Legende zur Linie. Bei Fest-,Gleitkomma die Nachkommastellen 0..14, sonst Anzahl der gültigen Ziffern 1..14
	"legend.numerical.option" : Zahlenwerte in Legende zur Linie zulassen (0 auto, 1 nein)
	"legend.sample" : Linienprobe in Legende zur Linie zeigen (0 auto, 1 ja, 2 nein)
	"legend.show" : Legende zur Linie zeigen (0 auto, 1 ja, 2 nein, 3 Text mit Platzhaltern)
	"legend.text" : Text, der in der Legende dargestellt wird. Er darf Platzhalter enthalten.
	"linestyle.printer" : Linienart auf dem Drucker (0 auto, 1 durchgezogen, 2 gepunktet, 3 gestrichelt, 4..13 andere Kombinationen)
	"linestyle.screen" : Linienart auf dem Bildschirm (0 auto, 1 durchgezogen, 2 gepunktet, 3 gestrichelt, 4..13 andere Kombinationen)
	"measure.cursor.set" : Der Messcursor wird auf diese Linie gesetzt (1 zur linken Maustaste, 2 zur rechten Maustaste).
	"scale.type" : Skalierung der Linie (0 auto, 1 von 0 bis 1 relativ, 2 von 0 bis 1 mit Offset, 3 in mm relativ, 4 in mm mit Offset, 5 Bild strecken, 6 Bild horizontal anpassen, 7 Bild vertikal anpassen, 8 Bild komplett darstellen, 9 Bild Größe beibehalten)
	"shift.x" : Timeshift in physikalischen Einheiten. Bei linear ein Offset, bei logarithmisch ein Faktor.
	"shift.y" : Amplitudenshift in physikalischen Einheiten. Bei linear ein Offset, bei logarithmisch ein Faktor.
	"suppress" : Linie ausblenden (0 nein Standard, 1 ja)
	"symbol" : Symbol: 0 keins, 1 Qudrat, 2 Kreis, 3-4 Dreieck, 5 Raute, 6 Puls, 7 X, 8 dicke Punkte, 9 horz. Linie, 10-11 Dreieck, 12-13 X, 14 gef. Raute, 15 Quadrat, 16-19 Dreiecke, 20-21 Quadrat, 22-23 X, 24 Minus, 25 leer
	"symbol.count" : Symbolanzahl, zu deuten entsprechend Symbolanzahl Option
	"symbol.count.opt" : Symbolanzahl Option. 1: an jedem Sample (Symbolanzahl=0); 2 feste Anzahl über die Breite des Koordinatensystems gezeichnet (Symbolanzahl>0).
	"symbolsize.printer" : Symbolgröße auf dem Drucker in mm (-1 auto)
	"symbolsize.screen" : Symbolgröße auf dem Bildschirm in mm (-1 auto)
	"uncertainty.color" : Bei Darstellung der Messunsicherheit wird diese Farbe benutzt. Format siehe rgb() und -1 für automatisch
	"uncertainty.show" : Messunsicherheit darstellen (0 nein, 1 farbige Fläche, 2 Linie). Siehe Eigenschaft Uncertainty
	"uncertainty.selection" : Messunsicherheit auswählen (0 auto, 1 Standardmessunsicherheit, 2 Erweiterte Messunsicherheit). Siehe Eigenschaften Uncertainty und Expanded uncertainty. Bei auto wird die Standardmessunsicherheit bevorzugt.
	"width.printer" : Liniendicke auf dem Drucker in mm (-1 auto)
	"width.printer.pt" : Liniendicke auf dem Drucker in pt (-1 auto)
	"width.screen" : Liniendicke auf dem Bildschirm in mm (-1 auto)
	"width.screen.pt" : Liniendicke auf dem Bildschirm in pt (-1 auto)
	"3D.surface" : 3D: Auswahl der Oberfläche (0 auto, 1 Drahtgitter gefüllt, 2 gefüllt, 3 Drahtgitter, 4 Punkte, 5 Drahtgitter in Farbpalette, 6 Raumkurve, 7 Raumkurve entsprechend Farbpalette)
Wert	Auf welchen Wert soll diese Eigenschaft gesetzt werden?

Beschreibung:

Beispiele:

Treppen einstellen

```
CwSelectByIndex("line", 1)
CwLineSet("style", 2)
```

Siehe auch:

CwAxisLineGet

CwLoadCCV

Anwendungsbereich: Kurvenfenster

Lädt die Kurvenkonfiguration aus einer *.ccv-Datei.

Deklaration:

```
CwLoadCCV ( Identifikation, Dateiname ) -> Fehlertext
```

Parameter:

Identifikation	Dieser Datensatz identifiziert das Kurvenfenster. Mit CwSelectMode wurde festgelegt, wie die Identifikation erfolgt.
Dateiname	Aus welcher Datei soll die Konfiguration geladen werden?
Fehlertext	Rückgabe ist ein Fehlertext, im Erfolgsfall ein leerer Text (optional)

Beschreibung:

Die Funktion erzeugt das Kurvenfenster neu, falls bislang noch kein Kurvenfenster zur angegebenen Identifikation vorhanden ist.

Die Funktion selektiert auch gleich das Kurvenfenster.

CwSelectWindow() braucht im Anschluss nicht aufgerufen zu werden, sondern erst später zum Selektieren dieses Kurvenfensters nach dem Selektieren von anderen Kurvenfenstern.

Ist die Identifikation ein Text, gilt: Wenn ein Kurvenfenster-Titel im geöffneten Panel oder Dialog vorhanden ist, wird dieses Kurvenfenster adressiert. Wenn nicht vorhanden, aber ein frei fliegendes Kurvenfenster mit dieser Identifikation existiert, wird dieses angesprochen. Sonst wird ein neues frei fliegendes Kurvenfenster erzeugt.

Ist die Identifikation ein Datensatz, wird ein frei fliegendes Kurvenfenster angesprochen bzw. erzeugt.

Die Konfiguration des Kurvenfensters enthält alle Attribute der Darstellung, aber nicht die dargestellten Messdaten selbst.

Das Verhalten der Funktion kann über einen vorherigen Aufruf der Funktion CwGlobalSet beeinflusst werden.

Beispiele:

Laden einer CCV-Datei

```
data = rampe(0,1,10)
CwLoadCCV(data, "c:\imc\ccv\1.ccv")
```

Laden einer CCV-Datei für ein eingebettetes Kurvenfenster im Panel. Die CCV Datei liegt im Projektverzeichnis.

```
CwLoadCCV("curve1", "1.ccv")
```

Falls im Panel mehrere Kurvenfenster denselben Titel haben, muss der Seitenname (hier "page1") mit Punkt getrennt vorangestellt werden.

```
CwLoadCCV("page1.curve1", "1.ccv")
```

Laden einer CCV-Datei für ein eingebettetes Kurvenfenster im Dialog

```
CwLoadCCV("curve1", "c:\imc\ccv\1.ccv")
```

Laden aus dem CCV-Verzeichnis oder aus dem Projekt-Verzeichnis

```
CwLoadCCV(data, "1.ccv")
```

Laden mit Fehlerabfrage

```
errortext = CwLoadCCV(data, "1.ccv")
wenn errortext <> ""
   ; ...
ende
```

Laden einer CCV-Datei, Identifikation über Variable

```
CwLoadCCV(data, "1.ccv")
...
CwSelectWindow(data)
```

Laden einer CCV-Datei, Identifikation mit Text

```
CwLoadCCV("t", "c:\imc\ccv\1.ccv")
...
CwSelectWindow("t")
```

Laden einer CCV-Datei, Erzeugen eines frei fliegenden Fensters ohne Identifikation

Im Anschluss ist das Fenster selektiert. Es kann später i. Allg. nicht mehr selektiert werden, nachdem ein anderes Kurvenfenster selektiert

wurde.

```
CwLoadCCV(0, "1.ccv")
```

Siehe auch:

CwSaveCCV

CwLoadSettings

Anwendungsbereich: Kurvenfenster

Die Funktion lädt eine globale Einstellung in den Kurvenmanager.

Deklaration:

```
CwLoadSettings ( TXDateiname, EWEinstellung, EWParameter )
```

Parameter:

TXDateiname	TXDateiname
EWEinstellung	EWEinstellung
	1 : Alle Kurvenfenster werden ab sofort auf dem Bildschirm in den neuen Farben dargestellt. Dazu gehören alle Einstellungen des Farben-Dialogs.
	2 : Für alle folgenden Druckvorgänge, das Übertragen in den Reportgenerator das Kopieren in die Ablage und den Grafikexport werden die neuen Farben übernommen.
	3 : Alle Einstellungen des Dialogs <Einstellungen Ablage> werden ab sofort gültig. Für alle folgenden Druckvorgänge, das Übertragen in den Reportgenerator, das Kopieren in die Ablage und den Grafikexport werden diese Einstellungen benutzt. Allerdings bleiben davon Reports unberührt, bei denen die aktuellen Einstellungen beim Transfer nicht übernommen werden sollen.
	4 : Einige Einstellungen des Dialogs <Kurvenfenster, Voreinstellungen...> werden geladen. Diese Einstellungen gelten für alle Kurvenfenster gemeinsam. Dazu zählen die Schrift- und Symbolgröße auf dem Bildschirm.
EWParameter	EWParameter

Beschreibung:

Einstellungen werden aus einer Datei mit dem Namen TXDateiname geladen. Die Datei wird standardmäßig im CCV-Verzeichnis bei imc FAMOS erwartet. Einige der Optionen haben direkte Auswirkungen auf alle angezeigten Kurvenfenster. Wenn nicht anders vermerkt, dann EWParameter = 0 setzen.

Die Funktion liefert keine Fehlermeldung, falls die Datei unvollständig ist oder gar überhaupt nicht existiert.

Beispiele:

Die Datei c:\imc\set\colors.set wurde vom Kurvenfenster aus gespeichert.

```
CwLoadSettings ( "..\set\colors.set", 1, 0 )
```

Siehe auch:

CwGlobalSet

CwMarkerGet

Anwendungsbereich: Kurvenfenster

Eigenschaft eines Markers abfragen

Deklaration:

```
CwMarkerGet ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"x.type" : Art der x-Koordinate (1 physikalische Einheit, 2 Prozent der Achsenlänge)
	"x" : x-Koordinate, deren Bedeutung mit "Art der x-Koordinate" festgelegt ist
	"y.type" : Art der y-Koordinate (1 physikalische Einheit, 2 Prozent der Achsenlänge)
	"y" : y-Koordinate, deren Bedeutung mit "Art der x-Koordinate" festgelegt ist
	"angle" : Winkel der Verbindungslinie in Grad (-360 .. +360). Nicht gültig bei "Art der Linienlänge" = "XY prozentual".
	"arrow" : Art des Pfeils (0 keiner, 1 breit, 2 schmal, 3 breit voll, 4 schmal voll, 5 groß, 6 groß voll 7 Kreis, 8 Stern, 9 schräger Strich, 10 Punkt)
	"arrow.size" : Größe des Pfeils in mm (0.5 .. 10.0), 0 für auto
	"border" : Rahmen um den Text (0 keiner, 1 einfach, 2 mit Spitze, 3 doppelt)
	"calculation" : Bei Ordnungsline die Berechnung (0 Ordungslinie im Drehzahlspektrum, 1 RPM über Frequenz, 2 Frequenz über RPM, 3 Drehfrequenz über Frequenz, 4 Frequenz über Drehfrequenz, 5 Hyperbel im Ordnungsspektrum, 6 RPM und Ordnung, 7 Drehfrequenz und Ordnung)
	"color.background" : Farbe des Hintergrundes, Format siehe rgb(), -1 für automatisch, -2 für transparent
	"color.text" : Farbe des Textes, Format siehe rgb() und -1 für automatisch
	"dimline.arrow.position" : Maßlinie: Position Pfeil (0 auto, 1 außen)
	"dimline.dimension line.extend" : Maßlinie: Maßlinie verlängern in mm, >=0
	"dimline.distance" : Maßlinie: Abstand der Maßlinie vom Marker: In mm von einem der beiden Bezugsmarker. Oder relativ als Anteil der Achsenlänge, z.B. 0.1 für 10% der Achsenlänge nach rechts oder unten. Oder fest: 0..1 für den Bereich links vom bzw. über dem Koordinatensystem, 1..2 innerhalb, 2..3 rechts vom bzw. unter dem Koordinatensystem.
	"dimline.distance type" : Maßlinie: Wie wird der Abstand angegeben? (0 fest, 1 relativ ab 1. Bezugsmarker, 2 mm ab 1. Bezugsmarker, 3 relativ ab 2. Bezugsmarker, 4 mm ab 2. Bezugsmarker)
	"dimline.projection line.distance" : Maßlinie: Maßhilfslinie Abstand in mm, >=0
	"dimline.projection line.extend" : Maßlinie: Maßhilfslinie verlängern in mm, >=0
	"extension" : Bei horizontaler/vertikaler Linie die Ausdehnung (0 alle Koordinatensysteme, 1 vom Marker nach links/unten, 2 vom Marker bis freie Position, 3 zwischen 2 freien Positionen, 4 über ein Koordinatensytem)
	"font.size" : Größe der Schrift für den Markertext, in pt, z.B. 8.
	"harmonic.type" : Falls der Marker ein harmonischer Cursor ist, der Typ (0 kein harmonischer Cursor, 1 Harmonische einer Grundschwingung, 2 Offset erste Schwingung, weitere sind periodisch, 3 eine oder mehrere Harmonische mit Seitenbändern, 4 zwei Linien, deren Abstand ein Faktor ist)
	"harmonic.index" : Falls der Marker ein harmonischer Cursor ist, ein Index zum Unterscheiden der einzelnen Cursoren
	"index" : Index des Markers. Der Index auf die Liste der Marker beginnt bei 1.
	"line.end" : Bei Ordnungsline das Ende der Linie in Prozent (0 .. 100)
	"line.index" : Index der zugeordneten Linie (Linienelement, nicht Verbindungslinie!), >= 1 bei gültiger Linie, sonst 0
	"line.start" : Bei Ordnungsline der Start der Linie in Prozent (0 .. 100)
	"line.text.pos" : Bei Ordnungsline die Position der Beschriftung entlang der Linie in Prozent
	"linelength" : Länge der Linie zwischen Markerpunkt und Markertext. Die Bedeutung der Länge ist mit "Art der x-Linienlänge" festgelegt ist. Bei "Art der Linienlänge" = "XY prozentual" wird hier die Länge der Linie in x-Richtung angegeben.
	"linelength.type" : Art der Linienlänge (1 x-Einheiten, 2 % der x-Achse, 3 y-Einheiten, 4 % der y-Achse, 5 % der Texthöhe, 6 XY prozentual: % x-Achse und Linienlänge2 % der y-Achse)
	"linelength2" : Länge der Linie in y-Richtung, nur bei "Art der Linienlänge" = "XY prozentual".

	"linestyle" : Linienart (0 auto, 1 durchgezogen, 2 gepunktet, 3 gestrichelt, 4..13 andere Kombinationen)
	"linewidth" : Liniendicke in mm (0 auto)
	"multiple" : Bei Ordnungsline kann der Parameter nur Vielfache dieses Wertes annehmen. Falls gleich null, dann keine Einschränkung
	"parameter" : Bei Ordnungsline der Parameter, Bedeutung abhängig von der Berechnung. Die Ordnung bei Ordnungsline, die Frequenz bei Hyperbel.
	"pinnedtext" : Textanschluss. In welche Richtung soll sich die Textbox erstrecken. Typisch 0. (0=auto, 1 rechts oben, 2 links oben, 3 rechts unten, 4 links unten)
	"ref.1" : Index des 1. Referenzmarkers einer Maßlinie. Der Index auf die Liste der Marker beginnt bei 1 und wählt einen Standard-Marker aus.
	"ref.2" : Index des 2. Referenzmarkers einer Maßlinie. Der Index auf die Liste der Marker beginnt bei 1 und wählt einen Standard-Marker aus.
	"text.orientation" : Neigung des Textes in Grad (-90 .. +90). Wie stark ist der Text gegen die Horizontale geneigt?
	"type" : Typ des Markers (0 Standard, 1 vertikale Linie, 2 horizontale Linie, 3 Text, 4 vertikale Maßlinie, 5 horizontale Maßlinie)
	"value.abs" : Maßlinie: Absolutbetrag der Differenz zum Beschriften nutzen(0 nein, 1 ja)
	"within.cosys" : Wird der Marker nur innerhalb des Koordindatensystems gezeichnet (0 nein, 1 ja)
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

```
mini = CwMarkerGet("min")
```

Siehe auch:

CwMarkerSet, CwMarkerGetText

CwMarkerGetText

Anwendungsbereich: Kurvenfenster

Text-Eigenschaft eines Markers abfragen

Deklaration:

```
CwMarkerGetText ( Eigenschaft ) -> Wert
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"text" : Text
	"text.placeholders" : Text mit Platzhaltern
Wert	Der Wert der Eigenschaft

Beschreibung:

Beispiele:

Text eines Markers abfragen

```
CwSelectByIndex("marker", 1)
text = CwMarkerGet("text")
```

Siehe auch:

CwMarkerSet, CwMarkerGet

CwMarkerSet

Anwendungsbereich: Kurvenfenster

Eigenschaft eines Markers setzen

Deklaration:

```
CwMarkerSet ( Eigenschaft, Wert )
```

Parameter:

Eigenschaft	Welche Eigenschaft soll gesetzt werden?
	"text" : Text
	"x.type" : Art der x-Koordinate (1 physikalische Einheit, 2 Prozent der Achsenlänge)
	"x" : x-Koordinate, deren Bedeutung mit "Art der x-Koordinate" festgelegt ist
	"y.type" : Art der y-Koordinate (1 physikalische Einheit, 2 Prozent der Achsenlänge)
	"y" : y-Koordinate, deren Bedeutung mit "Art der x-Koordinate" festgelegt ist
	"angle" : Winkel der Verbindungslinie in Grad (-360 .. +360). Nicht gültig bei "Art der Linienlänge" = "XY prozentual".
	"arrow" : Art des Pfeils (0 keiner, 1 breit, 2 schmal, 3 breit voll, 4 schmal voll, 5 groß, 6 groß voll 7 Kreis, 8 Stern, 9 schräger Strich, 10 Punkt)
	"arrow.size" : Größe des Pfeils in mm (0.5 .. 10.0), 0 für auto
	"border" : Rahmen um den Text (0 keiner, 1 einfach, 2 mit Spitze, 3 doppelt)
	"calculation" : Bei Ordnungsline die Berechnung (0 Ordungslinie im Drehzahlspektrum, 1 RPM über Frequenz, 2 Frequenz über RPM, 3 Drehfrequenz über Frequenz, 4 Frequenz über Drehfrequenz, 5 Hyperbel im Ordnungsspektrum, 6 RPM und Ordnung, 7 Drehfrequenz und Ordnung)
	"color.background" : Farbe des Hintergrundes, Format siehe rgb(), -1 für automatisch, -2 für transparent
	"color.text" : Farbe des Textes, Format siehe rgb() und -1 für automatisch
	"dimline.arrow.position" : Maßlinie: Position Pfeil (0 auto, 1 außen)
	"dimline.dimension line.extend" : Maßlinie: Maßlinie verlängern in mm, >=0
	"dimline.distance" : Maßlinie: Abstand der Maßlinie vom Marker: In mm von einem der beiden Bezugsmarker. Oder relativ als Anteil der Achsenlänge, z.B. 0.1 für 10% der Achsenlänge nach rechts oder unten. Oder fest: 0..1 für den Bereich links vom bzw. über dem Koordinatensystem, 1..2 innerhalb, 2..3 rechts vom bzw. unter dem Koordinatensystem.
	"dimline.distance type" : Maßlinie: Wie wird der Abstand angegeben? (0 fest, 1 relativ ab 1. Bezugsmarker, 2 mm ab 1. Bezugsmarker, 3 relativ ab 2. Bezugsmarker, 4 mm ab 2. Bezugsmarker)
	"dimline.projection line.distance" : Maßlinie: Maßhilfslinie Abstand in mm, >=0
	"dimline.projection line.extend" : Maßlinie: Maßhilfslinie verlängern in mm, >=0
	"extension" : Bei horizontaler/vertikaler Linie die Ausdehnung (0 alle Koordinatensysteme, 1 vom Marker nach links/unten, 2 vom Marker bis freie Position, 3 zwischen 2 freien Positionen, 4 über ein Koordinatensytem)
	"font.size" : Größe der Schrift für den Markertext, in pt, z.B. 8.
	"line.end" : Bei Ordnungsline das Ende der Linie in Prozent (0 .. 100)
	"line.selected" : Der Marker wird der selektierten Linie (Linienelement, nicht Verbindungslinie!) zugeordnet. Wert = 1.
	"line.start" : Bei Ordnungsline der Start der Linie in Prozent (0 .. 100)
	"line.text.pos" : Bei Ordnungsline die Position der Beschriftung entlang der Linie in Prozent
	"linelength" : Länge der Linie zwischen Markerpunkt und Markertext. Die Bedeutung der Länge ist mit "Art der x-Linienlänge" festgelegt ist. Bei "Art der Linienlänge" = "XY prozentual" wird hier die Länge der Linie in x-Richtung angegeben.
	"linelength.type" : Art der Linienlänge (1 x-Einheiten, 2 % der x-Achse, 3 y-Einheiten, 4 % der y-Achse, 5 % der Texthöhe, 6 XY prozentual: % x-Achse und Linienlänge2 % der y-Achse)
	"linelength2" : Länge der Linie in y-Richtung, nur bei "Art der Linienlänge" = "XY prozentual".
	"linestyle" : Linienart (0 auto, 1 durchgezogen, 2 gepunktet, 3 gestrichelt, 4..13 andere Kombinationen)
	"linewidth" : Liniendicke in mm (0 auto)
	"multiple" : Bei Ordnungsline kann der Parameter nur Vielfache dieses Wertes annehmen. Falls gleich null, dann keine Einschränkung

	"parameter" : Bei Ordnungsline der Parameter, Bedeutung abhängig von der Berechnung. Die Ordnung bei Ordnungsline, die Frequenz bei Hyperbel.
	"pinnedtext" : Textanschluss. In welche Richtung soll sich die Textbox erstrecken. Typisch 0. (0=auto, 1 rechts oben, 2 links oben, 3 rechts unten, 4 links unten)
	"ref.1" : Index des 1. Referenzmarkers einer Maßlinie. Der Index auf die Liste der Marker beginnt bei 1 und wählt einen Standard-Marker aus.
	"ref.2" : Index des 2. Referenzmarkers einer Maßlinie. Der Index auf die Liste der Marker beginnt bei 1 und wählt einen Standard-Marker aus.
	"text.orientation" : Neigung des Textes in Grad (-90 .. +90). Wie stark ist der Text gegen die Horizontale geneigt?
	"within.cosys" : Wird der Marker nur innerhalb des Koordindatensystems gezeichnet (0 nein, 1 ja)
	"value.abs" : Maßlinie: Absolutbetrag der Differenz zum Beschriften nutzen(0 nein, 1 ja)
Wert	Auf welchen Wert soll diese Eigenschaft gesetzt werden?

Beschreibung:

Beispiele:

Den Text eines Markers setzen

```
CwSelectByIndex("marker", 1)
CwMarkerSet("text", "Max!")
```

Den Text eines Markers mit Platzhaltern setzen

```
CwSelectByIndex("marker", 1)
CwMarkerSet("<auto> Left!", "Max!")
```

Siehe auch:

CwMarkerGet, CwMarkerGetText, CwNewElement

CwNewChannel

Anwendungsbereich: Kurvenfenster

Ein Kanal wird im Kurvenfenster dargestellt.

Deklaration:

```
CwNewChannel ( Position, Kanal )
```

Parameter:

Position	An welcher Stelle sollen die neuen Daten eingefügt werden?
	"append last axis" : Die neuen Daten werden mit einer neuen Linie an die letzte vorhandene y-Achse angehängt. Die neue Linie wird selektiert.
	"append line" : Hinter der selektierten Linie wird eine neue Linie angehängt, der die neuen Daten zugewiesen werden. Die neue Linie wird selektiert.
	"append new axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse an das letzte Koordinatensystem angehängt. Die neue Linie und die neue Achse werden selektiert.
	"append new cosys" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse in ein neu angehängtes Koordinatensystem eingefügt. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"append to cosys" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird als letzte Achse an das vorher selektierte Koordinatensystem angehängt. Die neue Achse und die neue Linie werden selektiert.
	"first axis" : Die neuen Daten werden mit einer neuen Linie an erster Stelle mit einer eigenen Achse eingefügt. Die neue Linie samt Achse und Koordinatensystem wird selektiert.
	"first line" : Die neuen Daten werden mit einer neuen Linie an erster Stelle eingefügt. Die neue Linie samt Achse und Koordinatensystem wird selektiert.
	"insert axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird hinter der vorher selektierten y-Achse eingefügt. Die neue Achse und die neue Linie werden selektiert.
	"insert cosys" : Die neuen Daten werden mit einer neuen Linie, einer neuen y-Achse und einem neuen Koordinatensystem eingefügt. Das neue Koordinatensystem wird hinter dem vorher selektierten Koordinatensystem eingefügt. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"insert first axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird als erste Achse in das vorher selektierte Koordinatensystem eingefügt. Die neue Achse und die neue Linie werden selektiert.
	"insert first cosys" : Die neuen Daten werden mit einer neuen Linie, einer neuen y-Achse und einem neuen Koordinatensystem eingefügt. Das neue Koordinatensystem ist das erste. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"insert first line" : Die neuen Daten werden mit einer neuen Linie, die die erste zur vorher selektierten Achse sein wird, eingefügt. Die neue Linie wird selektiert.
	"insert line" : Vor der selektierten Linie wird eine neue Linie eingefügt, der die neuen Daten zugewiesen werden. Die neue Linie wird selektiert.
	"replace data" : Das selektierte Datenelement erhält den neuen Kanal.
	"replace line" : Die Daten zur selektierten Linie werden durch die neuen Daten ersetzt.
Kanal	Datensatz, der im Kurvenfenster dargestellt werden soll.

Beschreibung:

Je nach Darstellungsart werden ggf. keine neuen Achsen oder Koordinatensysteme angelegt.

Beispiele:

Kanäle einem Kurvenfenster hinzufügen

```
CwSelectWindow("curve1")
CwNewChannel("append last axis", channel1)
CwNewChannel("append last axis", channel2)
```

Den Betrag eines Kanals hinzufügen

```
CwSelectWindow("curve1")
CwNewChannel("append new axis", Spectrum.b)
```

Einen dargestellten Kanal durch einen anderen ersetzen

```
CwSelectWindow("curve1")
CwSelectByIndex("line", 1)
CwNewChannel("replace line", channel2)
```

Siehe auch:

CwNewChannel_xy, CwNewChannel_xyz

CwNewChannel_xy

Anwendungsbereich: Kurvenfenster

Zwei Kanäle werden als xy-Darstellung im Kurvenfenster dargestellt.

Deklaration:

```
CwNewChannel_xy ( Position, Kanal 1, Kanal 2, Option )
```

Parameter:

Position	An welcher Stelle sollen die neuen Daten eingefügt werden?
	"append last axis" : Die neuen Daten werden mit einer neuen Linie an die letzte vorhandene y-Achse angehängt. Die neue Linie wird selektiert.
	"append line" : Hinter der selektierten Linie wird eine neue Linie angehängt, der die neuen Daten zugewiesen werden. Die neue Linie wird selektiert.
	"append new axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse an das letzte Koordinatensystem angehängt. Die neue Linie und die neue Achse werden selektiert.
	"append new cosys" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse in ein neu angehängtes Koordinatensystem eingefügt. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"append to cosys" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird als letzte Achse an das vorher selektierte Koordinatensystem angehängt. Die neue Achse und die neue Linie werden selektiert.
	"first axis" : Die neuen Daten werden mit einer neuen Linie an erster Stelle mit einer eigenen Achse eingefügt. Die neue Linie samt Achse und Koordinatensystem wird selektiert.
	"first line" : Die neuen Daten werden mit einer neuen Linie an erster Stelle eingefügt. Die neue Linie samt Achse und Koordinatensystem wird selektiert.
	"insert axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird hinter der vorher selektierten y-Achse eingefügt. Die neue Achse und die neue Linie werden selektiert.
	"insert cosys" : Die neuen Daten werden mit einer neuen Linie, einer neuen y-Achse und einem neuen Koordinatensystem eingefügt. Das neue Koordinatensystem wird hinter dem vorher selektierten Koordinatensystem eingefügt. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"insert first axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird als erste Achse in das vorher selektierte Koordinatensystem eingefügt. Die neue Achse und die neue Linie werden selektiert.
	"insert first cosys" : Die neuen Daten werden mit einer neuen Linie, einer neuen y-Achse und einem neuen Koordinatensystem eingefügt. Das neue Koordinatensystem ist das erste. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"insert first line" : Die neuen Daten werden mit einer neuen Linie, die die erste zur vorher selektierten Achse sein wird, eingefügt. Die neue Linie wird selektiert.
	"insert line" : Vor der selektierten Linie wird eine neue Linie eingefügt, der die neuen Daten zugewiesen werden. Die neue Linie wird selektiert.
	"replace line" : Die Daten zur selektierten Linie werden durch die neuen Daten ersetzt.
Kanal 1	1. Kanal, der im Kurvenfenster dargestellt werden soll.
Kanal 2	2. Kanal, der im Kurvenfenster dargestellt werden soll.
Option	Option
	"yx" : 1. Kanal wird y-Komponente, 2. Kanal x-Komponente.
	"xy" : 1. Kanal wird x-Komponente, 2. Kanal y-Komponente.

Beschreibung:

Je nach Darstellungsart werden ggf. keine neuen Achsen oder Koordinatensysteme angelegt.

Beispiele:

Kanäle einem Kurvenfenster hinzufügen

```
CwSelectWindow("curve1")
CwNewChannel_xy("append last axis", level, speed, "yx")
CwNewChannel_xy("append last axis", temperature, speed, "yx")
```

Einen dargestellten Kanal durch eine neue xy-Darstellung ersetzen

```
CwSelectWindow("curve1")
CwSelectByIndex("line", 1)
CwNewChannel_xy("replace line",  level, speed, "yx")
```

Siehe auch:

CwNewChannel, CwNewChannel_xyz

CwNewChannel_xyz

Anwendungsbereich: Kurvenfenster

Drei Kanäle werden als xyz-Darstellung im Kurvenfenster dargestellt.

Deklaration:

```
CwNewChannel_xyz ( Position, Kanal 1, Kanal 2, Kanal 3, Option )
```

Parameter:

Position	An welcher Stelle sollen die neuen Daten eingefügt werden?
	"append last axis" : Die neuen Daten werden mit einer neuen Linie an die letzte vorhandene y-Achse angehängt. Die neue Linie wird selektiert.
	"append line" : Hinter der selektierten Linie wird eine neue Linie angehängt, der die neuen Daten zugewiesen werden. Die neue Linie wird selektiert.
	"append new axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse an das letzte Koordinatensystem angehängt. Die neue Linie und die neue Achse werden selektiert.
	"append new cosys" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse in ein neu angehängtes Koordinatensystem eingefügt. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"append to cosys" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird als letzte Achse an das vorher selektierte Koordinatensystem angehängt. Die neue Achse und die neue Linie werden selektiert.
	"first axis" : Die neuen Daten werden mit einer neuen Linie an erster Stelle mit einer eigenen Achse eingefügt. Die neue Linie samt Achse und Koordinatensystem wird selektiert.
	"first line" : Die neuen Daten werden mit einer neuen Linie an erster Stelle eingefügt. Die neue Linie samt Achse und Koordinatensystem wird selektiert.
	"insert axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird hinter der vorher selektierten y-Achse eingefügt. Die neue Achse und die neue Linie werden selektiert.
	"insert cosys" : Die neuen Daten werden mit einer neuen Linie, einer neuen y-Achse und einem neuen Koordinatensystem eingefügt. Das neue Koordinatensystem wird hinter dem vorher selektierten Koordinatensystem eingefügt. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"insert first axis" : Die neuen Daten werden mit einer neuen Linie und einer neuen y-Achse eingefügt. Die neue y-Achse wird als erste Achse in das vorher selektierte Koordinatensystem eingefügt. Die neue Achse und die neue Linie werden selektiert.
	"insert first cosys" : Die neuen Daten werden mit einer neuen Linie, einer neuen y-Achse und einem neuen Koordinatensystem eingefügt. Das neue Koordinatensystem ist das erste. Das neue Koordinatensystem, die neue Achse und die neue Linie werden selektiert.
	"insert first line" : Die neuen Daten werden mit einer neuen Linie, die die erste zur vorher selektierten Achse sein wird, eingefügt. Die neue Linie wird selektiert.
	"insert line" : Vor der selektierten Linie wird eine neue Linie eingefügt, der die neuen Daten zugewiesen werden. Die neue Linie wird selektiert.
	"replace line" : Die Daten zur selektierten Linie werden durch die neuen Daten ersetzt.
Kanal 1	1. Kanal, der im Kurvenfenster dargestellt werden soll.
Kanal 2	2. Kanal, der im Kurvenfenster dargestellt werden soll.
Kanal 3	3. Kanal, der im Kurvenfenster dargestellt werden soll.
Option	
	"yxz" : 1. Kanal wird y-Komponente, 2. Kanal x-Komponente, 3. Kanal z-Komponente.
	"xyz" : 1. Kanal wird x-Komponente, 2. Kanal y-Komponente, 3. Kanal z-Komponente.

Beschreibung:

Je nach Darstellungsart werden ggf. keine neuen Achsen oder Koordinatensysteme angelegt.

Nicht alle Darstellungsarten unterstützen diese Art der Überlagerung.

Beispiele:

Kanäle einem Kurvenfenster hinzufügen

```
CwSelectWindow("curve1")
CwNewChannel_xyz("append last axis", level, speed, torque, "yxz")
```

Einen dargestellten Kanal durch eine neue Überlagerung aus 3 Kanälen ersetzen

```
CwSelectWindow("curve1")
CwSelectByIndex("line", 1)
CwNewChannel_xyz("append last axis", level, speed, torque, "yxz")
```

Siehe auch:

CwNewChannel, CwNewChannel_xy

CwNewElement

Anwendungsbereich: Kurvenfenster

Ein neues Element im Kurvenfenster wird angelegt.

Deklaration:

```
CwNewElement ( Elementsorte )
```

Parameter:

Elementsorte	Welche Sorte von Element an welcher Position angelegt werden?
	"marker" : Ein neuer Marker wird hinten an die Liste der Marker angehängt.
	"marker.abs" : Ein neuer Marker wird hinten an die Liste der Marker angehängt. Seine Position wird durch x- und y-Koordinaten bestimmt.
	"marker.user" : Ein neuer Marker wird hinten an die Liste der Marker angehängt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.harmonic.harmonic" : Alle Marker, die zu einem harmonischen Cursor für Grundschwingung mit Harmonischen gehören, werden erzeugt.
	"marker.harmonic.harmonic.user" : Alle Marker, die zu einem harmonischen Cursor für Grundschwingung mit Harmonischen gehören, werden erzeugt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.harmonic.offset" : Alle Marker, die zu einem harmonischen Cursor für periodische Vorgänge mit beliebigem Start gehören, werden erzeugt.
	"marker.harmonic.offset.user" : Alle Marker, die zu einem harmonischen Cursor für periodische Vorgänge mit beliebigem Start gehören, werden erzeugt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.harmonic.ratio" : Alle Marker, die zu einem harmonischen Cursor für 2 Schwingungen in festem Verhältnis gehören, werden erzeugt.
	"marker.harmonic.ratio.user" : Alle Marker, die zu einem harmonischen Cursor für 2 Schwingungen in festem Verhältnis gehören, werden erzeugt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.harmonic.sideband" : Alle Marker, die zu einem harmonischen Cursor für Grundschwingung mit Seitenbändern gehören, werden erzeugt.
	"marker.harmonic.sideband.user" : Alle Marker, die zu einem harmonischen Cursor für Grundschwingung mit Seitenbändern gehören, werden erzeugt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.hori.dimline" : Ein neuer Marker in der Form einer horizontalen Maßlinie wird hinten an die Liste der Marker angehängt.
	"marker.hori.dimline.user" : Ein neuer Marker in der Form einer horizontalen Maßlinie wird hinten an die Liste der Marker angehängt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.hori.line" : Ein neuer Marker in der Form einer horizontalen Linie wird hinten an die Liste der Marker angehängt.
	"marker.hori.line.user" : Ein neuer Marker in der Form einer horizontalen Linie wird hinten an die Liste der Marker angehängt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.text" : Ein neuer Marker in der Form eines Textes wird hinten an die Liste der Marker angehängt.
	"marker.text.user" : Ein neuer Marker in der Form eines Textes wird hinten an die Liste der Marker angehängt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.vert.dimline" : Ein neuer Marker in der Form einer vertikalen Maßlinie wird hinten an die Liste der Marker angehängt.
	"marker.vert.dimline.user" : Ein neuer Marker in der Form einer vertikalen Maßlinie wird hinten an die Liste der Marker angehängt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.vert.line" : Ein neuer Marker in der Form einer vertikalen Linie wird hinten an die Liste der Marker angehängt.
	"marker.vert.line.user" : Ein neuer Marker in der Form einer vertikalen Linie wird hinten an die Liste der Marker angehängt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.
	"marker.order" : Ein neuer Marker in der Form einer Ordnungslinie wird hinten an die Liste der Marker angehängt.
	"marker.order.user" : Ein neuer Marker in der Form einer Ordnungslinie wird hinten an die Liste der Marker angehängt. Die aktuellen Voreinstellungen am Kurvenfenster werden beachtet.

Beschreibung:

Das neu angelegte Element wird auch gleich selektiert

Beispiele:

Einen neuen Marker dem Kurvenfenster hinzufügen und parametrieren

```
CwNewElement("marker")
CwMarkerSet("text", "Max!")
```

Siehe auch:

CwSelectByIndex, CwMarkerSet

CwNewWindow

Anwendungsbereich: Kurvenfenster

Erzeugt ein leeres Kurvenfenster.

Deklaration:

```
CwNewWindow ( Identifikation, ShowOption )
```

Parameter:

Identifikation	Dieser Datensatz identifiziert das Kurvenfenster. Damit soll später das Kurvenfenster wieder identifizierbar sein.
ShowOption	Soll das Kurvenfenster gleich angezeigt werden?
	"show" : anzeigen
	"hide" : Nicht anzeigen.

Beschreibung:

Ein frei fliegendes leeres Kurvenfenster wird erzeugt. Nur falls zu der angegebenen Identifikation noch kein Kurvenfenster existiert.

Die Funktion selektiert auch gleich das Kurvenfenster.

Wird das Fenster versteckt erzeugt, muss es für manche Operation wirklich sichtbar dargestellt werden.

Existiert zu der angegebenen Identifikation bereits ein Kurvenfenster, wird der Parameter ShowOption nicht angewendet: Das Fenster bleibt unverändert.

Beispiele:

Ein leeres Kurvenfenster anzeigen, dann einen Kanal darin darstellen

```
data=rampe(0,1,10)
CwNewWindow(data, "show")
CwNewChannel("append new axis",data)
```

Siehe auch:

CwLoadCCV

CwPosition

Anwendungsbereich: Kurvenfenster

Einstellung von Position und Größe des selektierten Kurvenfensters

Deklaration:

```
CwPosition ( X, Y, dX, dY )
```

Parameter:

X	X
Y	Y
dX	dX
dY	dY

Beschreibung:

EWX, EWY - Obere, linke Ecke des Fensters

EWdX, EWdY - Größe des Fensters

Beispiele:

```
CwSelectWindow("curve1")
CwPosition ( 0, 0, 640, 480 )
```

Siehe auch:

CwLoadCCV

CwPrintSet

Anwendungsbereich: Kurvenfenster

Eigenschaften für das Drucken eines Kurvenfensters setzen

Deklaration:

```
CwPrintSet ( Eigenschaft, Wert )
```

Parameter:

Eigenschaft	Welche Eigenschaft soll gesetzt werden?
	"individual" : Einstellungen individuell: 0 nein, 1 ja
	"layout" : Layout: 0 Größe Koordinatensystem, 1 Proportionen wie auf Schirm, 2 gesamte Größe
	"cosys.open" : Koordinatensystem offen: 0 nein, 1 ja
	"signature.show" : Unterschrift zeigen: 0 nein, 1 ja
	"signature" : Unterschrift
	"measure.show" : Messcursor und Messwerte zeigen: 0 nein, 1 ja
	"width" : Breite in mm. Je nach Layout für das Koordinatensystem oder gesamt
	"height" : Höhe in mm. Je nach Layout für das Koordinatensystem oder gesamt
	"zlength" : Länge der z-Achse bei 3D in mm
	"symbol.size" : Symboldurchmesser in mm
	"ticks.in" : Länge große Ticks innen in mm
	"ticks.out" : Länge große Ticks außen in mm
	"small ticks.in" : Länge kleine Ticks innen in mm
	"small ticks.out" : Länge kleine Ticks außen in mm
	"xscale" : Höhe x-Skala in mm. 0 auto
	"yscale" : Breite y-Skala in mm. 0 auto
	"linewidth.cosys" : Linienbreite Koordinatensystem. <0 für Breite in Pixeln. Z.B. -3 für 3 Pixel.
	"linewidth.grid" : Linienbreite Hauptgitter. <0 für Breite in Pixeln. Z.B. -3 für 3 Pixel.
	"linewidth.sec grid" : Linienbreite Nebengitter. <0 für Breite in Pixeln. Z.B. -3 für 3 Pixel.
	"linewidth.cursor" : Linienbreite Cursor. <0 für Breite in Pixeln. Z.B. -3 für 3 Pixel.
	"linewidth.curve" : Linienbreite Kurven. <0 für Breite in Pixeln. Z.B. -3 für 3 Pixel.
	"linestyle.cosys" : Linienart Koordinatensystem (0 durchgezogen, 1 gepunktet, 2 gepunktet weit, 3 gestrichelt, 4 gestrichelt weit, 5 Strich-Punkt)
	"linestyle.grid" : Linienart Hauptgitter (0 durchgezogen, 1 gepunktet, 2 gepunktet weit, 3 gestrichelt, 4 gestrichelt weit, 5 Strich-Punkt)
	"linestyle.sec grid" : Linienart Nebengitter (0 durchgezogen, 1 gepunktet, 2 gepunktet weit, 3 gestrichelt, 4 gestrichelt weit, 5 Strich-Punkt)
	"linestyle.cursor" : Linienart Cursor (0 durchgezogen, 1 gepunktet, 2 gepunktet weit, 3 gestrichelt, 4 gestrichelt weit, 5 Strich-Punkt)
	"angle 3D" : Winkel z-Achse 3D in Grad
	"font.name" : Schriftart Name, z.B. "Arial".
	"font.size" : Schriftgröße, in pt, z.B. 8
	"font.style" : Schriftstil (0 auto, 1 Standard, 2 fett, 3 kursiv, 4 fett und kursiv, 5 unterstrichen, 6 fett und unterstrichen, 7 kursiv und unterstrichen, 8 fett und kursiv und unterstrichen)
	"small font.name" : Schriftart Name, kleine Schrift, z.B. "Arial".
	"small font.size" : Schriftgröße, kleine Schrift, in pt, z.B. 8
	"small font.style" : Schriftstil, kleine Schrift (0 auto, 1 Standard, 2 fett, 3 kursiv, 4 fett und kursiv, 5 unterstrichen, 6 fett und unterstrichen, 7 kursiv und unterstrichen, 8 fett und kursiv und unterstrichen)

Wert	Auf welchen Wert soll diese Eigenschaft gesetzt werden?

Beschreibung:

Wenn die Einstellungen individuell (siehe "individual") pro Kurvenfenster sind, wird nur an diesem Kurvenfenster verändert. Sonst global.

Die gesetzten Eigenschaften werden beim Drucken, beim Exportieren von Grafik und beim Kopieren in die Ablage und in den Reportgenerator beachtet.

Beispiele:

Größe der gesamten Grafik für Querformat setzen. Nur für dieses Kurvenfenster

```
CwPrintSet("individual", 1)
CwPrintSet("layout", 2)
CwPrintSet("width", 180)
CwPrintSet("height", 260)
```

Vorbereitung für pdf-Export

```
CwGlobalSet("graphics.type", 0)
CwGlobalSet("export.dpi", 300)
CwGlobalSet("export.pdf.append", 0)
CwGlobalSet("export.orientation", 1)
CwGlobalSet("export.pdf.method", 2)
CwPrintSet("layout", 2)
CwPrintSet("font.size", 10)
CwPrintSet("font.style", 0)
CwPrintSet("font.name", "Arial")
```

Siehe auch:

CwDisplaySet, CwGlobalSet

CwReplace

Anwendungsbereich: Kurvenfenster

In einem Kurvenfenster wird ein Kanal mit der Bezeichnung "AlteBezeichnung" dargestellt. Dieser Kanal soll nun durch KanalErsatz in diesem Kurvenfenster ersetzt werden. Anschließend wird KanalErsatz anstelle des Kanals mit der Bezeichnung AlteBezeichnung dargestellt.

Deklaration:

```
CwReplace ( KanalErsatz, AlteBezeichnung ) -> AnzahlErsetzt
```

Parameter:

KanalErsatz	Datensatz, der im Kurvenfenster dargestellt werden soll.
AlteBezeichnung	Bezeichnung des dargestellten Kanals
AnzahlErsetzt	Rückgabe ist die Anzahl der ersetzten Kanäle. (optional)

Beschreibung:

Taucht der Kanal mehrfach auf, wird auch mehrfach ersetzt. Ist der Kanal gar nicht vorhanden, bleibt die Funktion ohne Wirkung.

Erlaubte Kanalnamen haben die Form "channel" oder bei Kanälen in Gruppen "group:channel".

Die Funktion wird angewendet, wenn nach dem Laden einer Kurvenfenster-Konfiguration andere Kanäle in dieser Konfiguration dargestellt werden sollen als einst dargestellt waren, als die Konfiguration gesichert wurde.

Beim ersten Ersetzen werden das Datenelement, die zugehörige Linie, Achse und das Koordinatensystem selektiert.

Der Parameter AlteBezeichnung wird stets ohne .X, .Y etc. angegeben.

Der Parameter KanalErsatz kann .X, .Y etc. enthalten, wenn ab nun allein genau diese Komponente darzustellen ist.

Beispiele:

Laden einer Konfiguration, die abgespeichert wurde, als die Kanäle anders benannt waren (damals CH1 und CH2). Die Kanäle, die nun temperature und pressure benannt sind, sollen dargestellt werden.

```
CwLoadCCV("curve1", "1.ccv")
CwReplace(temperature, "CH1")
CwReplace(pressure, "CH2")
```

Der Kanal S wird durch den Betrag eines Datensatzes ersetzt. Die darzustellende Komponente wird auf .B gesetzt.

```
CwReplace(Spektrum.B, "S")
```

Siehe auch:

CwNewChannel

CwSaveCCV

Anwendungsbereich: Kurvenfenster

Speichert die Konfiguration des selektierten Kurvenfensters in einer *.ccv-Datei.

Deklaration:

```
CwSaveCCV ( Dateiname ) -> Fehlertext
```

Parameter:

Dateiname	In welche Datei soll die Konfiguration gespeichert werden?
Fehlertext	Rückgabe ist ein Fehlertext, im Erfolgsfall ein leerer Text (optional)

Beschreibung:

Die Konfiguration des Kurvenfensters enthält alle Attribute der Darstellung, aber nicht die dargestellten Messdaten selbst.

Beispiele:

Sichern in einer CCV-Datei

```
CwLoadCCV(data, "1.ccv")
CwDisplaySet("displaymode", 2)
CwSaveCCV("2.ccv")
```

Sichern in einer CCV-Datei

```
CwSelectWindow(data)
CwSaveCCV("2.ccv")
```

Siehe auch:

CwLoadCCV

CwSelectByChannel

Anwendungsbereich: Kurvenfenster

Selektiert ein Element (z.B. Achse) des selektierten Kurvenfensters mittels eines Kanals.

Deklaration:

```
CwSelectByChannel ( Elementsorte, Kanal )
```

Parameter:

Elementsorte	Welche Sorte von Element soll selektiert werden?
	"axis" : y-Achse zum Kanal
	"cosys" : Koordinatensystem zum Kanal
	"data" : Datenelement zum Kanal
	"line" : Linie zum Kanal
Kanal	Datensatz, der im Kurvenfenster dargestellt ist. Die Variable selbst oder ihr Name

Beschreibung:

Wenn der Datensatz mehrfach dargestellt ist, wird das erste gefundene Element ermittelt.

Wenn das Element nicht existiert, ist keines mehr von dieser Sorte selektiert.

Wird der Kanal über seinen Namen angegeben, dann ist erlaubt: "channel" oder "group:channel"

Diese Technik funktioniert auch, wenn der Kanal gerade nicht existiert, aber im Kurvenfenster sein Name vermerkt ist.

Beispiele:

Die Achse auswählen, die die Variable data darstellt.

```
CwSelectWindow("curve1")
CwSelectByChannel("axis", data)
CwAxisSet("min", -10)
CwAxisSet("max", 10)
```

Die Linie auswählen, die den Kanal mit Namen "ch1" darstellt.

```
CwSelectWindow("curve1")
CwSelectByChannel("line", "ch1")
CwLineSet("symbol", 2)
```

Siehe auch:

CwSelectByIndex

CwSelectByIndex

Anwendungsbereich: Kurvenfenster

Selektiert ein Element (z.B. Achse) innerhalb eines selektierten Kurvenfensters.

Deklaration:

```
CwSelectByIndex ( Elementsorte, Index )
```

Parameter:

Elementsorte	Welche Sorte von Element soll selektiert werden?
	"axis from data" : y-Achse zum selektierten Datenelement (Index = 1)
	"axis from line" : y-Achse zur selektierten Linie (Index = 1)
	"x-axis" : x-Achse (Index = 1)
	"y-axis" : y-Achse
	"y-axis in cosys" : y-Achse im selektierten Koordinatensystem
	"z-axis" : z-Achse oder Winkel-Achse (Index = 1)
	"3D color-axis" : Achse mit 3D Farblegende (Index = 1)
	"cosys" : Koordinatensystem
	"cosys from axis" : Koordinatensystem zur selektierten y-Achse (Index = 1)
	"cosys from data" : Koordinatensystem zum selektierten Datenelement (Index = 1)
	"cosys from line" : Koordinatensystem zur selektierten Linie (Index = 1)
	"data" : Datenelement
	"data in axis" : Datenelement in der selektierten y-Achse
	"data in cosys" : Datenelement im selektierten Koordinatensystem
	"data in line" : Datenelement in der selektierten Linie
	"line" : Linie
	"line from data" : Linie zum selektierten Datenelement (Index = 1)
	"line from marker" : Linie zum selektierten Marker (Index = 1)
	"line in axis" : Linie in der selektierten y-Achse
	"line in cosys" : Linie im selektierten Koordinatensystem
	"line.measure" : Linie zum Messcursor (1 links, 2 rechts)
	"marker" : Marker
	"usertick" : Eigene Ticks Achse
	"header" : Kopf- oder Fußzeile oder Überschrift
Index	Index, das soundsovielte Element von dieser Sorte. Beginnend bei 1.

Beschreibung:

Wenn das Element nicht existiert, ist keines mehr von dieser Sorte selektiert.

Ein Koordinatensystem, eine Achse, eine Linie, ein Datenelement und ein Marker können gleichzeitig selektiert sein. Nur ein Element von einer Sorte kann selektiert sein.

Wenn der Anwender mittels Menü und Dialogen das Kurvenfenster verändert, kann sich das auf die Selektion auswirken. Insbesondere kann die Selektion verschwinden.

Beispiele:

Die 1. y-Achse auswählen und parametrieren

```
CwSelectWindow("curve1")
CwSelectByIndex("y-axis", 1)
CwAxisSet("min", -10)
CwAxisSet("max", 10)
```

Siehe auch:

CwSelectByChannel

CwSelectMode

Anwendungsbereich: Kurvenfenster

Legt fest, wie anschließend Kurvenfenster identifiziert werden.

Deklaration:

```
CwSelectMode ( Identifikationsart )
```

Parameter:

Identifikationsart	Wie soll das Kurvenfenster identifiziert werden?
	"auto" : Automatische Identifikation
	"caption" : Die Titelzeile des Kurvenfensters ist Bezug. Diese gibt es nur sichtbar bei frei fliegenden Kurvenfenstern, die nicht in ein anderes Fenster eingebettet sind.
	"newest" : Zuletzt erzeugtes Kurvenfenster.
	"title" : Der Titel des Kurvenfensters soll das Kurvenfenster identifizieren. Für Kurvenfenster in Dialogen und im Panel.
	"variable" : Eine Variable ist Bezugsdatensatz

Beschreibung:

Wenn die Funktion noch nicht aufgerufen wurde, wird "auto" angenommen.

Bei automatischer Identifikation wird zunächst der Bezugsdatensatz überprüft. Falls dabei kein Kurvenfenster gefunden wird, wird der Titel überprüft.

Wird bei automatischer Identifikation ein Kurvenfenster neu erzeugt, so wird im Fall einer Textvariable als Identifikation der Inhalt dieser Textvariable als Titel gedeutet.

Die Funktion CwSelectMode selbst führt keine Identifikation fest. Sie legt nur fest, wie andere Funktionen später identifizieren, z.B. CwSelectWindow oder CwNewWindow.

Die Identifikationsart "newest" ist nur für Anwendungen geeignet, bei denen keine Identifikation bekannt ist wie z.B. zum Selektieren eines gerade erzeugten Zwillingsfensters, eines manuell geöffneten Fensters. Als Parameter für die folgende Funktion CwSelectWindow() genügt dann ein leerer Text: CwSelectWindow("")

Die Identifikationsart "caption" ist nur für Sonderanwendungen geeignet, denn die Titelzeile ändert sich durch viele Operationen.

Beim Neustart von imc FAMOS oder auch beim Neustart einer Sequenz, die nicht Teil eines Panels oder Dialogs ist, wird der Selektionsmodus wieder auf automatisch gesetzt.

Wenn eine Sequenz den Selektionsmodus explizit setzt, ist es empfohlen, dass sie am Ende den Selektionsmodus wieder auf automatisch schaltet.

Beispiele:

Zum Identifizieren von Kurvenfenstern soll nur die Technik mit dem Bezugsdatensatz benutzt werden

```
CwSelectMode("variable")
CwSelectWindow(data)
```

Zum Identifizieren von Kurvenfenstern soll nur die Technik mit dem Titel benutzt werden

```
CwSelectMode("title")
CwSelectWindow("curve1")
```

Herstellen des automatischen Modus

```
CwSelectMode("auto")
CwSelectWindow("curve1")
; ... Arbeiten mit dem Kurvenfenster
CwSelectWindow(data)
; ... Arbeiten mit dem Kurvenfenster
```

Identifizieren eines gerade erzeugten Zwillingsfensters:

```
CwAction("win.twin")
CwSelectMode("newest")
CwSelectWindow("")
```

Siehe auch:

CwSelectWindow

CwSelectWindow

Anwendungsbereich: Kurvenfenster

Selektiert ein Kurvenfenster anhand von einer Variablen, die als Bezug benutzt wird oder auch anhand seines Titels.

Deklaration:

```
CwSelectWindow ( Identifikation ) -> Exist
```

Parameter:

Identifikation	Dieser Datensatz identifiziert das Kurvenfenster. Mit CwSelectMode wurde festgelegt, wie die Identifikation erfolgt.
Exist	Rückgabe 1, falls das Kurvenfenster existiert; 0 sonst. (optional)

Beschreibung:

Wenn das Fenster nicht existiert, wird keins erzeugt und auch keins selektiert. Wenn das Erzeugen gewünscht ist, kann CwNewWindow benutzt werden.

Falls das Kurvenfenster inzwischen geschlossen worden sein kann, sollte der Rückgabewert abgefragt werden.

Existiert das Kurvenfenster nicht, wird im eine Fehlermeldung angezeigt, die aber nicht zum Abbruch der Sequenz führt.

Beispiele:

Die Variable data ist als Kurvenfenster dargestellt

```
CwSelectWindow(data)
CwDisplaySet("displaymode", 1)
CwAction("unzoom")
```

In einem Dialog oder Panel existiert ein Kurvenfenster mit dem Titel "curve1"

```
CwSelectWindow("curve1")
CwAction("unzoom")
```

Danach soll ein weiteres Kurvenfenster mit dem Titel "curve2" bearbeitet werden:

```
CwSelectWindow("curve2")
CwDisplaySet("displaymode", 2)
```

Auswahl eines Kurvenfensters "curve1" im Panel.

```
CwSelectWindow("curve1")
```

Falls im Panel mehrere Kurvenfenster denselben Titel haben, muss der Seitenname (hier "page1") mit Punkt getrennt vorangestellt werden.

```
CwSelectWindow("page1.curve1")
```

Übertragung zum Reportgenerator

```
CwSelectWindow(data)
CwDisplaySet("name", "nn")
CvPosi( "nn", 0, 0, 600, 400 )
RgCurveSet("r1", "nn", 0)
```

Alternativ für frei fliegende Kurvenfenster:

```
CwSelectWindow(data)
CwDisplaySet("displaymode", 2)
CvPosi( data, 0, 0, 600, 400 )
RgCurveSet("r1", data, 0)
```

Alternativ für eingebettete Kurvenfenster:

```
CwSelectWindow("curve2")
CwDisplaySet("displaymode", 2)
RgCurveSet("r1", "curve2", 0)
```

Der Erfolg der Funktion kann überprüft werden (alternativ: CwIsWindow):

```
if CwSelectWindow("curve1") = 0
   error handling...
end
```

Vor dem Selektieren wird der Selektionsmodus auf einen definierten Wert gesetzt.

```
CwSelectMode("auto")
```

```
CwSelectWindow("curve1")
```

Siehe auch:

CwSelectMode

CwSequenceEnable

Anwendungsbereich: Kurvenfenster

Festlegen der Situationen, in denen FAMOS eine Ereignis-Sequenz zu einem Kurvenfenster aufruft. Die Situation besteht aus einer Angabe der Location (z.B. über dem Koordinatensystem) und einer Mausoperation (z.B. linke Maustaste geklickt).

Deklaration:

```
CwSequenceEnable ( Location, MouseLeft [, MouseRight] [, MouseMove] )
```

Parameter:

Location	Angabe des Ortes. Wo ist die Maus?
	"cosys" : Innerhalb des Koordinatensystems; MouseLeft=("no", "click", "drag", "2"); MouseRight=("no", "click"); MouseMove=("no", "move")
	"area" : Fläche außerhalb des Koordinatensystems; MouseLeft=("no", "click", "drag", "2"); MouseRight=("no", "click"); MouseMove=("no", "move")
	"measure" : Messcursor; MouseLeft=("no", "click", "drag"); MouseRight=("no", "click"); MouseMove=("no").
	"link" : Linklinie; MouseLeft=("no", "click", "drag"); MouseRight=("no"); MouseMove=("no")
	"all" : Alle reset. Die Einträge für alle Locations werden gelöscht. MouseLeft=("no"); MouseRight=("no"); MouseMove=("no")
MouseLeft	Maus-Operation mit linker Taste. Bei welcher Bedienung der linken Maustaste soll an der angegebenen Location eine Sequenz ausgeführt werden.
	"no" : nein, nicht zu benutzen. Keine Sequenz ausführen.
	"click" : Klick mit linker Maustaste. Die Sequenz wird gestartet, wenn die Maustaste losgelassen wird.
	"drag" : Drag. Zieh-Bewegung mit linker Maustaste. Linke Maustaste niederdrücken, Maus verschieben, Taste loslassen. Die Sequenz wird gestartet, wenn die Taste niedergedrückt wird. Sie läuft typisch solange bis die Taste wieder losgelassen wird.
	"2" : Linke Maustaste doppelt (zweimal) geklickt
MouseRight	Maus-Operation mit rechter Taste. Bei welcher Bedienung der rechten Maustaste soll an der angegebenen Location eine Sequenz ausgeführt werden. (optional , Standardwert: "no")
	"no" : nein, nicht zu benutzen. Keine Sequenz ausführen.
	"click" : Klick mit rechter Maustaste. Die Sequenz wird gestartet, wenn die Maustaste losgelassen wird.
MouseMove	Soll bei Mausbewegung an der angegebenen Location eine Sequenz ausgeführt werden. (optional , Standardwert: "no")
	"no" : nein, nicht zu benutzen. Keine Sequenz ausführen.
	"move" : Die Maus wird bewegt (verschoben). Keine Maustaste ist gedrückt.

Beschreibung:

Im Panel von FAMOS können für ein Kurvenfenster Ereignissequenzen hinterlegt werden: So kann z.B. konfiguriert werden, dass ein Klick mit der Maus auf das Kurvenfenster eine solche Ereignissequenz anstößt. In der Ereignissequenz kann dann die Position des Mauszeigers ausgewertet und entsprechend reagiert werden.

Die Funktion CwSequenceEnable() wird vor dem möglichen Auslösen von Ereignis-Sequenzen zum Kurvenfenster aufgerufen. Damit wird dann gesteuert bzw. festgelegt, in welchen Situationen überhaupt eine Ereignis-Sequenzen ausgelöst wird.

Das Kurvenfenster muss selektiert sein.

Die Funktion kann nacheinander für verschiedene Situationen aufgerufen werden. Dabei wird für jede Situation festgelegt, bei welcher Aktion des Anwenders eine Sequenz ausgeführt wird.

Wenn die rechte Maustaste eine Sequenz auslösen soll, wird am Kurvenfenster bei Rechtsklick kein Kontextmenü angezeigt.

Wenn die linke Maustaste eine Sequenz auslösen soll, werden die üblicherweise am Kurvenfenster durch die linke Maustaste ausgelösten Aktionen nicht durchgeführt.

Beim Messen startet die Kombination "drag" und Rechtsklick mit dem ersten Niederdrücken einer der beiden Tasten die Sequenz. Die Sequenz läuft typisch solange wie mindestens eine der beiden Tasten gedrückt gehalten wird.

Die Funktion sollte nicht innerhalb einer Ereignis-Sequenz zum Kurvenfenster aufgerufen werden. Ausnahme: Ganz am Ende mit den Parametern ("all", "no").

Beispiele:

Bei der Initialisierung des Panels wird festgelegt, dass ein Zweifachklick auf die Fläche des Koordinatensystems eine Sequenz anstoßen soll.

```
CwSequenceEnable("all", "no")
CwSequenceEnable("cosys", "2", "no", "no")
```

Bei der Initialisierung des Panels wird festgelegt, dass ein Klick auf die Fläche des Koordinatensystems eine Sequenz anstoßen soll.

Außerdem wird verdeutlicht, dass an den anderen Locations keine Sequenz gestartet werden soll.

```
CwSequenceEnable("cosys", "click", "no", "no")
CwSequenceEnable("area", "no", "no", "no")
CwSequenceEnable("measure", "no", "no", "no")
CwSequenceEnable("link", "no", "no", "no")
```

Das Kurvenfenster soll wieder seine Standard-Bedienung erhalten. Keine Sequenz soll ausgelöst werden.

```
CwSequenceEnable("all", "no")
```

Bei der Initialisierung des Panels wird festgelegt, dass ein Klick überall und ein Rechtsklick innerhalb des Koordinatensystems eine Sequenz anstoßen soll.

In all diesen Situationen wird dieselbe Ereignissequenz gestartet. Innerhalb der Sequenz wird dann die aktuelle Situation mit CwSequenceState() abgefragt.

```
CwSequenceEnable("all", "no")
CwSequenceEnable("cosys", "click", "click", "no")
CwSequenceEnable("area", "click", "no", "no")
```

Bei der Initialisierung des Panels wird festgelegt, dass nach Bewegung des Messcursors eine Sequenz angestoßen werden soll.

Die Sequenz soll beim Loslassen der Maustaste ausgeführt werden, nicht schon während der Bewegung.

```
CwSequenceEnable("all", "no")
CwSequenceEnable("measure", "click", "no", "no")
```

Bei der Initialisierung des Panels wird festgelegt, dass nach Bewegung der Linklinie eine Sequenz angestoßen werden soll.

Die Sequenz soll beim Loslassen der Maustaste ausgeführt werden, nicht schon während der Bewegung.

```
CwSequenceEnable("all", "no")
CwSequenceEnable("link", "click", "no", "no")
```

Siehe auch:

CwSequenceState

CwSequenceState

Anwendungsbereich: Kurvenfenster

Abfrage des Zustands und der Mausposition innerhalb einer FAMOS-Ereignis-Sequenz zum Kurvenfenster

Deklaration:

```
CwSequenceState ( Eigenschaft ) -> Zustand
```

Parameter:

Eigenschaft	Welche Eigenschaft soll abgefragt werden?
	"dragging" : drag läuft noch (0 nein, 1 ja). Wird benutzt, um eine Schleife aufzubauen.
	"cancel" : Ziehbewegung (drag Operation) abgebrochen (0 nein, 1 ja). Eine Ziehbewegung kann z.B. durch ESC oder Timeout abgebrochen werden.
	"val.x" : x-Koordinate in Einheiten der x-Achse
	"val.y" : y-Koordinate in Einheiten der y-Achse
	"operation" : Die durchgeführte Aktion (0 keine, 1 linker Klick, 2 rechter Klick, 3 Drag, 4 Move, 5 doppelter linker Klick)
	"location" : Die Location, an der die Operation begann, siehe Parameter Location von CwSequenceEnable(). (1 Koordinatensystem, 2 Fläche, 3 Messcursor, 4 Linklinie)
	"key.shift" : Umschalten-Taste (Shift) gedrückt (0 nein, 1 ja)
	"key.ctrl" : Strg-Taste (Ctrl) gedrückt (0 nein, 1 ja)
	"mouse.left" : Linke Maustaste gedrückt (0 nein, 1 ja)
	"mouse.right" : Rechte Maustaste gedrückt (0 nein, 1 ja)
	"outside" : Mausposition außerhalb (0 nein, 1 ja). Bei Zieh-Operationen, wenn die Maus das eigentliche Objekt verlässt.
	"over.cosys" : Maus über diesem Koordinatensystem (0 nein, sonst Index Koordinatensystem >= 1)
	"over.line" : Maus über diesem Linienzug (0 nein, sonst Index Linie >= 1). Diese Abfrage kann länger dauern. Diese Abfrage liefert den aktuellen Wert, nicht den passend zum Status hinterlegten Wert.
	"over.marker" : Maus über diesem Marker (0 nein, sonst Index Marker >= 1)
	"over.x-axis" : Maus über x-Achse (0 nein, sonst Index x-Achse >= 1)
	"over.y-axis" : Maus über y-Achse (0 nein, sonst Index y-Achse >= 1)
	"pix.x" : x-Pixelposition. Mit null beginnend am linken Rand des Kurvenfensters. Nur bei Koordinatensystem und Fläche
	"pix.y" : y-Pixelposition. Mit null beginnend am oberen Rand des Kurvenfensters. Nur bei Koordinatensystem und Fläche
	"related.cosys" : Beteiligtes Koordinatensystem (0 nein, sonst Index Koordinatensystem >= 1)
	"related.line" : Beteiligte Linie (0 nein, sonst Index Linie >= 1). Z.B. bei Messcursor entlang Linie
	"related.marker" : Beteiligter Marker (0 nein, sonst Index Marker >= 1)
	"related.x-axis" : Beteiligte x-Achse (0 nein, sonst Index x-Achse >= 1)
	"related.y-axis" : Beteiligte y-Achse (0 nein, sonst Index y-Achse >= 1)
Zustand	Der abgefragte Zustand

Beschreibung:

Im Panel von FAMOS können für ein Kurvenfenster Ereignissequenzen hinterlegt werden: So kann z.B. konfiguriert werden, dass ein Klick mit der Maus auf das Kurvenfenster eine solche Ereignissequenz anstößt. In der Ereignissequenz kann dann die Position des Mauszeigers ausgewertet und entsprechend reagiert werden.

Die Funktion CwSequenceState() wird innerhalb einer Ereignis-Sequenz zum Kurvenfenster aufgerufen.

Das Kurvenfenster muss selektiert sein.

Mit dem Beginn einer Ereignissequenz ist das Kurvenfenster bereits selektiert.

Rückgabe 0, wenn die passende Situation nicht vorliegt.

Wenn eine Zieh-Bewegung (drag) ausgeführt wird, dann wird in einer Schleife mittels Eigenschaft="dragging" abgefragt, ob der Ziehvorgang noch andauert.

Innerhalb der Schleife kann der aktuelle Status abgefragt werden. Im ersten Durchlauf werden die Koordinaten zu Beginn der Bewegung zurückgegeben. Im letzten Durchlauf werden die Koordinaten am Ende der Bewegung zurückgegeben. Das kann ausgenutzt werden, Sequenzzeilen, die von den Koordinaten abhängen, nicht vor, innerhalb und nach der Schleife mehrfach zu schreiben.

Erst nach erneutem Aufruf von Eigenschaft="dragging" können neue andere Koordinaten zurückgegeben werden. Das kann ausgenutzt werden, um innerhalb der Schleife stimmige und zueinander passende Werte abzufragen.

Bei einer Zieh-Bewegung (drag) liefert die Eigenschaft="dragging" mindestens einmal 1 zurück, sodass eine entsprechende Schleife mindestens einmal durchlaufen wird.

Bei Abbruch einer Ziehbewegung (z.B. ESC) liefert Eigenschaft="dragging" ggf. ab dem 2. Aufruf eine 0 zurück.

Während die Ereignis-Sequenz ausgeführt wird, werden keine neuen Mausklick von Famos angenommen. Deshalb sollte eine Ereignis-Sequenz möglichst schnell abgearbeitet sein. Der Anwender muss bei der Bedienung des Kurvenfensters vor einem Klick die Maus ggf. kurz stillhalten, um das Ende einer noch laufenden Ereignis-Sequenz abzuwarten.

Innerhalb einer Schleife mit Bedingung "dragging" darf CwUpdateEnable(0) nicht dauerhaft aktiv sein. Denn dabei werden jegliche Mausbewegungen und -klicks ignoriert, die zu einer Zustandsänderung führen würden.

Die Abfrage von "dragging" liefert nach ca. 5s ohne Mausbewegung und -klicks eine 0 zurück. Das wird als Abbruch gedeutet. Eine von "cancel" wird eine 1 liefern.

Die Abarbeitung einer Ereignissequenz kann merklich länger dauern, wenn ins FAMOS Ausgabe-Fenster geschrieben wird. So z.B. bei Warnungen oder Info-Nachrichten. Siehe z.B. Aufruf von SetOption("Func.NoInfoMessages", "Yes")

Beispiele:

Ereignis-Sequenz bei Klick auf das Koordinatensystem: Die Koordinaten des Klicks werden abgefragt.

```
x = CwSequenceState("val.x")
y = CwSequenceState("val.y")
```

Diese und andere Zustände können abgefragt und ausgewertet werden.

An geeigneter Stelle, z.B. bei der Initialisierung des Panels wurde der Klick am Kurvenfenster erlaubt:

```
CwSequenceEnable("all", "no")
CwSequenceEnable("cosys", "click", "no", "no")
```

Ereignis-Sequenz bei Ziehbewegung (Drag) auf dem Koordinatensystem: In einer Schleife wird abgefragt, ob die Ziehbewegung noch andauert.

```
x = CwSequenceState("val.x")
; Hier wird eingetragen, was beim Niederdrücken der Maustaste geschehen soll.
while CwSequenceState("dragging") <> 0
   x = CwSequenceState("val.x")
   y = CwSequenceState("val.y")
   ; Hier wird eingetragen, was während des Ziehens geschehen soll.
end
x = CwSequenceState("val.x")
; Hier wird eingetragen, was nach dem Loslassen der Maustaste geschehen soll.
```

An geeigneter Stelle, z.B. bei der Initialisierung des Panels wurde das Ziehen am Kurvenfenster erlaubt:

```
CwSequenceEnable("all", "no")
CwSequenceEnable("cosys", "drag", "no", "no")
```

Verkürzte Ereignis-Sequenz bei Ziehbewegung (Drag)

Dabei wird ausgenutzt, dass alle Koordinaten während der Bewegung, vor allem auch bei Anfang und Ende, in der Schleife aufgezählt werden.

Wenn also keine besondere Behandlung bei Beginn oder Ende der Ziehbewegung nötig ist, kann die ganze Verarbeitung innerhalb der Schleife erfolgen.

```
while CwSequenceState("dragging") <> 0
   x = CwSequenceState("val.x")
   ; Hier wird eingetragen, was während des Ziehens geschehen soll.
end
```

Eine Ereignis-Sequenz wird für mehrere Situationen benutzt, z.B. Rechtsklick und Linksklick.

Die Situation muss mit CwSequenceState("operation") oder auch CwSequenceState("location") abgefragt werden abhängig von den mit CwSequenceEnable() festgelegten Kombinationen.

In komplexeren Anwendungen werden dann switch case Konstrukte eingesetzt.

```
x = CwSequenceState("val.x")
if CwSequenceState("operation") = 1
   y = CwSequenceState("val.y")
   ; Hier wird eingetragen, was bei Linksklick geschehen soll.
else
   ; Hier wird eingetragen, was bei Rechtsklick geschehen soll.
end
```

An geeigneter Stelle, z.B. bei der Initialisierung des Panels wurde der Klick am Kurvenfenster erlaubt:

```
CwSequenceEnable("all", "no")
CwSequenceEnable("cosys", "click", "click", "no")
```

Ereignis-Sequenz bei Loslassen des linken Messcursors. Mit Hilfe des Messcursors soll eine Position festgelegt werden.

```
x1 = CwSequenceState("val.x")
x2 = CwDisplayGet("measure.x.left")
```

x1 ist die Position genau beim Loslassen der Maustaste.

x2 ist die Position des Messcursors zum aktuellen Zeitpunkt. Meist vergeht eine (kleine) Zeit, bis nach dem Loslassen der Taste die Sequenz gestartet und ausgeführt wird. Dabei kann sich die Mausposition schon verändert haben.

An geeigneter Stelle, z.B. bei der Initialisierung des Panels wurde der Messcursor zum Auslösen der Ereignissequenz erlaubt:

```
CwSequenceEnable("all", "no")
CwSequenceEnable("measure", "click", "no", "no")
```

Ziehen des Messcursors mit linker oder rechter Maustaste oder sogar mit beiden gleichzeitig.

```
while CwSequenceState("dragging") <> 0
   x = CwSequenceState("val.x")
   LeftMouseButtonDown = CwSequenceState("mouse.left")
   RightMouseButtonDown = CwSequenceState("mouse.right")
   xLeft = CwDisplayGet("measure.x.left")
   xRight = CwDisplayGet("measure.x.right")
   ; Hier wird eingetragen, was während des Ziehens geschehen soll.
end
```

An geeigneter Stelle, z.B. bei der Initialisierung des Panels wurde diese Unterstützung aktiviert:

```
CwSequenceEnable("all", "no")
CwSequenceEnable("measure", "drag", "click", "no")
```

War eine Ziehbewegung erfolgreich? Dabei soll am Ende der Ziehbewegung eine wichtige Operation ausgeführt werden. Aber auf keinen Fall, wenn der Anwender abgebrochen hat oder aus anderem Grund abgebrochen wurde.

```
while CwSequenceState("dragging") <> 0
end
if CwSequenceState("cancel") = 0
   ; hier erfolgreiches Ende. Nicht abgebrochen!
   ; hier die wichtige Operation ausführen
end
```

Siehe auch:

CwSequenceEnable

CwUpdateEnable

Aktualisierung der Kurvenfenster sperren

Deklaration:

```
CwUpdateEnable ( EwUpdate )
```

Parameter:

EwUpdate	Aktualisierung an/aus
	0 : Aktualisieren der Kurvenfenster sperren
	1 : Aktualisieren der Kurvenfenster (wieder) erlauben

Beschreibung:

Wenn [EWUpdate] ungleich 0 gesetzt ist, werden während des Ablaufs einer Sequenz WM_PAINT- und andere Nachrichten erlaubt und imc FAMOS ist bedienbar.

Sonst nicht! So kann z. B. mehrmaliges Updaten eines Kurvenfensters bei Neugestaltung verhindert werden.

Achtung:

[EwUpdate] sollte nur vor einer Gruppe von Funktionen zur Konfigurierung eines Kurvenfensters auf 0 gesetzt werden und gleich danach wieder auf 1. Schweres Fehlverhalten ist bei unsachgemäßer Anwendung der Funktion möglich!!!

Beispiele:

Die Achsen eines Kurvenfensters werden parametriert. Das Kurvenfenster wird erst nach der letzten Anweisung neu gezeichnet.

```
CwUpdateEnable(0)
CwSelectWindow("curve1")
CwSelectByIndex("y-axis", 1)
CwAxisSet("range", 4)
CwAxisSet("min", -10)
CwAxisSet("max", 10)
CwSelectByIndex("x-axis", 1)
CwAxisSet("scale", 4)
CwAxisSet("range", 1)
CwUpdateEnable(1)
```

DataFormat?

Das Datenformat einer Variablen wird ermittelt.
Alternativer Name: **DatFormat?**

Deklaration:

```
DataFormat? ( Variable ) -> EwFormatCode
```

Parameter:

Variable	Variable, dessen Datenformat ermittelt werden soll
EwFormatCode	Format
	0 : 4 Byte reell (float)
	1 : 8 Byte reell (double)
	2 : 1 Byte ganzzahlig
	3 : 2 Byte ganzzahlig
	4 : 4 Byte ganzzahlig
	5 : 1 Byte ganzzahlig ohne Vorzeichen
	6 : 2 Byte ganzzahlig ohne Vorzeichen
	7 : 4 Byte ganzzahlig ohne Vorzeichen
	8 : Digital
	9 : 2 Byte ganzzahlig Differenzen
	10 : 6 Byte ganzzahlig ohne Vorzeichen
	11 : ASCII mit Zeitstempel
	12 : 8 Byte ganzzahlig
	13 : 8 Byte ganzzahlig ohne Vorzeichen
	-1 : Unbekanntes Datenformat
	-2 : Datengruppe
	-3 : Text
	-4 : Textfeld

Beschreibung:

Die Funktion ermittelt, in welchem Datenformat eine Variable vorliegt.

Das Datenformat gibt an, wie die einzelnen Werte im Speicher bzw. auf dem Datenträger abgelegt werden. Der Speicherplatzbedarf eines Datensatzes, der Wertebereich und die erreichbare Genauigkeit werden durch das Datenformat bestimmt.

Wenn dieser Funktion ein XY-Datensatz übergeben wird, wird das Datenformat der Y-Komponente zurückgegeben. Wenn ein komplexer Datensatz übergeben wird, wird das Datenformates des Betrages bzw. des Realteiles ermittelt. Um die jeweils andere Komponente abzufragen, können die Komponentenkennungen benutzt werden.

```
DFormPhase = DataFormat?(MagnitudePhase.P)
DFormImag = DataFormat?(RealImag.I)
DFormX = DataFormat?(XYdata.X)
```

Um den Datentyp einer Variablen abzutesten, ist die neuere Funktion VerifyVar() im Allgemeinen besser geeignet.

Beispiele:

```
Format = DataFormat?(MyData)
IF Format <> 0
   SetDataFormat(MyData, 1, 0, 0)
END
```

Falls der Datensatz nicht bereits im reellen 4-Byte-Format vorliegt, wird er in dieses konvertiert.

Siehe auch:

SetDataFormat, VerifyVar, GetScale

dB

Umrechnung in Dezibel, d.h. 20 * log...

Deklaration:

```
dB ( EingangsDaten ) -> Transformiert
```

Parameter:

EingangsDaten	Daten, die in dB ausgedrückt werden sollen.
Transformiert	Resultierender Datensatz.

Beschreibung:

Die als Parameter übergebenen Daten werden in Dezibel ausgedrückt. Bei komplexen Parametern wird in den Typ DP transformiert, d. h. Polarkoordinatendarstellung mit Betrag in dB. Dezibel bedeutet den zwanzigfachen Zehnerlogarithmus der übergebenen Größe. Dezibel werden mit dB abgekürzt. dB sind keine Einheit, sondern deuten die genannte Rechenvorschrift an.

Eine Berechnung in dB ist üblich bei Übertragungsfunktionen und Schallbetrachtungen. Eine Berechnung einer Größe in dB lohnt sich, wenn betragsmäßig große und kleine Werte mit gleicher relativer Genauigkeit darzustellen sind.

- Bei der dB-Berechnung von reellen Zahlen wird vorab eine implizite Absolutbetragsbildung vorgenommen.
- Bei normalen oder XY-Datensätzen sind die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Komplexe Datensätze, die bereits vom Typ DP sind, können nicht mit dieser Funktion bearbeitet werden.
- Der Parameter darf strukturiert sein (Events/ Segmente).
- Bei der dB-Berechnung wird die y-Einheit auf dB gesetzt. Bei grafischer Darstellung ist eine solche Achsenbeschriftung üblich. Beim Kombinieren mit anderen Einheiten wird dB ersatzlos herausgekürzt, da es keine Einheit ist.
- Wird die Funktion auf einen komplexen Datensatz angewendet, der als Ortskurve dargestellt ist, verschwindet die Ortskurvendarstellung.

Beispiele:

```
NDdecibel = dB(NDdata)
; Diese Formel ist äquivalent zu Formel:
NDdecibel = 20 * log(Abs(NDdata))
; nur die Einheit ist unterschiedlich.
```

Berechnung eines Spektrums in dB:

```
DPspectrum = dB(FFT(NDdata))
```

Bei der Berechnung von dB können betragsmäßig sehr große negative Werte auftreten, die nicht sinnvoll sind. Es sollte zweckmäßigerweise die Funktion Clip() benutzt werden, um die dB-Zahlen auf sinnvolle minimale Werte zu begrenzen, z. B. -100dB. Die Obergrenze von 1000 sollte nur groß genug angesetzt werden:

```
NDdecibel = Clip(dB(NDdata), 1000, -100)
```

Siehe auch:

idB, log, Clip

DbBeginTransaction

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Beginn einer Transaktion.

Deklaration:

```
DbBeginTransaction ( ConnectId ) -> ErrorCode
```

Parameter:

ConnectId	Verbindungs-Identifikator.
ErrorCode	Ergebnis: Im Fehlerfall eine Fehlernummer, sonst Null.
	= 0 : Kein Fehler.
	< 0 : Fehlernummer.

Beschreibung:

Mit dieser Funktion wird eine Transaktion begonnen. Diese Funktion kann verwendet werden, wenn mehrere Aufgaben in der Sequenz miteinander verbunden werden sollen, damit sie als eine einzelne Verarbeitungseinheit ausgeführt werden können.

Diese Funktion muss nicht verwendet werden, wenn nur ein einzelner Aufruf von DbSql(), DbInsert(), DbUpdate() oder DbUpdate1() erfolgt. Diese Funktionen haben eine interne Transaktionssteuerung.

Wird die Funktion DbBeginTransaction() aufgerufen, so wird die interne Transaktionssteuerung außer Kraft gesetzt. Erst nach dem Aufruf von DbEndTransaction() wird die interne Transaktionssteuerung wieder aktiv.

Diese Funktion muss immer paarweise mit der Funktion DbEndTransaction() aufgerufen werden.

Beispiele:

In die Tabelle "Benutzer" wird die Gruppe grpInsert1 und in die Tabelle "Gruppe" die FAMOS-Gruppe grpInsert2 eingefügt. Dazu wird eine Transaktion gestartet. Wurden beide Funktionen ohne Fehler beendet, so wird die Transaktion mit einem Commit beendet. Im Fehlerfall erfolgt ein Rollback.

```
errorcode=DbBeginTransaction(ConnectID)
if errorcode < 0
   errortext=DbGetLastErrorText(ConnectID,1)
end
result1=DbInsert(ConnectID,"Benutzer",grpInsert1)
result2=DbInsert(ConnectID,"Gruppe",grpInsert2)
commit=1
if result1 < 0 or result2 < 0
   errortext=DbGetLastErrorText(ConnectID,1)
   commit=0
end
errorcode = DbEndtransaction(ConnectID,commit)
if errorcode < 0
   errortext=DbGetLastErrorText(ConnectID,1)
end
```

Siehe auch:

Transaktionen, DbEndTransaction, DbGetLastErrorText, DbGetLastErrorCode

DbClosePanel

Das aktive Panel wird geschlossen.
Die Funktion ist veraltet, statt dessen sollte die neuere Funktion PnClose() verwendet werden (ab V2022).

Deklaration:

```
DbClosePanel ( EwOption )
```

Parameter:

EwOption	Optionsparameter bzw. Rückgabewert eines Paneldialogs.

Beschreibung:

Das aktive Panel oder alle Panele werden geschlossen.

Die Bedeutung des Optionsparameters ist abhängig vom Kontext des Aufrufs:

Aufruf innerhalb einer Ereignissequenz eines Panel-Dialogs, der mit der Funktion Dialog() gestartet wurde:

Der Panel-Dialog wird geschlossen und der übergebene Parameter wird als Rückgabewert des Dialog()-Befehls verwendet. Die Funktion verhält sich so analog zur Funktion DlgCloseDialog() bei Anwenderdefinierten Dialogen.

Sonst:

Ein Wert von 0 bedeutet, dass das aktive Panel geschlossen wird. Eine 1 schließt alle geöffneten Panele.

Die Panele werden nicht gesichert, eventuelle Änderungen gehen verloren.

Beispiele:

Eine Panel-Datei wird geladen. Es werden diverse Aktualisierungen durchgeführt, danach wird das Panel gedruckt und wieder geschlossen.

```
err = DbLoadPanel("d:\templates\result.panel", 0)
IF err <> 0
   BoxMessage("Fehler", GetLastError(), "!1")
ELSE
   ; Verschiedene Aktualisierungen
   ; ...
   PnPrint(0)
   DbClosePanel(0)
END
```

Ein Panel 'InputValue.panel' besteht u.a. aus einem Eingabefeld "input" zur Eingabe eines positiven Zahlenwertes sowie 2 Schaltflächen 'OK' und 'Abbrechen'. Der Dialog()-Befehl liefert den eingegebenen Wert zurück bzw. -1, falls abgebrochen werden soll.

Ereignis-Sequenz 'Knopf gedrückt' für den 'OK'-Knopf:

```
value = PnGetValue("input")
DbClosePanel(value)
```

Ereignis-Sequenz 'Knopf gedrückt' für den 'Abbrechen'-Knopf

```
DbClosePanel(-1)
```

Ereignis-Sequenz 'Schließen' (Anwender verwendet Systemmenü zum Schließen):

```
; Selbes Verhalten wie 'Abbrechen'-Knopf
DbClosePanel(-1)
```

Aufruf des Dialogs:

```
value = Dialog("InputValue.panel", "", 0)
IF value < 0
   EXITSEQUENCE 0
END
;Weiter in der Sequenz...
...
```

Siehe auch:

PnClose, PnLoad, DbLoadPanel, DbShow, Dialog

DbConnect

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Herstellen einer Verbindung zu dem angegebenen Datenbanksystem.

Deklaration:

```
DbConnect ( ServerType, TxServerName, TxDatabaseName, TxUserName, TxPassword, TxExtConnectionString ) ->
ConnectId
```

Parameter:

ServerType	Typ des Datenbanksystems.
	1 : Zugriff auf Microsoft SQL Server Compact Edition 4.0. mittels ADO.Net Provider
	2 : Zugriff auf Microsoft SQL Server (2005, 2008) mittels ADO.Net Provider
	3 : Zugriff auf ein Oracle-Datenbanksystem (10g, 11g, 12c) mittels Oracle Data Provider for .NET.
	4 : Zugriff auf einen MySQL Server (5.5, 5.6) mittels MySQL Connector/NET
	5 : Zugriff auf einen Server über ODBC
	6 : Zugriff auf ein Oracle-Datenbanksystem (10g, 11g, 12c) mittels Oracle Data Provider for .NET, Managed Driver
TxServerName	Name des Servers.
TxDatabaseName	Name der Datenbank.
TxUserName	Der Benutzername zum Anmelden am Server.
TxPassword	Das Passwort für den angegebenen Benutzernamen.
TxExtConnectionString	Erweiterung zur Standardverbindungszeichenkette.
ConnectId	Ergebnis: Verbindungs-Identifikator oder Fehlernummer.
	> 0 : Verbindungs-Identifikator
	< 0 : Fehlernummer.

Beschreibung:

Aus den Parametern der Funktion wird eine Verbindungszeichenkette zusammengebaut. Es wird ein Verbindungsobjekt erstellt und ein Verbindungstest durchgeführt. Nach einem erfolgreichen Test, wird ein gültiger Verbindungs-Identifikator zurückgegeben. Um einen Fehler zu ermitteln, sind im Anschluss die Funktionen DbGetLastErrorCode() und/oder DbGetLastErrorText() zu verwenden.

Multithreading: Alle Funktionen des Datenbank-Kits dürfen nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

Es wird eine Verbindung zu einem Oracle-Datenbankserver unter Verwendung der tnsnames.ora Datei hergestellt.

```
DbInitialize()
ConnectId = DbConnect(3,"ORCL","","MyUsername"," MyPassword","")
if ConnectId < 0
   errortext=DbGetLastErrorText(0,1)
   exitsequence 1
end
```

Es wird eine Verbindung zu einem MySQL- Server aufgebaut. Dieser arbeitet über den TCP- Port 3307. Die Port-Nummer wird als Erweiterung der Verbindungszeichenkette angegeben.

```
DbInitialize()
ConnectId=DbConnect(4,"localhost","Sample_DB","MyUsername","MyPassword ", "port=3307;")
if ConnectId < 0
   errortext=DbGetLastErrorText(0,1)
   exitsequence 1
end
```

Es wird eine Verbindung zu einem MS SQL Server mittels Windows Authentifizierung aufgebaut.

```
DbInitialize()
ConnectId=DbConnect(2,"MyPC\SQL2008EXPRESS","SampleDB","","","")
if ConnectId < 0
   errortext=DbGetLastErrorText(0,1)
```

```
   exitsequence 1
end
```

Verbindung zu einem MySQL-Server über ODBC herstellen. Im ODBC-Manager ist eine Systemdatenquelle namens "DSN_MySql" dafür eingerichtet

```
DbInitialize()
ConnectId=DbConnect(5,"DSN_MySql","","MyUsername","MyPassword ","")
if ConnectId < 0
   errortext=DbGetLastErrorText(0,1)
   exitsequence 1
end
```

Weitere Beispiele siehe Handbuch

Siehe auch:

Datenbankverbindung, DbDisconnect, DbGetLastErrorText, DbGetLastErrorCode

DbDisconnect

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Verbindung zu einem Datenbankserver trennen.

Deklaration:

```
DbDisconnect ( ConnectId ) -> ErrorCode
```

Parameter:

ConnectId	Verbindungs-Identifikator.
ErrorCode	Ergebnis: Im Fehlerfall eine Fehlernummer, sonst Null.
	= 0 : Kein Fehler.
	< 0 : Fehlernummer.

Beschreibung:

Die Verbindung zu dem Datenbankserver wird getrennt.

Die Funktionen DbConnect() und DbDisconnect() sollten immer paarweise aufgerufen werden.

Beispiele:

Es wird eine Verbindung zu einem Oracle Datenbankserver aufgebaut. Dann werden Zugriffe auf das Datenbanksystem ausgeführt. Am Ende wird die Verbindung getrennt.

```
ConnectId = DbConnect(3,"ORCL","","MyUsername"," MyPassword","")
   :
   :
errorcode = DbDisconnect(ConnectId)
```

Siehe auch:

Datenbankverbindung, DbConnect, DbGetLastErrorText, DbGetLastErrorCode

DbEndTransaction

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Eine Transaktion wird festgeschrieben oder zurückgesetzt.

Deklaration:

```
DbEndTransaction ( ConnectId, Commit ) -> ErrorCode
```

Parameter:

ConnectId	Verbindungs-Identifikator.
Commit	Schritte der Transaktion speichern oder rückgängig machen.
	0 : Alle Schritte der Transaktion werden rückgängig gemacht (Rollback)
	<> 0 : Verarbeitungsschritte werden dauerhaft gespeichert (Commit)
ErrorCode	Ergebnis: Im Fehlerfall eine Fehlernummer, sonst Null.
	= 0 : Kein Fehler.
	< 0 : Fehlernummer.

Beschreibung:

Diese Funktion muss immer paarweise mit der Funktion DbBeginTransaction() aufgerufen werden.

Beispiele:

Siehe auch:

Transaktionen, DbBeginTransaction, DbGetLastErrorText, DbGetLastErrorCode

DbGetLastErrorCode

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Ermittelt die Fehlernummer des letzten aufgetretenen Fehlers des Datenbank-Kits.

Deklaration:

```
DbGetLastErrorCode ( ConnectId, ClearError ) -> ErrorCode
```

Parameter:

ConnectId	Verbindungs-Identifikator.
ClearError	Gibt an, ob der letzte Fehler gelöscht werden soll.
	1 : Fehler löschen
	0 : Fehler nicht löschen
ErrorCode	Ergebnis: Die Fehlernummer des letzten aufgetretenen Fehlers des Datenbank-Kits

Beschreibung:

Mit dieser Funktion kann die Fehlernummer zu dem letzten aufgetretenen Fehler gelesen werden.

Wird für den Parameter "ClearError" eine 1 angegeben, so wird anschließend der Fehlerspeicher gelöscht.

Jede Verbindung hat einen eigenen Fehlerspeicher. Aus diesem Grund ist der Parameter "ConnectId" anzugeben.

Die meisten Kit- Funktionen liefern im Fall eines Fehlers im Ergebnis die Fehlernummer zurück. Fehlernummern sind immer negative Zahlen.

Im Handbuch sind im Abschnitt Fehlercodes die Fehler beschrieben.

Beispiele:

Siehe auch:

Verschiedenes, DbGetLastErrorText

DbGetLastErrorText

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Ermittelt den Fehlertext zum letzten aufgetretenen Fehlers des Datenbank-Kits.

Deklaration:

```
DbGetLastErrorText ( ConnectId, ClearError ) -> TxError
```

Parameter:

ConnectId	Verbindungs-Identifikator.
ClearError	Gibt an, ob der letzte Fehler gelöscht werden soll.
	1 : Fehler löschen
	0 : Fehler nicht löschen
TxError	Fehlertext

Beschreibung:

Mit dieser Funktion kann der Text zu dem letzten aufgetretenen Fehler gelesen werden.

Wird für den Parameter "ClearError" eine 1 angegeben, so wird anschließend der Fehlerspeicher gelöscht.

Jede Verbindung hat einen eigenen Fehlerspeicher. Aus diesem Grund ist der Parameter "ConnectId" anzugeben.

Schlägt die Funktion DbConnect() fehlt, so ist beim Aufruf von DbGetLastError() der Parameter ConnectId =0 anzugeben.

Im Handbuch sind im Abschnitt Fehlercodes die Fehler beschrieben.

Beispiele:

Siehe auch:

Verschiedenes, DbGetLastErrorCode

DbInitialize

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Initialisierung der Einstellungen im Kit. Bestehende Datenbankverbindung werden entfernt.

Deklaration:

```
DbInitialize ( ) -> KitVersion
```

Parameter:

KitVersion	Ergebnis: Nummer der Kit Version.

Beschreibung:

Die Funktion initialisiert das Datenbank- Kit. Es wird ein definierter Anfangszustand im Kit hergestellt. Eventuelle Transaktionen werden mit einem Rollback beendet. Alle Verbindungsobjekte und der interne Fehlerspeicher werden gelöscht.

Es wird empfohlen, diese Funktion immer am Anfang einer Sequenz aufzurufen.

Multithreading: Alle Funktionen des Datenbank-Kits dürfen nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

DbInsert

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Ausführung einer INSERT-Anweisung über mehrere Datenbankzeilen

Deklaration:

```
DbInsert ( ConnectId, TxTableName, GrpInsertValues ) -> Result
```

Parameter:

ConnectId	Verbindungs-Identifikator.
TxTableName	Name der Datenbanktabelle
GrpInsertValues	Die Gruppe enthält Textfelder und Datensätze, die in die Datenbanktabelle eingefügt werden
Result	Ergebnis: Im Fehlerfall eine Fehlernummer, sonst die Anzahl der eingefügten Datenbankzeilen.
	>= 0 : Anzahl der eingefügten Datenbankzeilen.
	< 0 : Fehlernummer.

Beschreibung:

Mit dieser Funktion können Daten aus FAMOS- Objekten in eine Datenbanktabelle eingefügt werden. Die Namen der Datensätze und Textfelder in der Gruppe müssen in der Datenbanktabelle als Spalten vorhanden sein. Es werden alle Elemente, die in der Gruppe sind in die Tabelle eingefügt. Die Datensätze und Textfelder der Gruppe müssen die gleiche Größe haben. Die Größe bestimmt die Anzahl der eingefügten Tabellenzeilen.

Wenn eine Blob- Spalte dabei ist, so darf die Größe der Gruppenelemente nur 1 sein. Der Name des Datensatzes, der der Blob- Spalte entspricht, wird in das eine Feld der Tabelle eingefügt. Haben die Gruppenelemente eine Größe > 1, so werden nur die Daten mit dem ersten Index eingefügt.

Bei einem aufgetretenen Fehler kann der Fehlertext mit der Funktion DbGetLastErrorText() ermittelt werden.

Beispiele:

Es wird eine neue Zeile mit einer Blob- Spalte erzeugt.

```
; --- Wichtig! Es darf nur eine Zeile geladen werden ------------
grpResult = DbSelect(ConnectID,"Select * from Messung  WHERE Id = 1","")
errorcode = DbGetLastErrorCode(ConnectID,0)
if errorcode < 0
   errortext = DbGetLastErrorText(ConnectID,1)
end
; --- Neuen ID Wert für das Einfügen bestimmen ------------------
grpMax = DbSelect(ConnectID,"Select Max(Id) AS MaxID from Messung ","")
maxid = grpMax:MaxID[1]
grpResult:ID[1]=maxid+1
; --- Kanal Sintest1 laden --------------------------------------
FileLoad("Sintest1.dat","",0)
grpResult:KANAL=sintest1
; --- Triggerzeit als Datum speichern ---------------------------
grpResult:Datum[1]=Zeit?(grpResult:KANAL)
grpResult:Name[1] ="Sintest1"
grpResult:Maximum[1]=Max(sintest1)
; --- Kanal wird als Blob in die Tabelle eingefügt --------------
; --- Wichtig! Die Gruppenelemente dürfen nur die Größe 1 haben -
Result = DbInsert(ConnectID,"Messung",grpResult)
if result < 0
   errortext = DbGetLastErrorText(ConnectID,1)
end
```

Siehe auch:

Datenzugriff, DbSelect, DbUpdate1, DbUpdate, DbBeginTransaction, DbEndTransaction, DbGetLastErrorText, DbGetLastErrorCode

DbLoadPanel

Eine Panel-Datei wird geladen und angezeigt.
Die Funktion ist veraltet, statt dessen sollte die neuere Funktion PnLoad() verwendet werden (ab V2022).

Deklaration:

```
DbLoadPanel ( TxDateiname, Null ) -> Erfolg
```

Parameter:

TxDateiname	Name der zu öffnenden Panel-Datei.
Null	Reserviert, immer auf 0 zu setzen
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die angegebene Panel-Datei wird geladen.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird ".panel" angenommen.

Wenn bei dem Dateinamen kein kompletter Pfad angegeben ist, wird die Datei nacheinander in den folgenden Verzeichnissen gesucht:

- Projektverzeichnis: Wenn ein Projekt aktiv ist, wird zunächst im aktuellen Projektverzeichnis gesucht.
- Standardverzeichnis für Panel-Dateien: FAMOS-Voreinstellung für Panele/Dialoge/Sequenzen.

Um ein Panel im Dialog-Modus zu starten, verwenden Sie den Befehl Dialog().

Beispiele:

Eine Panel-Datei wird geladen. Es werden diverse Aktualisierungen durchgeführt, danach wird das Panel gedruckt und wieder geschlossen.

```
err = DbLoadPanel("d:\templates\result.panel", 0)
IF err <> 0
   BoxMessage("Fehler", GetLastError(), "!1")
ELSE
   ; Verschiedene Aktualisierungen
   ; ...
   PnPrint(0)
   PnClose(0)
END
```

Siehe auch:

PnLoad, PnClose, DbClosePanel

DbOption

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Setzen von optionalen Parametern.

Deklaration:

```
DbOption ( ConnectId, TxParametername, TxParameterValue ) -> ErrorCode
```

Parameter:

ConnectId	Verbindungs-Identifikator.
TxParametername	Name des vereinbarten optionalen Parameters
	TimeStampMap : Ist der Parameterwert "yes", werden Date/Time- Spalten in ein Text-Array konvertiert, bei "no" in einen normalen Datensatz. Standardwert = "no".
	NumericAsDouble : Ist der Parameterwert "yes", wird jede numerische Spalte in einen Datensatz im Datenformat 8 Byte reell konvertiert, bei "no" hat der Datensatz das passende Datenformat (siehe Handbuch Abschnitt Konvertierung der Daten). Standardwert = "no".
	CommandTimeOut : Über diesen Parameter wird die Zeit in Sekunden festgelegt, die gewartet werden soll, bis der Versuch einer Befehlsausführung beendet und ein Fehler generiert wird.Der Standardwert ist 30.
	BlobAsChannel : Ist der Parameterwert "yes", wird der Inhalt des Blob-Feldes als Kanal interpretiert, bei "no" als binäres Object(z.B. Bild). Standardwert = "yes".
TxParameterValue	Parameterwert.
ErrorCode	Ergebnis: Im Fehlerfall eine Fehlernummer, sonst Null.
	= 0 : Kein Fehler.
	< 0 : Fehlernummer.

Beschreibung:

Mit dieser Funktion können optionale Parameter pro Verbindung gesetzt werden.

Der Name des optionalen Parameters muss einem vereinbarten Namen entsprechen.

Beispiele:

Siehe auch:

Verschiedenes, DbSelect

DbSelect

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Ausführung einer SELECT Anweisung. Die selektierten Daten werden zu einer FAMOS Gruppe zusammengefaßt und zurückgegeben

Deklaration:

```
DbSelect ( ConnectId, TxSqlStatement, TxTimeColumn ) -> GrpResult
```

Parameter:

ConnectId	Verbindungs-Identifikator.
TxSqlStatement	SQL SELECT Anweisung. Diese Anweisung wird unverändert an das Datenbanksystem übergeben
TxTimeColumn	Falls nicht "" wird der Inhalt dieser Spalte benutzt, um den numerischen Gruppenelementen eine Zeitspur zugeben.
GrpResult	Ergebnis: Diese Gruppe wird mit Textfeldern und Datensätzen befüllt, welche den ausgewählten Spaltennamen entspricht. Im Fehlerfall ist die Gruppe leer.

Beschreibung:

Die Funktion führt die SELECT Anweisung aus, konvertiert die Spalten in Datensätze bzw. Textfelder.

Diese werden zu einer FAMOS Gruppe zusammengefasst und zurückgegeben.

Jede Spalte ergibt ein gleichnamiges Gruppenelement (Datensatz oder Textfeld).

Die Anzahl der eingelesenen Zeilen bestimmt die Größe der Gruppenelemente.

Im Handbuch ist eine Übersicht, wie die Datentypen der Spalten in FAMOS- Objekte konvertiert werden.

Enthält eine Abfrage eine oder mehrere Blob- Spalten, so darf das Ergebnis nur eine Zeile ergeben.

Die Werte der Felder der Blob- Spalten ergeben je einen Datensatz. Ergibt die Abfrage mehrere Ergebniszeilen, so werden die Blob- Spalten nicht konvertiert.

Wird der Parameter TxTimeColumn gesetzt und entspricht er einer numerischen Spalte, so werden alle anderen numerischen Spalten in XY-Datensätze umgewandelt, mit der Zeitspalte als X-Komponente.

Über den optionalen Parameter TimeStampMap = yes wird festgelegt, dass alle Spalten vom Typ Datum bzw. Zeit in ein Textfeld umgewandelt werden. Standardmäßig werden diese Spalten in einen normalen Datensatz im imc Zeitformat konvertiert.

Mit dem optionalen Parameter NumericAsDouble = yes wird festgelegt, dass alle numerischen Spalten in Datensätze mit dem Format 8 Byte reell (double) konvertiert werden.

Um einen eventuell aufgetretenen Fehler zu ermitteln sind im Anschluss die Funktionen DbGetLastErrorCode() und/oder DbGetLastErrorText() zu verwenden.

Beispiele:

Mit dieser Abfrage sollen alle Spalten der Tabelle Messung eingelesen werden.

```
grpResult = DbSelect(ConnectID,"Select * from Messung WHERE Id > 0","")
errorcode=DbGetLastErrorCode(ConnectID,0)
if errorcode < 0
   errortext=DbGetLastErrorText(ConnectID,1)
end
```

Siehe auch:

Datenzugriff, DbOption, DbGetLastErrorText, DbGetLastErrorCode

DbSetActivePanel

Anwendungsbereich: Panele

Das aktive Panel wird festgelegt.

Deklaration:

```
DbSetActivePanel ( TxTitel )
```

Parameter:

TxTitel	Titel des zu aktivierenden Panels.

Beschreibung:

Die Funktion ist nur sinnvoll, wenn mehrere Panels gleichzeitig angezeigt werden. Die Funktionen des Kits beziehen sich stets auf das gerade aktive (d.h. sichtbare) Panel.

Beispiele:

Während mehrere Panele geöffnet sind, wird der Anwender gefragt, ob er das zuletzt geladene Panel drucken möchte. Da der Anwender in der Zwischenzeit manuell ein anderes Panel aktiviert haben könnte, muss sichergestellt werden, dass der folgende Druckbefehl tatsächlich auf das gewünschte Panel wirkt.

Multithreading: Die Funktionen zur Panel-Fernsteuerung dürfen überall aufgerufen werden und wirken global. Das hier selektierte Panel ist also für alle Ausführungs-Threads gültig.

```
err = PnLoad("Egal.Panel")
;...
err = PnLoad("Report.panel")
;...
IF err = 0
   ok = BoxMessage("Report","Ducken ?","?2")
   IF ok = 1
      DbSetActivePanel("Report")
      PnPrint(0)
   END
END
```

Siehe auch:

DbShow, PnLoad

DbSetPanelWindow

Anwendungsbereich: Panele

Setzt die Position des aktiven Panels auf dem Bildschirm.

Deklaration:

```
DbSetPanelWindow ( Status, links, oben, breit, hoch )
```

Parameter:

Status	Fenster-Status
	0 : Minimiert (als Symbol)
	1 : Normal (freischwebend)
	2 : Maximiert (Vollbild)
	3 : Gedockt im Hauptfenster
links	x-Koodinate der linken oberen Ecke
oben	y-Koordinate der linken oberen Ecke
breit	Breite des Fensters
hoch	Höhe des Fensters

Beschreibung:

Mit dieser Funktion kann die Lage und Größe des aktiven Panel-Fensters auf dem Schirm gesteuert werden.

Die angegebenen Fenster-Koordinaten bestimmen die Normal-Position des Fensters im freischwebenden Zustand. Die Angabe erfolgt in Bildschirm-Pixeln.

Wenn [breit] und [hoch] beide 0 sind, bleibt die aktuelle Größe erhalten, nur die Lage wird entsprechend korrigiert. Wenn zusätzlich auch [links] und [oben] 0 sind, bleibt die Normal-Position unverändert.

Beispiele:

Eine Panel-Datei wird geladen. Das Panel wird aus dem Hauptfenster ausgedockt und als frei schwebendes Fenster in der Größe 600x400 in der rechten oberen Ecke des primären Monitors angezeigt.

```
err = PnLoad("d:\templates\result.panel")
IF err = 0
   right = GetSystemInfo("Screen.PrimaryWorkArea", "left") + GetSystemInfo("Screen.PrimaryWorkArea", "width")
   DbSetPanelWindow(1, right-600, 0, 600, 400)
END
```

Siehe auch:

DbShow, PnClose

DbShow

Anwendungsbereich: Panele

Steuert die Sichtbarkeit aller offenen Panel-Fenster.

Deklaration:

```
DbShow ( Auftrag )
```

Parameter:

Auftrag	Kommando-Parameter
	0 : Alle Panele schließen
	1 : Als freie Fenster anzeigen
	2 : Als Symbole anzeigen
	3 : Maximiert anzeigen

Beschreibung:

Mit dieser Funktion kann der Anzeige-Status aller offenen Panel-Fenster gesteuert werden.

Beim Schließen (Auftrag = 0) gehen alle noch nicht gespeicherten Änderungen in den Panelen verloren.

Ab FAMOS 7.0 ist diese Funktion wegen der nötigen Kompatibilität zu älteren FAMOS-Sequenzen noch enthalten, in neu erstellten Sequenzen sollte die Funktion DbSetPanelWindow() verwendet werden, mit der die Sichtbarkeit für jedes Panel separat gesteuert werden kann.

Siehe auch:

PnLoad, PnClose, DbSetPanelWindow

DbSql

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Ausführung einer SQL Anweisung ohne Rückgabe von Daten, d.h. INSERT, UPDATE oder DELETE können ausgeführt werden.

Deklaration:

```
DbSql ( ConnectId, TxSqlStatement ) -> Result
```

Parameter:

ConnectId	Verbindungs-Identifikator.
TxSqlStatement	SQL Anweisung. Diese Anweisung wird unverändert an das Datenbanksystem übergeben
Result	Ergebnis: Im Fehlerfall eine Fehlernummer, sonst die Anzahl der betroffenen Datenzeilen.
	>= 0 : Anzahl der betroffenen Datenzeilen.
	< 0 : Fehlernummer.

Beschreibung:

Mit dieser Funktion können SQL- Anweisungen ausgeführt werden, die keine Ergebnisse zurückliefern (UPDATE, INSERT, DELETE). Der Rückgabewert entspricht der Anzahl der betroffenen Zeilen. Es lassen sich aber auch DDL Kommandos wie z.B. "ALTER TABLE Messung ADD MyCol INTEGER" ausführen. Bei einem DDL- Kommando ist der Rückgabewert bei erfolgreicher Ausführung 0.
Bei einem aufgetretenen Fehler kann der Fehlertext mit der Funktion DbGetLastErrorText() ermittelt werden.

Beispiele:

Alle Werte in der Spalte "Status" der Tabelle "Messung" werden auf "Ready" gesetzt.

```
result =DbSql(ConnectId,"Update Messung set Status ='Ready'")
if result < 0
   errortext=DbGetLastErrorText(ConnectId,1)
end
```

Der Tabelle "Messung" wird die Spalte "MyCol" mit dem Datentyp Integer zugefügt.

```
result =DbSql(ConnectId,"ALTER TABLE Messung ADD MyCol Integer")
if result < 0
   errortext=DbGetLastErrorText(ConnectId,1)
end
```

Siehe auch:

Datenzugriff, DbBeginTransaction, DbEndTransaction, DbInsert, DbUpdate1, DbUpdate, DbGetLastErrorText, DbGetLastErrorCode

DbUpdate

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Mehrfache Ausführung einer UPDATE Anweisung

Deklaration:

```
DbUpdate ( ConnectId, TxSqlStatement, GrpUpdateValues ) -> Result
```

Parameter:

ConnectId	Verbindungs-Identifikator.
TxSqlStatement	SQL UPDATE Anweisung
GrpUpdateValues	Enthält Daten zum Aktualisieren und für die WHERE Bedingung.
Result	Ergebnis: Im Fehlerfall eine Fehlernummer, sonst die Anzahl der aktualisierten Datenbankzeilen.
	>= 0 : Anzahl der aktualisierten Datenbankzeilen.
	< 0 : Fehlernummer.

Beschreibung:

Die Funktion führt die UPDATE-Anweisung entsprechend der Größe der übergebenen Gruppenelemente n-mal aus. Ein Index läuft in einer Schleife von 1 bis zur Größe der Gruppenelemente. Die Werte der Gruppenelemente, die durch den Index adressiert sind, ersetzen die Platzhalter in der UPDATE-Anweisung. Die Update-Anweisung wird ausgeführt.

Die UPDATE-Anweisung muss komplett als TxSqlStatement angegeben sein. Als Platzhalter für die zu ersetzenden Werte aus den Gruppenelementen ist das Fragezeichen **?+Gruppenelementname** zu verwenden. Die Platzhalter im SET-Teil und in der WHERE- Bedingung werden durch die Daten aus der Gruppe GrpUpdateValues ersetzt. Ist in der UPDATE- Anweisung eine Blob- Spalte, so wird nur die 1. Zeile in der Datenbanktabelle aktualisiert.

Bei einem aufgetretenen Fehler kann der Fehlertext mit der Funktion DbGetLastErrorText() ermittelt werden.

Beispiele:

Die Spalten ID, NAME und DATUM werden eingelesen. Die Gruppenelemente haben eine Größe von 13. Jetzt werden die einzelnen Namen verändert. Die veränderte Gruppe aktualisiert abschließend die Datenbanktabelle.

Die UPDATE-Anweisung wird 13 mal ausgeführt. Bei jedem UPDATE werden die Platzhalter durch die entsprechenden Daten aus den Gruppenelementen ersetzt. Das Ergebnis der Funktion ist 13.

```
grpResult =DbSelect(ConnectID,"Select Id, Name,Datum from Messung","")
; --- Neue Namen für das Aktualisieren bilden ---------------
for i= 1 to lang?(grpResult:ID)  Step 1
   messung=grpResult:Name[i] +"_"+ TForm(grpResult:ID[i],"")
   grpResult:Name[i]=messung
end
updatestatement="update Messung set Name=?Name where Id =?Id "
result=DbUpdate(ConnectId,updatestatement,grpResult)
if result < 0
   errortext=DbGetLastErrorText(ConnectID,1)
end
```

Siehe auch:

Datenzugriff, DbUpdate1, DbBeginTransaction, DbEndTransaction, DbGetLastErrorText, DbGetLastErrorCode

DbUpdate1

Anwendungsbereich: Databank-Fernsteuerung

Benötigtes Erweiterungs-Kit: (imc Database Kit)

Ausführung einer UPDATE Anweisung

Deklaration:

```
DbUpdate1 ( ConnectId, TxSqlStatement, GrpUpdateValues, SampleIndex ) -> Result
```

Parameter:

ConnectId	Verbindungs-Identifikator.
TxSqlStatement	SQL UPDATE Anweisung
GrpUpdateValues	Enthält Daten zum Aktualisieren und für die WHERE Bedingung
SampleIndex	Nur mit den Werten aus den Gruppenelementen, die durch den SampleIndex adressiert werden, erfolgt eine Aktualisierung der Datenbankwerte (1...).
Result	Ergebnis: Im Fehlerfall eine Fehlernummer, sonst die Anzahl der aktualisierten Datenbankzeilen.
	>= 0 : Anzahl der aktualisierten Datenbankzeilen.
	< 0 : Fehlernummer.

Beschreibung:

Mit dieser Funktion wird eine UPDATE-Anweisung ausgeführt. Die UPDATE-Anweisung muss komplett als TxSqlStatement angegeben sein. Als Platzhalter für die zu ersetzenden Werte aus den Gruppenelementen ist das Fragezeichen **?+Gruppenelementname** zu verwenden. Die Platzhalter im SET-Teil und in der WHERE- Bedingung werden durch die Daten aus der Gruppe GrpUpdateValues ersetzt.

Für die Update-Anweisung werden nur die Daten aus den Gruppenelementen verwendet, die durch den SampleIndex adressiert sind. In der Gruppe können mehr Gruppenelemente vorhanden sein, als für die Update-Anweisung benötigt werden. Es werden nur die Gruppenelemente benutzt, die in der Update-Anweisung angegeben sind.

Bei einem aufgetretenen Fehler kann der Fehlertext mit der Funktion DbGetLastErrorText() ermittelt werden.

Beispiele:

Es sollen die Werte der Spalte MAXIMUM auf 1000 gesetzt werden, wo der Wert der Spalte NAME auf 1x endet.

Nach dem Einlesen der Spalten wird der Wert vom Maximum mit dem SampleIndex = 1 auf 1000 gesetzt. In der UPDATE- Anweisung ist ein Platzhalter im SET-Teil. Die WHERE- Bedingung "Name LIKE '%1_' " filtert 4 Zeilen, die aktualisiert werden. Das Ergebnis der Funktion ist 4.

```
grpResult =DbSelect(ConnectID,"Select Id,Nname,Datum,Maximum from Msessung", "")
grpResult:Maximum[1]=1000;
updatestatement="update Messung set Maximum=?Maximum where Name LIKE '%1_' "
result=DbUpdate1(ConnectId,updatestatement,grpResult,1)
if result < 0
   errortext=DbGetLastErrorText(ConnectID,1)
end
```

Der Wert der Spalte MAXIMUM der Zeile mit dem ID-Wert 11 soll auf 0 gesetzt werden.

```
grpResult =DbSelect(ConnectID,"Select Id,Name,Datum,Maximum from Messung", "")
;Der 11. Wert vom Datensatz MAXIMUM wird auf 0 gesetzt.
grpResult:Maximum[11]=0;
updatestatement="update Messung set Maximum=?Maximum where Id =?Id"
;Die UPDATE-Anweisung wird mit den Daten, die durch den SampleIndex 11 adressiert sind, ausgeführt.
result=DbUpdate1(ConnectId,updatestatement,grpResult,11)
if result < 0
   errortext=DbGetLastErrorText(ConnectID,1)
end
```

Siehe auch:

Datenzugriff, DbUpdate, DbBeginTransaction, DbEndTransaction, DbGetLastErrorText, DbGetLastErrorCode

DDEInq

Verfügbar ab: Professional Edition

Über DDE wird von einer anderen Applikation ein Text, Einzelwert oder Datensatz abgefragt.
Alternativer Name: **DDEFrage**

DDE ist eine veraltete Technologie und wird von künftigen Windows- und MS-Office-Versionen möglicherweise nicht mehr unterstützt. Zur Kommunikation mit EXCEL sollten daher nach Möglichkeit die Funktionen aus der EXCEL-Funktionsgruppe, wie z.B. XLWbOpen() und XLSetValues(), verwendet werden.

Deklaration:

```
DDEInq ( TxApplikation, TxThema, TxElement, EwDatentyp ) -> Daten
```

Parameter:

TxApplikation	Der Name des Programms, von dem Daten abgefragt werden sollen.
TxThema	Das Thema (Topic) der Konversation.
TxElement	Abzufragendes Element
EwDatentyp	Datentyp
	1 : Text
	2 : Einzelwert
	3 : Datensatz
Daten	Die von der Applikation gelesenen Daten. Der Typ richtet sich nach [EwDatentyp].

Beschreibung:

Fragt mittels DDE aus einem anderen Programm einen Text, Wert oder Datensatz ab.

DDE (Dynamic Data Exchange) ist ein Kommunikations-Mechanismus unter Windows zum Austausch von Daten oder Kommandos, der von vielen Windows-Applikationen unterstützt wird.

Über die verfügbaren Themen und die Bezeichnung der Elemente informieren Sie sich bitte in der Dokumentation der Zielapplikation.

- Wenn nach Text gefragt wird, wird nur nach ASCII-Format gefragt.
- Wenn nach einem Text gefragt wird, darf der Text nicht länger als 255 Zeichen sein.
- Wenn nach einem Einzelwert oder einem Datensatz gefragt wird, wird zuerst eine Anfrage nach dem FAMOS-DDE-Format durchgeführt. Wenn dieses Format nicht akzeptiert wird, wird die Anfrage nach Daten im ASCII-Format wiederholt.
- Wenn von der anderen Applikation Daten im ASCII-Format gelesen werden, dürfen zwischen den Zahlen beliebige Trennzeichen stehen. Die Zahlen selbst dürfen reelle Zahlen sein, die ggf. mit einem Dezimalpunkt geschrieben werden dürfen. Ein Dezimalkomma ist nicht erlaubt.
- Antwortet die andere Applikation nicht oder stellt keine Daten bereit, wird eine Fehlermeldung erzeugt.
- Es wird stets eine komplette DDE-Konversation durchgeführt, die aus Initialisierung, Abfrage und Ende besteht.
- Wenn CwUpdateEnable(0) bzw. CvUpdate(0) aufgerufen wurde, ist keine DDE-Kommunikation möglich!

Beispiele:

Eine existierende Excel-Datei mit dem Tabellenblatt "Tabelle1" wird geöffnet.

Das Element in der 1.Zeile, 2.Spalte wird abgefragt und als Text zurückgeliefert.

```
Execute("Excel.exe","c:\temp\1.xlsx", "open", 0, 3)
TxZelle = DDEInq("EXCEL","Tabelle1","Z1S2", 1)
```

Hinweis: Zur Fernsteuerung von EXCEL sind die Funktionen des Excel-Kits (XlWbOpen, XlSetValues, XlGetValues ...) im Allgemeinen besser geeignet.

```
Dat1 = DDEInq("EXCEL","Tab1","Z1S1:Z100S1", 3)
```

Liest aus der 1.Spalte in der Excel-Tabelle "Tab1" maximal 100 Werte.

Siehe auch:

DDESend, DDESet

DDESend

Verfügbar ab: Professional Edition

Eine Kommando wird über DDE an eine andere Applikation gesendet.
Alternativer Name: **DDESende**

DDE ist eine veraltete Technologie und wird von künftigen Windows- und MS-Office-Versionen möglicherweise nicht mehr unterstützt. Zur Kommunikation mit EXCEL sollten daher nach Möglichkeit die Funktionen aus der EXCEL-Funktionsgruppe, wie z.B. XLWbOpen() und XLSetValues(), verwendet werden.

Deklaration:

```
DDESend ( TxApplikation, TxThema, TxKommando ) -> EwStatus
```

Parameter:

TxApplikation	Der Name des Programms, an das das Kommando gesendet werden sollen.
TxThema	Das Thema (Topic) der Konversation.
TxKommando	Das Kommando selbst, das gesendet werden soll.
EwStatus	0, wenn das Kommando ausgeführt wurden. -1, wenn die andere Applikation nicht angesprochen werden kann. Sonst: Rückgabewert der anderen Applikation.

Beschreibung:

Einer anderen Applikation wird mittels DDE ein Kommando zugesendet.

DDE (Dynamic Data Exchange) ist ein Kommunikations-Mechanismus unter Windows zum Austausch von Daten oder Kommandos, der von vielen Windows-Applikationen unterstützt wird.

Über die verfügbaren Themen und die Syntax des Kommandotextes informieren Sie sich bitte in der Dokumentation der Zielapplikation.

- Die Syntax des Kommandos wird von der empfangenen Applikation vorgegeben.
- Beachten Sie den Rückgabewert, um festzustellen, ob die andere Applikation das Kommando empfangen und ausgeführt hat.
- Es wird stets eine komplette DDE-Konversation durchgeführt, die aus Initialisierung, Abfrage und Ende besteht.
- Ist die andere Applikation beschäftigt, wird die Übertragung wiederholt, solange bis die andere Applikation nicht mehr beschäftigt ist.
- Wenn CwUpdateEnable(0) bzw. CvUpdate(0) aufgerufen wurde, ist keine DDE-Kommunikation möglich!

Beispiele:

```
error = DDESend("EXCEL","SYSTEM","[NEU]")
IF error
   PAUSE Kommando nicht ausgeführt!
END
```

An EXCEL wird ein Kommando gesendet. Dieses Kommando veranlasst EXCEL, eine neue Tabelle zu öffnen.

```
TxCommand = "[Öffnen(""Tabelle1.xls"")]"
eror = DDESend("EXCEL", "SYSTEM", TxCommand)
```

Excel wird angewiesen, die Datei "Tabelle1.xls" zu öffnen.

Hinweis: Zur Fernsteuerung von EXCEL sind die Funktionen des Excel-Kits (XlWbOpen, XlSetValues, XlGetValues ...) im Allgemeinen besser geeignet.

Siehe auch:

DDEInq, DDESet

DDESepar

Verfügbar ab: Professional Edition

Das Trennzeichen bei einer Übertragung von ASCII-Texten mit der Funktion DDESet() wird definiert
Alternativer Name: **DDETrenn**

Deklaration:

```
DDESepar ( TxZeichen )
```

Parameter:

TxZeichen	Ein Text, der das (die) Trennzeichen enthält.

Beschreibung:

Die Funktion DDESet() kann normale Datensätze im ASCII-Format übertragen. Die einzelnen Zahlenwerte müssen dabei durch irgendein sinnvolles Zeichen voneinander getrennt werden. Dieses Zeichen oder auch eine ganze Zeichenfolge kann mit den Funktionen DDESepar() oder SetOption() definiert werden.

Der übertragene ASCII-Text hat diese Form:

```
Zahl Trennzeichen Zahl Trennzeichen .. Zahl
```

- Sie können das Trennzeichen sowie weitere Formatoptionen auch über den Dialog "Optionen"/"DDE" oder die Funktion SetOption() einstellen.
- Die Zeichenfolge zur Trennung der Zahlen sollte i. a. keine Zahlen enthalten.
- Die Zeichenfolge darf maximal 4 Zeichen lang sein.
  Multithreading: Die Funktion wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
DDESepar(SvToChar(0x09))
; oder besser:
DDESepar("~009")
```

Als Trennzeichen wird ein TAB-Zeichen (ASCII-Darstellung: 9hex) gewählt. Damit kann z.B. eine Zeile nach EXCEL übertragen werden.

```
DDESepar(TAdd(SvToChar(0x0d), SvToChar(0x0a)))
; oder besser
DDESepar("~013~010")
```

Als Trennzeichen wird eine Carriage Return / Line Feed-Kombination gewählt. Damit kann z.B. eine Spalte nach EXCEL übertragen werden.

Siehe auch:

DDESet, DDEInq, SetOption

DDESet

Verfügbar ab: Professional Edition

Der Inhalt einer Variablen wird über DDE an eine andere Applikation übertragen.
Alternativer Name: **DDESetze**

DDE ist eine veraltete Technologie und wird von künftigen Windows- und MS-Office-Versionen möglicherweise nicht mehr unterstützt. Zur Kommunikation mit EXCEL sollten daher nach Möglichkeit die Funktionen aus der EXCEL-Funktionsgruppe, wie z.B. XLWbOpen() und XLSetValues(), verwendet werden.

Deklaration:

```
DDESet ( TxApplikation, TxThema, TxElement, Daten, EwDatentyp ) -> EwStatus
```

Parameter:

TxApplikation	Der Name des Programms, an das die Daten übertragen werden sollen.
TxThema	Das Thema (Topic) der Konversation.
TxElement	Die Bezeichnung des Elementes (der Variable) im empfangenden Programm, das die Daten erhält.
Daten	Die zu übertragenen Daten: Einzelwert, normaler Datensatz oder Text.
EwDatentyp	Datentyp
	1 : Daten als Text (ASCII-Format) übertragen
	2 : Daten im imc FAMOS-Format übertragen (auch bei Datentyp=3)
EwStatus	0, wenn Daten übertragen wurden. -1, wenn die andere Applikation nicht angesprochen werden kann. Sonst: Rückgabewert der anderen Applikation.

Beschreibung:

Setzt mittels DDE in einem anderen Programm einen Text, Wert oder Datensatz ein.

DDE (Dynamic Data Exchange) ist ein Kommunikations-Mechanismus unter Windows zum Austausch von Daten oder Kommandos, der von vielen Windows-Applikationen unterstützt wird.

Über die verfügbaren Themen und die Bezeichnung der Elemente informieren Sie sich bitte in der Dokumentation der Zielapplikation.

- Benutzen Sie die Funktion DDESepar() oder SetOption(), um für eine Übertragung von normalen Datensätzen im ASCII-Format das Trennzeichen zu definieren.
- Wenn Zahlenwerte im ASCII-Format übertragen werden, können Sie das Format über den Dialog "Optionen"/"DDE" näher spezifizieren.
- Beachten Sie den Rückgabewert, um festzustellen, ob die andere Applikation die Daten empfangen hat.
- Es wird stets eine komplette DDE-Konversation durchgeführt, die aus Initialisierung, Abfrage und Ende besteht.
- Wenn CwUpdateEnable(0) bzw. CvUpdate(0) aufgerufen wurde, ist keine DDE-Kommunikation möglich!

Beispiele:

Eine existierende Excel-Datei mit dem Tabellenblatt "Tabelle1" wird geöffnet.

Die Zelle in der 1.Zeile, 2.Spalte wird auf "Hallo Excel" gesetzt.

```
Execute("Excel.exe", "c:\temp\Mappe1.xlsx", "open",0,3)
ret = DDESet("EXCEL", "Tabelle1", "Z1S2", "Hallo Excel", 1)
```

Hinweis: Zur Fernsteuerung von EXCEL sind die Funktionen des Excel-Kits (XlWbOpen, XlSetValues, XlGetValues ...) im Allgemeinen besser geeignet.

Trägt in der Excel-Tabelle "Tab1" die Werte des Datensatzes "daten" in der 1.Spalte ein. Vor der Übertragung werden die Daten in das ASCII-Format überführt.

```
ret = DDESet("EXCEL", "Tabelle1", "Z2S1:Z100S1", daten, 1)
```

Siehe auch:

DDEInq, DDESepar, DDESend, SetOption

DEFAULT

Leitet den 'Sonst'-Zweig in einer durch den Befehl SWITCH eingeleiteten Fallunterscheidung (mehrfache Verzweigung) ein. Der nachfolgende Block wird ausgeführt, wenn keine der vorherigen CASE-Bedingungen erfüllt ist.

Deklaration:

```
DEFAULT
```

Beschreibung:

Das Ende der zu diesem DEFAULT gehörigen Anweisungen wird durch den zum SWITCH gehörenden END-Befehl bestimmt.

Beispiele:

Zu einem Wert, der normalerweise im Bereich von 0 bis 100 liegt, wird ein beschreibender Text gebildet. Liegt er außerhalb dieses Bereichs, wird eine Fehlermeldung angezeigt.

```
SWITCH Round(value, 1)
CASE 0 TO 48
    Tx = "Lower half"
CASE 49,50,51
    Tx = "Center"
CASE 52 To 100
   Tx = "Upper half"
DEFAULT
   PAUSE Invalid Value
END
```

Siehe auch:

SWITCH, CASE, IF

DELETE

Variable entfernen
Alternativer Name: **ENTFERNEN**

Deklaration:

```
DELETE Variablenname
```

Parameter:

Variablenname	Name der zu entfernenden Variablen

Beschreibung:

Die als Parameter angegebene Variable wird aus der Variablenliste entfernt.

Sie können im Variablennamen Jokerzeichen (Wildcards) angeben, um eine Reihe von Variablen zu löschen. Das Jokerzeichen '?' steht dabei für genau ein beliebiges Zeichen, das Jokerzeichen '*' für eine unbestimmte Anzahl von beliebigen Zeichen.

```
DELETE *
```

Alle Variablen werden entfernt.

```
DELETE ??
```

Alle Variablen, deren Name genau 2 Zeichen lang sind, werden entfernt

```
DELETE *1a*
```

Alle Variablen, in denen die Zeichenfolge '1a' (auch zu Beginn oder am Ende) vorkommt, werden entfernt.

```
DELETE a*:kanal?
```

Aus allen Datengruppen, deren Name mit einem 'a' beginnt, werden all jene Kanäle gelöscht, die den Namen 'Kanal', gefolgt von einem beliebigen Zeichen, besitzen.

```
DELETE temp?:@Test_2022_12_01
```

Es werden alle Kanäle gelöscht, die zu der Messung "Test_2022_12_01" gehören und deren Name aus "temp" und einem weiteren Zeichen gebildet wird.

Beispiele:

Eine Variable wird erzeugt, gesichert und danach entfernt:

```
Var = 2
FileSave("Ergebnis.dat", "", 0, Var)
DELETE Var
```

Ein Kanal einer Datengruppe wird gelöscht:

```
DELETE Gruppe:Kanal1
```

Siehe auch:

SHOW, RENAME

DeqCalc

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Die eigentliche Berechnung

Deklaration:

```
DeqCalc ( ) -> Restanzahl
```

Parameter:

Restanzahl	Restanzahl, noch verbleibende Anzahl von Stützstellen zur Berechnung

Beschreibung:

Diese Funktion führt die eigentliche Berechnung durch. Zuvor wurden alle Integrationsvariablen, Konstanten, die Zeitbasis und Eingangssignale festgelegt.

Beim ersten Aufruf dieser Funktion erfolgt die Aufbereitung und Prüfung, die vorher i.a. nicht erfolgen kann, da die gesamte Definition der Aufgabe noch unvollständig war.

Die Funktion beginnt beim ersten Aufruf mit der Berechnung. Falls die Berechnung nach einer gewissen Zeit nicht abgeschlossen ist, kehrt die Funktion dennoch zurück.

Die Funktion wird üblicherweise in einer Schleife aufgerufen, bis die Berechnung komplett fertig ist. Damit wird ein Abbrechen möglich.

Die Berechnung kann insgesamt sehr lange dauern.

Ein einziger Aufruf der Funktion dauert ca. 0.5s, wenn die gesamte Berechnung noch nicht abgeschlossen. Die Zeit kann auch mal überschritten werden, wenn die Berechnung eines nächsten Stützpunktes sehr aufwendig ist.

Wenn die Berechnung abgeschlossen ist, gibt die Funktion 0 zurück.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Nach allen benötigten Initialisierungen erfolgt die eigentliche Berechnung mit DeqCalc(), bevor die Abfrage der Ergebnisse mit DeqResult() erfolgt.

Beispiele:

Berechnung mit Kontrolle der noch verbleibenden zu berechnenden Stützstellen.

```
local Remain = 1
while Remain > 0
   Remain = DeqCalc()
end
```

Einfache Schleife

```
while DeqCalc() > 0
end
```

DeqConst

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Festlegen von Konstanten

Deklaration:

```
DeqConst ( Wert, Formelzeichen )
```

Parameter:

Wert	Wert der Konstante
Formelzeichen	Formelzeichen für diese Konstante, z.B. "C1"

Beschreibung:

In den Formeln y'= können Konstante direkt mit ihrem Zahlenwert eingetragen werden. Alternativ auch über ein Symbol. Der Wert der Konstanten wird mit dieser Funktion festgelegt und kann dann bequem aus einer Famos-Variable genommen werden.

Die Funktion wird für jede in einer Formel benutzte Konstante einmal aufgerufen.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Vor der eigentlichen Berechnung mit DeqCalc() erfolgen bei Bedarf die Aufruf von DeqConst().

Beispiele:

ungedämpftes Pendel

```
DeqInit(2, 0, 0)
local Length = 1.5
DeqConst(Length, "Length")
DeqConst(9.81, "g")
DeqX(ramp(0,0.01,1000))
DeqY("phi'", 3 )
DeqY("phi''=-(g/Length)*sin(phi)", 0 )
```

Alternative ohne Konstante

```
DeqInit(2, 0, 0)
DeqX(ramp(0,0.01,1000))
DeqY("phi'", 3 )
DeqY("phi''=-(9.81/1.5)*sin(phi)", 0 )
```

DeqError

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Abfrage von Fehlern

Deklaration:

```
DeqError ( ) -> Fehler
```

Parameter:

Fehler	Fehler (0 OK, 1 Genauigkeit nicht erfüllt, 2 Unendlich)

Beschreibung:

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Nach der eigentlichen Berechnung mit DeqCalc() erfolgt die Abfrage auf Fehler mit DeqError().

Beispiele:

Werte gehen gegen unendlich, Error=2

```
DeqInit(1, 0, 0)
DeqX(ramp(0,1,1000))
DeqY("y'=y", 1)
while DeqCalc() > 0
end
Error = DeqError()
```

Überprüfung auf Wertebereich und Einhalten der Genauigkeit

```
DeqInit(1, 0, 0)
DeqX(ramp(0,1,100))
DeqY("y'=-0.1*y", 1)
while DeqCalc() > 0
end
Error = DeqError()
Verify(Error=0)
```

DeqFinish

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Abschluss der Auswertung, Speicherfreigabe

Deklaration:

```
DeqFinish ( )
```

Parameter:

Beschreibung:

Die Funktion gibt sämtlichen Speicher frei, der in Zusammenhang mit der Berechnung benötigt wurde.

Der Aufruf ist empfohlen, aber nicht zwingend nötig.

Mit dem Beginn einer neuen Berechnung mittels DeqInit() wird sämtlicher Speicher einer vorherigen Auswertung freigegeben.

Nach dem Aufruf der Funktion können keine weiteren Deq*() Funktionen aufgerufen werden. Lediglich mit DeqInit() kann eine ganz neue Auswertung begonnen werden.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Der Abschluss der Auswertung erfolgt mit DeqFinish().

Beispiele:

Abklingfunktion, Speicherfreigabe am Ende

```
DeqInit(1, 0, 0)
DeqY("y'=-0.1*y", 1, 1e-7, 1e-8)
DeqX(ramp(0,0.3, 300)))
while DeqCalc() > 0
end
Y = DeqResult("y")
DeqFinish()
```

DeqInit

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Initialisierung

Deklaration:

```
DeqInit ( [Anzahl_Y] [, Anzahl_U] [, Verfahren] )
```

Parameter:

Anzahl_Y	Anzahl der Gleichungen des Systems (optional , Standardwert: 1)
Anzahl_U	Anzahl der Eingangsdatensätze (optional , Standardwert: 0)
Verfahren	Nach welchem Verfahren wird berechnet? (optional , Standardwert: 0)
	0 : RK 45: explizites Runge-Kutta 5. Ordnung, Dormand-Prince; eingebettete Methode
	1 : Mix aus implizitem und explizitem Runge-Kutta. Abschnittsweise Entscheidung für das wahrscheinlich schnellere Verfahren.
	2 : RK 34, implizites Runge-Kutta 4. Ordnung, GRK4T Methode, ein Rosenbrock-Wanner (ROW) Verfahren mit Koeffizienten nach Kaps-Rentrop
	3 : RK 4, klassisches Runge-Kutta 4. Ordnung; k1/6+k2/3+k3/3+k4/6

Beschreibung:

Die Lösung des Systems von Differentialgleichungen erfolgt über mehrere Funktionsaufrufe. Stets wird mit DeqInit() begonnen.

Form der Differentialgleichungen

Jeder der Gleichungen des Systems von gewöhnlichen Differentialgleichungen hat die Form y'=f(x,y,u)

y ist die zu integrierende Variable. Jede der einzelnen Gleichungen des Systems muss auf die Form "y'=..." gebracht werden.

u sind die Eingangsdaten. In der Gleichung sind das meist bekannte Zeitfunktionen, die hier über einen bestehenden Zeitdatensatz vorgegeben werden.

x ist die Zeit, die unabhängige Koordinate. Alle y sind Funktionen der Zeit, also y=y(x)

Als Alternative ist M*y'=f(x,y,u) mit Massenmatrix ebenfalls möglich. y bezeichnet dabei den Vektor mit den zu integrierenden Variablen. f ist selbst ein Vektor.

Verfahren

Das Verfahren arbeitet mit automatischer Schrittweitensteuerung, die aber bei einer Verfeinerung um den Faktor von ca. 10000 endet. Falls das nicht ausreicht, wird ein Verlust an Rechengenauigkeit vermerkt.

Eine kleinere Schrittweite erhöht den Rechenaufwand.

Verfahren=0 ist meist die beste Wahl.

Verfahren=2, implizites Runge-Kutta i.a. nur sinnvoll bei sehr steifen Systemen. Dieses Verfahren unterstützt nicht alle Optionen.

Verfahren=3 (RK 4) benutzt zur Schrittweitensteuerung die Richardson-Extrapolation (Vergleich des Ergebnisses bei halbierter Schrittweite).

Ein Aufruf mit einer Gruppe als Parameter ist nicht möglich.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Prinzipieller Ablauf der Funktionsaufrufe s.u. Beispiel.

Wenn schwerwiegende Fehler im Ablauf auftreten, wie etwa nicht ausreichend Speicherplatz, muss meist die ganze Abfolge mit DeqInit() beginnend neu gestartet werden.

Beispiele:

Einmassenschwinger (SDOF). Gegen ist der Kraftverlauf. Gesucht sind Weg, Geschwindigkeit und Beschleunigung.

Die Gleichung lautet m*y'' + c*y' + k*y = F; Mit m Masse, y Weg, F Kraft; c=2*damp*omega; k=omega^2;omega=f0*PI2

Umformung ergibt:v' = u - 2*damp*omega*v - omega^2*y ; und y'=v; mit u = F/m

```
Force_m=1/(0.1+ramp(0,1e-3,1000))
DeqInit(2, 1, 0)
DeqInput(Force_m,"u")
DeqConst(20, "f0")
DeqConst(0.1, "damp")
DeqY("y'=v", 0, 1e-5, 1e-7, "m" )
DeqY("v' = u-2*damp*(f0*PI2)*v-((f0*PI2)^2)*y", 0.02, 1e-5, 1e-8, "m/s", 1 )
while DeqCalc() > 0
end
Displacement = DeqResult("y")
Speed = DeqResult("v")
Acceleration = DeqResult("v'")
```

Prinzipieller Ablauf

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Danach werden die Formeln der Gleichungen, Eingangsdaten usw. vorgegeben. Dabei muss mindestens ein Aufruf von DeqY() erfolgen. Außerdem muss mindestens ein Aufruf von DeqInput() oder DeqX() erfolgen. Danach erfolgt die Berechnung mit DeqCalc(). Danach werden die Ergebnisse abgefragt. Optionale Fehlerabfragen mit DeqError() und Speicherfreigaben mit DeqFinish() am Ende.

```
DeqInit()
DeqInput() or DeqX()
DeqY()
while DeqCalc() > 0
end
DeqResult()
DeqError()
DeqFinish()
```

DeqInput

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Festlegen einer Eingangsfunktion

Deklaration:

```
DeqInput ( Eingangssignal [, Formelzeichen] )
```

Parameter:

Eingangssignal	Datensatz mit dem Eingangssignal U
Formelzeichen	Formelzeichen für dieses Eingangssignal, z.B. u1. Wenn nicht angegeben, werden u1, u2 etc. angenommen. (optional , Standardwert: "")

Beschreibung:

Das Eingangssignal wird als treppenstufig interpoliert angesehen.

In der Funktion DeqInit() wird angegeben, wie viele Eingangssignale benutzt werden. Für jedes Eingangssignal wird die Funktion DeqInput() einmal aufgerufen

Alle Eingangssignale (sowie auch der Datensatz für X) müssen dieselbe Zeitbasis haben, also gleiche Startzeit, gleiche Abtastzeit, gleiche Länge.

Wird DeqX() nicht aufgerufen, also ohne Datensatz für X, dann gilt: Startzeit, Abtastzeit und Länge legen die Stützstellen der Ergebnisdaten fest.

Die Abtastzeit muss angemessen für das zu berechnende Problem vorgegeben werden. Ist sie zu groß, braucht der Algorithmus zu viel Zeit mit Schrittweitenverfeinerung.

Der angegebene Datensatz ist äquidistant. Er kann keine Events und keine Segmente haben.

Ein Aufruf mit einer Gruppe als Parameter ist nicht möglich.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Vor der eigentlichen Berechnung mit DeqCalc() muss mindestens ein Aufruf von DeqInput() oder DeqX() erfolgen.

Beispiele:

SDOF. Vorgabe der Kraft F. Eine Famos-Variable existiert bereits, die den Kraftverlauf vorgibt. In der Formel soll die Kraft mit F bezeichnet werden.

```
Force=ramp(0,1e-3,1000)
DeqInit(2, 1, 0)
DeqInput(Force, "F")
DeqY("y'", 0.0)
DeqY("y''=F/8-10*y'-20000*y", 1)
```

2 Eingangssignale

```
hx=ramp(0,1e-3,1000)
gx=ramp(0,1e-3,1000)*2
DeqInit(1, 2, 0)
DeqInput(hx, "h")
DeqInput(gx, "g")
DeqY("y'=y-0.1*(h-g)", 1.0)
```

DeqMassMatrix

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Festlegen eines Elements der Massenmatrix

Deklaration:

```
DeqMassMatrix ( Element, Zeile, Spalte )
```

Parameter:

Element	Zahlenwert (Einzelwert) oder frei definierbare Formel
Zeile	Zeile, von 1 bis Anzahl_Y aus DeqInit()
Spalte	Spalte, von 1 bis Anzahl_Y aus DeqInit()

Beschreibung:

Das System von gewöhnlichen Differentialgleichungen hat bei Verwendung einer quadratischen Massenmatrix M die Form:

M*y'=f(x,y,u)

M hat häufig die Bedeutung der Massenmatrix.

Sobald die Funktion einmal aufgerufen wird, wird eine Massenmatrix vor-initialisiert: Mit 1 auf der Diagonalen, sonst 0. Die Funktion wird nur für die wenigen Elemente der Matrix aufgerufen, die davon abweichen.

Das Element kann als bekannter Zahlenwert angegeben werden, z.B. 7.7.

Das Element kann alternativ als Formel vorgegeben werden. Die Formel kann auf einen konstanten Wert führen. Sie darf aber auch die anderen Größen enthalten, die in f(x,y,u) auftauchen. In der Formel wird der Term angegeben, der das Element definiert.

Die Funktion wird für jedes Element der Matrix einmal aufgerufen. Das ist aber nur nötig, wenn das Element vom entsprechenden Element der Einheitsmatrix abweicht.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Vor der eigentlichen Berechnung mit DeqCalc() erfolgen bei Bedarf die Aufrufe von DeqMassMatrix().

Beispiele:

Laufkatze mit pendelnder Last gedämpft

Massenmatrix:

[	1	0	0	0	]
[	0	1	0	0	]
[	0	0	mLd+mcr	Lng*mLd*cos(phi)	]
[	0	0	Lng*mLd*cos(phi)	JLd+mLd*Lng^2	]

```
DeqInit(4, 0, 0)
DeqConst(9.81, "g"); m /s^2
DeqConst(150, "mcr"); mass crab, kg
DeqConst(900, "mLd"); mass load, kg
DeqConst(2, "Lng"); distance between crab and load, m
DeqConst(120, "JLd"); kgm^2
DeqConst(200, "c") ; damper
DeqMassMatrix("mLd+mcr", 3, 3)
DeqMassMatrix("JLd+mLd*Lng^2", 4, 4)
DeqMassMatrix("Lng*mLd*cos(phi)", 3, 4)
DeqMassMatrix("Lng*mLd*cos(phi)", 4, 3)
DeqX(ramp(0,2e-3,6000), "t")
DeqY("x'=v", 0, 1e-5, 1e-7,"m", 0)
DeqY("phi'=omega", 0.5, 1e-5, 1e-7,"", 0)
DeqY("v'=mLd*Lng*omega^2*sin(phi)-v*c", 0, 1e-5, 1e-7,"m/s", 0)
DeqY("omega'=-mLd*g*Lng*sin(phi)", 0, 1e-5, 1e-7,"1/s", 0)
while DeqCalc() > 0
end
X = DeqResult("x")
phi= DeqResult("phi")
v = DeqResult("v")
omega = DeqResult("omega")
```

DeqResult

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Abfrage eines Ergebnisses

Deklaration:

```
DeqResult ( Name_Y ) -> Ergebnis
```

Parameter:

Name_Y	Bezeichnung der gewünschten Variable. Z.B. "y"
Ergebnis	Der berechnete Verlauf der gewünschten Variable

Beschreibung:

Alle Größen, die mit DeqY() eingeführt wurden, können abgeholt werden. Wurde etwa "y'=..." definiert, so kann das Ergebnis y abgefragt werden.

Zusätzliche Ableitungen können abgefragt werden, wenn deren Berechnung vorher mit DeqY() und dem Parameter "Ableitungen" beauftragt wurde: Wurde etwa "y'=..." definiert, so kann mit "Ableitungen"=1 auch y' abgefragt werden. Mit "Ableitungen"=2 auch y'' abgefragt werden.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Nach der eigentlichen Berechnung mit DeqCalc() erfolgt die Abfrage der Ergebnisse mit DeqResult().

Beispiele:

Abklingfunktion

```
DeqInit(1, 0, 0)
DeqY("y'=-0.1*y", 1, 1e-7, 1e-8, "", 0)
DeqX(ramp(0,0.3, 300))
while DeqCalc() > 0
end
Y = DeqResult("y")
```

Abklingfunktion, Ableitungen gewünscht

```
DeqInit(1, 0, 0)
DeqY("y'=-0.1*y", 1, 1e-7, 1e-8, "", 2)
DeqX(ramp(0,0.3, 300))
while DeqCalc() > 0
end
Y = DeqResult("y")
{Y'} = DeqResult("y'")
{Y''} = DeqResult("y''")
```

ungedämpftes Pendel

```
DeqInit(2, 0, 0)
DeqX(ramp(0,0.01,1000))
DeqY("phi'", 3 )
DeqY("phi''=-(9.81/1.5)*sin(phi)", 0 )
while DeqCalc() > 0
end
phi = DeqResult("phi")
{phi'} = DeqResult("phi'")
```

DeqX

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Festlegen der Zeitbasis

Deklaration:

```
DeqX ( Zeitbasis_Signal [, Formelzeichen x] )
```

Parameter:

Zeitbasis_Signal	Signal, das als Zeitbasis dient.
Formelzeichen x	Formelzeichen für x, die unabhängige Koordinate, nach der abgeleitet wird. Kann z.B. auf "t" umdefiniert werden. (optional , Standardwert: "")

Beschreibung:

Hier wird ein Datensatz mit der gewünschten Zeitbasis des Ergebnisses übergeben.

Der Inhalt des angegebenen Signals wird nicht beachtet.

Startzeit, Abtastzeit, x-Einheit und Länge legen die Stützstellen der Ergebnisdaten fest.

Alle Eingangssignale (sowie auch der Datensatz für X) müssen dieselbe Zeitbasis haben, also gleiche Startzeit, gleiche Abtastzeit, gleiche Länge.

Wenn keine Eingangsdaten U vorliegen, DeqInput() also nicht aufgerufen wird, muss DeqX() unbedingt aufgerufen werden.

Die Abtastzeit muss angemessen für das zu berechnende Problem vorgegeben werden. Ist sie zu groß, braucht der Algorithmus zu viel Zeit mit Schrittweitenverfeinerung.

Der angegebene Datensatz ist äquidistant. Er kann keine Events und keine Segmente haben.

Ein Aufruf mit einer Gruppe als Parameter ist nicht möglich.

Wenn DeqX() nicht aufgerufen wird, wird "x" als Formelzeichen angenommen.

Auch wenn DeqInput() aufgerufen, darf DeqX() aufgerufen werden. Damit kann das Formelzeichen dann umdefiniert werden. Bequemerweise kann als Datensatz einer der an DeqInput() übergebenen Datensätze benutzt werden.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Vor der eigentlichen Berechnung mit DeqCalc() muss mindestens ein Aufruf von DeqInput() oder DeqX() erfolgen.

Beispiele:

Integration der Parabel, deren Ableitung gegeben ist: y'= 3*t^2

```
tt = ramp(0,1e-3, 10000)
DeqInit(1, 0, 0)
DeqX(tt, "t")
DeqY("y'=0.03*t^2", 0)
```

DeqY

Verfügbar ab: Professional Edition

Lösung eines Systems von gewöhnlichen Differentialgleichungen: Festlegen einer zu integrierenden Variable

Deklaration:

```
DeqY ( Formel [, Anfangswert] [, Toleranz relativ] [, Toleranz absolut] [, y-Einheit] [, Ableitungen] [, Vorgeschichte] )
```

Parameter:

Formel	Frei definierbare Formel. Z.B. " y'=-0.1*y ". Damit wird eine Gleichung des Gleichungssystem definiert.
Anfangswert	Anfangswert (für y, wenn die Formel mit y'= beginnt) (optional , Standardwert: 0)
Toleranz relativ	Toleranz relativ (für y, wenn die Formel mit y'= beginnt) (optional , Standardwert: 0)
Toleranz absolut	Toleranz absolut (für y, wenn die Formel mit y'= beginnt) (optional , Standardwert: 0)
y-Einheit	y-Einheit für das berechnete y (optional , Standardwert: "")
Ableitungen	Sollen zusätzlich zu y auch Ableitungen wie y' und y'' als Rechenergebnis angeboten werden? (optional , Standardwert: 0)
	0 : nein
	1 : 1. Ableitung
	2 : 1. und 2. Ableitung
Vorgeschichte	Vorgeschichte. Wert vor dem Anfang. Nur bei retardierten Differentialgleichungen, wenn dieses Y auch verzögert mittels delay() in einer Formel benutzt wird. (optional , Standardwert: 0)

Beschreibung:

Toleranzen

Die Toleranzen werden pro Schritt angewendet. Aber nicht errechneter Wert gegen wahrer Wert, weil letzter nicht bekannt ist. Sondern Vergleich von zwei errechneten Werten.

Bei Schrittweitensteuerung und Verfeinerung der Schrittweite werden pro verfeinertem Schritt engere Toleranzen benutzt. Die sich ergebene Abweichung für den gesamten Schritt kann aber größer werden.

Sind die Toleranzen zu groß gewählt, besteht eine besondere Gefahr, dass Instabilität entsteht. Denn grob falsche Werte werden als akzeptierte Werte eingestuft.

Sind absolute und relative Toleranz beide = 0 sind, wird relative Toleranz = 1e-6 und absolute Toleranz = 1e-20 angenommen.

Vorsicht, wenn einer der Werte für absolute bzw. relative Toleranz = 0 wird: Wenn absolute Toleranz = 0, dann bei winzigem Betrag stark verfeinerte Schritte. Wenn relative Toleranz = 0, dann bei riesigem Betrag stark verfeinerte Schritte. Also sollten möglichst beide sinnvoll festgelegt werden.

Zu integrierenden Variablen

Bei der Formel wird i.a. die Ableitung einer Größe links vom Gleichheitszeichen angegeben. Rechts davon der Term, wie diese Ableitung errechnet wird.

Liegt eine Differentialgleichung vor, bei der eine höhere Ableitung definiert ist, etwa "y1"=...", so wird i.a. eine Zwischengröße eingeführt: y1'=y2; Das führt dann auf die Formel "y2'=..." und eine weitere Formel "y1'=y2". DeqY() wird dann zweimal aufgerufen.

Alternativ kann auf die Zwischengröße verzichtet werden: "y1"=..." wird als eine Formel angegeben. "y1'" ohne Gleichheitszeichen wird als zweite Formel in einem weiteren Aufruf von DeqY() angegeben. Denn für y1' müssen auch Startwerte, Toleranzen etc. vorgegeben werden.

Die Lösung von Differentialgleichungen beginnt stets mit einem Aufruf von DeqInit(). Vor der eigentlichen Berechnung mit DeqCalc() muss mindestens ein Aufruf von DeqY() erfolgen.

Stetigkeit

Die Formel sollte auf eine stetige Funktion führen. Das Runge-Kutta nimmt das an. Bei Sprüngen zwischen den Stützstellen reagiert das Runge-Kutta i.a. mit (starker) Verfeinerung der Schrittweite.

Bei implizitem Runge-Kutta wird die Jacobi-Matrix benötigt. Dabei sollten dann auch sämtliche Ableitungen der Formel stetig sein.

Wenn mittels DeqInput() Eingangsgrößen vorgegeben werden, werden diese als Treppen gedeutet. Da die entstehenden Unstetigkeiten aber nur an den Stützstellen auftreten, stören sie das Runge-Kutta Verfahren nicht.

Formel Inhalt

Die Formel kann neben den Rechenzeichen auch die Funktionen sin, cos, tan, exp, asin, acos, atan, ln, log, sqrt benutzen.

Alle trigonometrischen Funktionen arbeiten im Bogenmaß. Falls eine Phase im Gradmaß gewünscht ist, können die vordefinierten Konstanten PI und PI2 benutzt werden.

Besondere Funktionen

abs	Beim Absolutbetrag abs(x) ist die unstetige Ableitung bei x=0 zu beachten.
sign	Vorzeichenfunktion. sign(x) Liefert 1, 0 -1 bei x>0, x=0, x<0. Die Funktion ist bei x=0 unstetig.
condi	Entscheidungsfunktion. condi(y,a,b) Liefert a, falls y ungleich 0; sonst b. Beispiel: condi(x>0.1, x, 0). Stetigkeit abhängig von den Parametern.
delay	Verzögerungsfunktion. delay(y,delaytime); y ist dabei eine der zu integrierenden Variablen. Diese Variable wird um die fest vorgegebene Zeit delaytime verzögert. Wenn diese Funktion benutzt wird, führt das auf eine retardierte Differentialgleichung.

Retardierte Differentialgleichungen

Die verzögerten Variablen, also y(x-x0) statt y(x) werden mittels Funktion delay() innerhalb einer Formel notiert.

Die verzögerten Verläufe werden zwischen den ermittelten Stützstellen kubisch interpoliert.

Der unverzögerte Funktionsverlauf startet mir dem als Parameter "Anfangswert" übergebenen Wert. Da bei verzögerten Verläufen auch Werte vor dem Start benötigt werden, wird dabei auf den Parameter "Vorgeschichte" zurückgegriffen. Dieser definiert den Wert vor dem Startzeitpunkt.

Wenn die Verzögerungszeit sehr groß ist, muss ggf. temporär eine entsprechend große Datenmenge gehalten werden.

Wenn die Verzögerungszeit sehr klein ist, muss ggf. mit verkleinerter Schrittweite gearbeitet werden.

Beispiele:

Abklingfunktion mit Startwert y=3.0

```
DeqInit(1, 0, 0)
DeqY("y'=-0.1*y", 3.0)
```

SDOF. Die Gleichung "y''=u-12*y'-400000*y" ist gegeben. y ist der Weg. y'' ist die Beschleunigung, y'=v für die Geschwindigkeit wird eingeführt.

```
DeqInit(2, 1, 0)
DeqY("y'=v", 0.0)
DeqY("v'=u-12*v-400000*y", 1)
```

SDOF. Die Gleichung "y''=u-12*y'-400000*y" ist gegeben. Auf das zusätzliche Symbol y'=v kann verzichtet werden. Dennoch muss für y' und v'(=y'') ein Startwert vorgegeben werden. Die definierende Gleichung gibt es nur einmal, nämlich bei der höchsten Ableitung.

```
DeqInit(2, 1, 0)
DeqY("y'", 0.0)
DeqY("y''=u-12*y'-400000*y", 1)
```

Abklingfunktion, Ableitungen gewünscht

```
DeqInit(1, 0, 0)
DeqY("y'=-0.1*y", 1, 1e-7, 1e-8, "", 2)
DeqX(ramp(0,0.3, 300))
while DeqCalc() > 0
end
Y = DeqResult("y")
{Y'} = DeqResult("y'")
{Y''} = DeqResult("y''")
```

Retardierte Differentialgleichung, Mackey-Glass. P bei Start und vorher auf 0.1

```
P_ini=0.1
DeqInit(1, 0, 0)
N=10
Theta= 1
DeqConst(Theta^N, "theta")
DeqConst(22, "tau")
DeqConst(0.2, "beta")
DeqConst(0.1, "gamma")
DeqConst(N, "N")
DeqX(ramp(0,0.2,4000))
DeqY("P'=(beta*theta*delay(P,tau))/(theta+delay(P,tau)^N)-gamma*P", P_ini, 1e-6, 1e-6, "", 0, P_ini )
while DeqCalc() > 0
end
P = DeqResult("P")
```

DFilt

Digitales Filter

Deklaration:

```
DFilt ( Daten, Koeffizienten ) -> Filtrat
```

Parameter:

Daten	Zu filternder Datensatz. Erlaubte Datentypen: [ND]
Koeffizienten	Datensatz, der die Nenner- und Zählerkoeffizienten enthält
Filtrat	Ergebnis der Filterung.

Beschreibung:

Ein Datensatz wird digital gefiltert. Das Filter wird über einen Koeffizienten-Datensatz spezifiziert. Es können nur kausale Filter berechnet werden.

Die Koeffizienten können in 2 verschiedenen Formen vorliegen.

1. In ausmultiplizierter Form

Die Koeffizienten müssen folgender Form genügen:

$$\frac{Y}{U} = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{a_0 + a_1 z^{-1} + \ldots + a_n z^{-n}} \quad ; \quad a_0 \neq 0$$

oder im Zeitbereich ausgedrückt:

$$y[t] = \frac{b_0}{a_0} u[t] + \frac{b_1}{a_0} u[t-1] + \frac{b_2}{a_0} u[t-2] + \ldots - \frac{a_1}{a_0} y[t-1] - \frac{a_2}{a_0} y[t-2] \ldots$$

Der Koeffizienten-Datensatz muss erst alle an und dann alle bn enthalten. Der Koeffizienten-Datensatz muss immer eine gleiche Anzahl von Nenner- und Zähler-Koeffizienten enthalten. Nicht benötigte Koeffizienten werden mit Null angegeben. Die Nenner-Koeffizienten sind als erste anzugeben. Der erste Wert der Nennerkoeffizienten a0 muss ungleich Null sein.

2. In Biquad-Darstellung

Das Filter wurde als "Reihenschaltung" von Filtern 2.Ordnung entworfen.

Jedes dieser Filter ist durch einen Biquad-Term der Form

$$\frac{Y}{U} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}$$

bzw. im Zeitbereich wie folgt definiert.

```
y[t] = b0u[t] + b1u[t-1] + b2u[t-2]+ a1y[t-1] + a2y[t-2]
```

Der Koeffizientendatensatz muss dann in folgender Form angegeben werden:

```
0 | a2 a1 b2 b1 b0 | a2 a1 b2 b1 b0 |...| a2 a1 b2 b1 b0 |
0 |_1. biquad_____|_2. biquad_____|...|_n. biquad_____|
```

Die führende 0 ist die feste Kennung, um automatisch das Format identifizieren zu können.

Dieses Format der Koeffizientenübergabe wird vom imc-Filterentwurfsprogramm benutzt.

Beispiele:

```
file = GetSystemInfo("Famos.Path.SampleData", "") + "\Damped_Harmonic_Oszillator.dat"
FileLoad(file, "", 0)
ParamT = [1, -0.9, 2, 0.1]
Filtrat = DFilt(Damped_Harmonic_Oszillator, ParamT)
```

Die Beispiel-Datei 'Damped_Harmonic_Oszillator.dat' wird geladen. Eine Variable 'ParamT' wird mit Filterkoeffizienten eines Tiefpasses erster Ordnung gefüllt. Der Datensatz wird mit dem so definierten Tiefpass gefiltert und auf die Variable 'Filtrat' gelegt.

Siehe auch:

Smo, Smo3, FilterAnalog, FiltHP, FiltLP, FiltBP, FiltBS

DFTSpectrum

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Die DFT (diskrete Fouriertransformation) wird auf das Zeitsignal angewendet. Dabei wird das Effektivwert-Spektrum des Zeitsignals bestimmt. Die Länge des Zeitsignals muss keine 2er-Potenz sein.

Deklaration:

```
DFTSpectrum ( Zeitsignal, Fenstertyp ) -> Ergebnis
```

Parameter:

Zeitsignal	Zeitlicher Verlauf des Signals, von dem das Spektrum berechnet werden soll.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Ergebnis	Das ermittelte Spektrum ist ein komplexer Datensatz mit Betrag und Phase. Der Betrag der einzelnen Frequenzlinien ist als Effektivwert angegeben.

Beschreibung:

Beispiele:

Von einem Zeitsignal t soll die DFT mit Rechteckfenster bestimmt werden.

```
Spektrum = DFTSpectrum ( t, 0 )
```

Siehe auch:

FFT, ZoomSpectrumChirpZ

Dialog

Aufruf eines benutzerdefinierten Dialoges oder eines Panels im Dialog-Modus.

Deklaration:

```
Dialog ( TxDateiName, TxParameter, EwOption ) -> EwRück
```

Parameter:

TxDateiName	Dateiname der Dialog-Definitionsdatei (*dlg) oder der Panel-Datei (*.panel).
TxParameter	Der Parameter wird an die Ereignis-Sequenz "Initialisierung" durchgereicht.
EwOption	Optionsparameter. Reserviert, auf 0 zu setzen.
EwRück	Rückgabewert des Dialogs/Panels. Entspricht dem Wert des an die Funktion DlgCloseDialog() bzw. PnClose() übergebenen Parameters.

Beschreibung:

Die Funktion startet einen anwenderdefinierten Dialog oder ein Panel im Dialog-Modus.

Die Funktion kehrt erst zurück, nachdem der Dialog bzw. das Panelfenster vom Anwender wieder geschlossen wurde. In der Zwischenzeit ist das FAMOS-Hauptfenster nicht bedienbar.

Der Rückgabewert der Funktion ist gleich dem an die Funktion DlgCloseDialog() bzw. PnClose() übergebenen Parameter.

Für Panele, die mit diesem Befehl gestartet werden, sollte die Option "Panel Einstellungen"/"Verwendung als Dialog-Fenster" aktiviert sein.

Der 2. Parameter wird unverändert an die Ereignis-Sequenz "Initialisierung" durchgereicht.

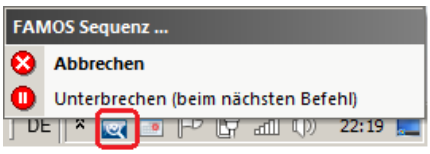
Wenn bei dem Dateinamen kein kompletter Pfad angegeben ist, wird die Datei nacheinander in den folgenden Verzeichnissen gesucht:

- Projektverzeichnis: Wenn ein Projekt geladen ist, wird im aktuellen Projektverzeichnis gesucht.
- Aktuelles Arbeitsverzeichnis: Dieses entspricht dem Verzeichnis, aus dem die aufrufende Sequenz geladen wurde.
- Standardverzeichnis für Sequenzen, Dialoge und Panele: Dieses steht beim Aufstart von imc FAMOS zunächst auf dem unter "Optionen"/"Verzeichnisse" vereinbarten Verzeichnis. Es kann mit dem Befehl MDIR oder der Funktion SetOption() umgesetzt werden.

Wenn keine Dateierweiterung angegeben ist, wird der Dateiname zuerst um '.dlg' ergänzt. Falls keine solche Datei vorhanden ist, wird die Erweiterung '.panel' probiert.

Wenn eine Sequenz aktiv ist, finden Sie im Info-Bereich der Windows-Taskleiste ein zusätzliches Symbol. Mit einem Rechtsklick erhalten Sie ein Kontextmenü, mit dem Sie die Ausführung abbrechen können.

Zum Unterbrechen können Sie auch die Tastenkombination "Strg" + "UNTBR" verwenden.



Zur Erstellung und Verwendung von Dialogen und Panelen informieren Sie sich bitte in den Kapiteln 'Anwenderdefinierte Dialoge' und 'Panel' im imc FAMOS-Bedienerhandbuch.

Multithreading: Die Funktion darf nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

In einem Dialog 'EditProperties.dlg' werden diverse Kennwerte eines Datensatzes angezeigt, unter anderem auch die Triggerzeit in einem 'Kalender'- und einem 'Uhrzeit'-Element. Der Name des Datensatz wird als Parameter an die Funktion Dialog() übergeben. Beim Beenden des Dialogs mit dem Knopf 'OK' wird eine eventuelle Änderung der Triggerzeit durch den Anwender in den Datensatz übernommen und der Dialog geschlossen.

Aufruf-Beispiel:

```
OK = Dialog("EditProperties", "Chan1", 0)
```

Ereignis-Sequenz 'Dialog Initialisierung'

```
TxVarName = PA1
time = Time?(<TxVarName>)
DlgSetValue("time", time)
DlgSetValue("date", time)
```

Ereignis-Sequenz 'Gedrückt' des 'OK'-Knopfes

```
date = DlgGetValue("time")
```

```
time = DlgGetValue("date")
SetTime(<TxVarName>, TimeAdd(date, time))
DlgCloseDialog(1)
```

Ereignis-Sequenz 'Gedrückt' des 'Abbrechen'-Knopfes

```
DlgCloseDialog(0)
```

Wenn der Dialog mit der [OK]-Schaltfläche beendet wurde, erhält die Variable OK den Wert 1. Wenn dagegen die [Abbrechen]-Schaltfläche verwendet wurde, wird eine 0 zurückgegeben.

Ein Panel 'InputValue.panel' besteht u.a. aus einem Eingabefeld "input" zur Eingabe eines positiven Zahlenwertes sowie 2 Schaltflächen 'OK' und 'Abbrechen'. Der Dialog()-Befehl liefert den eingegebenen Wert zurück bzw. -1, falls abgebrochen werden soll.

Ereignis-Sequenz 'Knopf gedrückt' für den 'OK'-Knopf:

```
value = PnGetValue("input")
PnClose(value)
```

Ereignis-Sequenz 'Knopf gedrückt' für den 'Abbrechen'-Knopf

```
PnClose(-1)
```

Ereignis-Sequenz 'Schließen' (Anwender verwendet Systemmenü zum Schließen):

```
; Selbes Verhalten wie 'Abbrechen'-Knopf
PnClose(-1)
```

Aufruf des Dialogs:

```
value = Dialog("InputValue.panel", "", 0)
IF value < 0
   EXITSEQUENCE 0
END
;Weiter in der Sequenz...
...
```

Siehe auch:

SEQUENCE, BoxMessage, BoxValue?, BoxText?

Diff

Differentiation, Ableitung

Deklaration:

```
Diff ( Daten ) -> Ergebnis
```

Parameter:

Daten	Zu differenzierender Datensatz. Erlaubte Datentypen: [ND],[XY]
Ergebnis	Ergebnis der Differentiation.

Beschreibung:

Der übergebene Datensatz wird differenziert (abgeleitet). Es wird nach einem einfachen, effektiven Algorithmus abgeleitet:

Zwischen benachbarten Abtastwerten wird die Differenzen der y-Werte gebildet und durch den x-Abstand beider Punkte dividiert. Bei normal reellen Datensätzen ist dieser Abstand gleich der Abtastzeit (bzw. x- Delta).

Damit wird die Differentiation umkehrbar zur Integration.

Die Ableitung eines Datensatzes liefert die Steigung (Steilheit) in jedem Punkt. Beachten Sie, dass Differenzieren den Datensatz "aufraut".

- Da beim Differenzieren aus je 2 benachbarten Werten eine Differenz gebildet wird, wird die Länge des erzeugten Datensatzes um 1 geringer sein. Das Differenzieren eines leeren Datensatzes oder eines Einzelwertes führt auf einen leeren Datensatz.
- Besonderheit bei XY-Daten: Der letzte berechnete Wert wird nochmal angehängt. Damit wird eine sinnvolle Darstellung des Ergebnisses im Kurvenfenster (Treppendarstellung) sowie die Umkehrbarkeit mit der Int()-Funktion gewährleistet. Die Länge von Eingangsdaten und Ergebnis ist hier also gleich.
- Die Einheit des differenzierten Datensatzes ist der Quotient aus der y- und x-Einheit des übergebenen Datensatzes.

Beispiele:

```
NDdiff = Diff(NDdata)
```

Einfache Berechnung der Steigung.

```
NDdiff = Diff(Smo3(NDdata))
```

Oft ist es angebracht, den Datensatz vor dem Differenzieren zu glätten, um den Einfluss von Rauschen zu vermindern. Rauschen verfälscht nämlich das Ergebnis sehr stark.

```
NDdiff = Red(Diff(IPol(NDdata, 3)), 3)
```

Bei wenig verrauschten Datensätzen, die jedoch relativ große Sprünge in den Funktionswerten aufweisen, ist eine Spline-Interpolation vor dem Ableiten angebracht. Nach dem Differenzieren kann die Datenmenge auf die angemessene Informationsmenge mit der Abtastfunktion Red() reduziert werden.

```
NDdata = Diff(Int(NDdata))
```

Diese Formel verändert den Datensatz nicht.

```
NDyoff = Int(Diff(NDdata))
```

Der berechnete Datensatz kann sich um einen Offset in y- Richtung unterscheiden.

Siehe auch:

Int, MInt

DisplayY?

Abfrage der festen Skalierung für die Anzeige im Kurvenfenster
Alternativer Name: **AnzeigeY?**

Deklaration:

```
DisplayY? ( Daten, EwAuswahl ) -> EwSkalWert
```

Parameter:

Daten	Datensatz, dessen Y-Skalierung ermittelt werden soll
EwAuswahl	Auswahl des abzufragenden Kennwertes
	0 : Unterer Skalenwert
	1 : Oberer Skalenwert
EwSkalWert	Unterer bzw. oberer Skalenwert

Beschreibung:

Einem Datensatz kann als zusätzliche Eigenschaft eine feste Skalierung der y-Achse zugewiesen werden. Diese wird dann bei der Anzeige des Datensatzes in einem Kurvenfenster benutzt, wenn die Einstellung "Automatische Skalierung der Y-Achse" am Kurvenfenster gesetzt ist.

Falls dem Datensatz eine feste Skalierung der y-Achse zugeordnet ist, liefert diese Funktion die Bereichsgrenzen.

Falls keine feste Skalierung zugeordnet ist, wird für beide Grenzen eine 0 zurückgegeben. Der Datensatz wird dann entsprechend seinem Wertebereich skaliert.

Beispiele:

```
yMin = DisplayY?(data, 0)
IF yMin > 0
   SetDisplayY(data, 0, DisplayY?(data, 1))
ELSE
   yMax = DisplayY?(data, 1)
   IF yMax < 0
      SetDisplayY(data, DisplayY?(data, 0),0)
   END
END
```

Falls dem Datensatz eine feste Skalierung zugewiesen ist und die Null-Linie nicht enthalten ist, wird der untere oder obere Skalenwert auf 0 korrigiert.

Siehe auch:

SetDisplayY, Color?

Div

Zeit- bzw. x-richtige Division

Deklaration:

```
Div ( Dividend, Divisor, EwOption ) -> Quotient
```

Parameter:

Dividend	Erster Parameter, Dividend. Erlaubte Typen: [ND],[XY].
Divisor	Zweiter Parameter, Divisor. Erlaubte Typen: [ND],[XY].
EwOption	Option
	0 : Die Triggerzeit der beiden Summanden wird ignoriert
	1 : Zeitrichtige Überlagerung mit Berücksichtigung der Triggerzeit
Quotient	Quotient, Ergebnis der Division. [XY]

Beschreibung:

Zwei Datensätze werden zeit- bzw. x-richtig dividiert, d. h. es werden immer 2 y-Werte dividiert, die die gleiche Zeit bzw. den gleichen x-Wert besitzen.

Das Ergebnis ist nur für den x-Bereich definiert, den die beiden Parameter-Datensätze gemeinsam haben. In diesem Bereich wird überall dort ein Ergebnis-Wert berechnet, wo mindestens einer der beiden Datensätze eine Stützstelle besitzt. Wenn an dieser Stelle der andere Datensatz keine Stützstelle besitzt, wird dieser dort linear interpoliert.

Die x-Spur beider Parameter-Datensätze muss monoton steigend sein, d.h. die x-Koordinaten müssen kontinuierlich wachsen.

Der Operator /(Division) führt dagegen eine punktweise Division von Datensätzen aus.

Beispiele:

Zwei Kanäle werden zwischen 11 und 13 Uhr bzw. 12 und 14 Uhr aufgenommen.

```
Quot12_13h = Div(voltage11_13h, voltage12_14h, 1)
```

Eine zeitrichtige Division der beiden Datensätze mit Berücksichtigung der Triggerzeit wird ausgeführt. Das Ergebnis ist zwischen 12 und 13 Uhr definiert.

Siehe auch:

/(Division), Add, Sub, Mult, Append

DlgApplyData

Anwendungsbereich: Anwenderdefinierte Dialoge

Änderungen in allen Eingabefeldern auf Gültigkeit prüfen und/oder in die verknüpften Variablen übernehmen.

Deklaration:

```
DlgApplyData ( ) -> Erfolg
```

Parameter:

Erfolg	0 bei Fehler. 1 bei Erfolg.

Beschreibung:

Für alle Dialog-Elemente mit aktiver Daten-Anbindung wird der aktuelle Wert entsprechend der in den Element-Eigenschaften festgelegten Kriterien (siehe 'Eingabeprüfung') überprüft und bei Erfolg der Wert in eine eventuell verknüpfte Variable übernommen.

Falls die Prüfung fehlschlägt, wird der Anwender mit einer Mitteilungsbox informiert. Eine typische Anwendung für diese Funktion wäre z.B. die 'Gedrückt'-Sequenz eines 'Übernehmen"-Knopfes oder die 'Dialog Schließen'-Sequenz.

Beispiele:

Beim Betätigen des Knopfes 'OK' werden alle Eingaben geprüft und eventuell verknüpfte Variablen aktualisiert. Bei Erfolg wird der Dialog beendet.

--> Ereignis-Sequenz 'Gedrückt' des 'OK'-Knopfes

```
IF DlgApplyData()
   DlgCloseDialog(0)
END
```

Siehe auch:

DlgCloseDialog

DlgCloseDialog

Anwendungsbereich: Anwenderdefinierte Dialoge

Der Dialog wird geschlossen.

Deklaration:

```
DlgCloseDialog ( EwRückgabe )
```

Parameter:

EwRückgabe	Rückgabewert des Dialoges. Entspricht dem Rückgabewert der Funktion Dialog().

Beschreibung:

Der Dialog wird geschlossen. Dies geschieht in 3 Schritten:

- Das Dialogfenster wird unsichtbar gemacht.
- Das Ereignis 'Dialog Ende' wird ausgelöst.
- Nach Abarbeitung der Ereignissequenz wird der Dialog geschlossen. Die Funktion Dialog(..), mit der der Dialog ursprünglich erzeugt wurde, kehrt zurück und liefert den hier als Parameter übergebenen Wert als Ergebnis..

Die Funktion wird standardmäßig in der Ereignissequenz 'Dialog Schließen' aufgerufen. Dieses Ereignis wird ausgelöst, wenn der Anwender den Knopf 'Schließen' in der Titelzeile des Fensters, den entsprechenden Menüpunkt im Systemmenü des Fensters oder die Tastenkombination ALT+F4 aufgerufen hat.

Ansonsten wird die Funktion oft in 'Knopf gedrückt' - Ereignissequenzen verwendet, um auf Knopfdruck den Dialog zu beenden, typisch sind z.B. Knöpfe mit den Beschriftungen 'OK' und 'Abbrechen'.

Empfehlenswert ist dann, für jeden Knopf einen eindeutigen Parameterwert zu übergeben. Da dieser an den Rückgabewert der Funktion Dialog(..) durchgereicht wird, kann der Aufrufer feststellen, mit welchem Knopf der Dialog geschlossen wurde.

Beispiele:

Ein Dialog 'Fortfahren.dlg' besteht lediglich aus der Frage 'Möchten Sie die Sequenzbearbeitung fortsetzen?' sowie 2 Schaltflächen 'OK' und 'Abbrechen'.

--> Ereignis-Sequenz 'Knopf gedrückt' für den 'OK'-Knopf:

```
DlgCloseDialog(1)
```

--> Ereignis-Sequenz 'Knopf gedrückt' für den 'Abbrechen'-Knopf:

```
DlgCloseDialog(0)
```

--> Aufruf des Dialoges:

```
Weiter = Dialog("Fortfahren.dlg", "", 0)
IF Weiter = 0
   EXITSEQUENCE 0
END
;Weiter in der Sequenz...
...
```

Siehe auch:

Dialog

DlgDeleteItem

Anwendungsbereich: Anwenderdefinierte Dialoge

Löscht einen bzw. alle Einträge (Liste, Tabelle etc.)

Deklaration:

```
DlgDeleteItem ( TxElementName, Index )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes.
Index	Index des zu löschenden Eintrages. Der erste Eintrag hat den Index 1. Um alle Einträge zu löschen, geben Sie eine 0 an.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, Tabelle, Baumansicht

Beispiele:

In einem Listenfeld mit Mehrfachselektion werden alle selektierten Einträge gelöscht:

```
Count = DlgGetItemCount("list1")
i = Count
WHILE i > 0
   IF DlgIsItemSelected("list1", i)
      DlgDeleteItem("list1", i)
   END
   i = i - 1
END
```

Siehe auch:

DlgGetItemCount, DlgGetItemText, DlgSetItemText, DlgInsertItem, DlgFindItem

DlgEnable

Anwendungsbereich: Anwenderdefinierte Dialoge

Der aktive Dialog bzw. das angegebene Dialogelement wird gesperrt (somit durch den Anwender nicht bedienbar) bzw. wieder freigegeben.

Deklaration:

```
DlgEnable ( TxElementName, AnAus )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes. Wenn ein leerer Text angegeben wird, wird der gesamte aktive Dialog gesperrt/freigegeben.
AnAus	
	0 : Bedienung sperren
	1 : Bedienung zulassen

Anwendbar auf:

Alle Dialogelemente (auch Menüelemente)

Beispiele:

Ein Dialog zur Abfrage eines Datums besteht aus einem Dialogelement vom Typ 'Kalender' und einem Knopf 'btnOk' zum Beenden des Dialoges. Der Knopf 'btnOK' ist erst bedienbar, nachdem der Bediener ein gültiges Datum eingegeben hat.

--> Ereignissequenz 'Dialog-Initialisierung'

```
;'OK'-Knopf erstmal sperren
DlgEnable("btnOK", 0)
```

--> Ereignissequenz 'Element geändert' des 'Kalender'-Dialogelements

```
;Test des eingestellten Datums
_Date = DlgGetValue( PA1 )
IF _Date >= TimeSystem?()
   DlgEnable( "btnOK", 1)
ELSE
   DlgEnable( "btnOK", 0)
END
```

--> Ereignissequenz 'Knopf gedrückt' des Knopfes 'btnOK'

```
;Dialog beenden und Datum zurückgeben
DlgCloseDialog(_Date)
DELETE _Date
```

Siehe auch:

DlgShow

DlgExpandTree

Anwendungsbereich: Anwenderdefinierte Dialoge

Auf- oder Zuklappen eines Knotens in einer Baumansicht.

Deklaration:

```
DlgExpandTree ( TxElementName, Index, Aktion )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes.
Index	Position gewünschten Baumeintrages. Der erste Eintrag hat den Index 1. Geben Sie eine 0 an, um alle Einträge auf- bzw. zuzuklappen
Aktion	Legt die durchzuführende Aktion fest.
	0 : Zuklappen des Knotens
	1 : Aufklappen des Knotens
	2 : Gesamten Unterbaum aufklappen

Anwendbar auf:

Baumansicht

Beispiele:

Der aktuell selektierte Knoten in einer Baumansicht wird aufgeklappt.

```
i = DlgGetSelectedItem("Tree")
IF i > 0
   DlgExpandTree("Tree", i, 1)
END
```

Siehe auch:

DlgInsertTreeItem, DlgGetItemLevel

DlgFileName

Ein Dialog zur Auswahl/Eingabe eines Dateinamens wird aufgerufen.
Alternativer Name: **DlgDateiName**

Deklaration:

```
DlgFileName ( TxVerzeichnis, TxErweiterung, TxTitel, EwAufgabe ) -> TxDateiname
```

Parameter:

TxVerzeichnis	Verzeichnis, mit dem der Dialog starten soll. Wenn ein leerer Text übergeben wird, wird das momantan eingestellte Standardverzeichnis zum Laden von Dateien benutzt.
TxErweiterung	Standard-Erweiterung für Dateinamen, z.B. "dat". Wenn eine gültige Erweiterung übergeben wird, werden beim Initialisieren des Dialoges nur Dateien mit dieser Endung angezeigt. Wenn der eingegebene Dateiname ohne Erweiterung ist, wird diese angehängt. Ein leerer Text ist erlaubt. Ab Version 2022 können auch mehrere Erweiterungen, durch Semikolon getrennt, angegeben werden, also z.B. "dat;raw".
TxTitel	Überschrift des Dialoges. Bei leerem Text wird ein Standardtitel benutzt.
EwAufgabe	Aufgabe
	0 : "Datei öffnen"-Dialog
	1 : "Datei speichern"-Dialog
TxDateiname	Eingegebener Dateiname (kompletter Pfad), bei Misserfolg (Abbruch durch den Nutzer) ein leerer Text.

Beschreibung:

Über den Windows-Standarddialog zur Dateiauswahl kann eine vorhandene Datei selektiert oder ein neuer Dateiname eingegeben werden.

- Die Position, an der die Dialogbox erscheint, kann mit der Funktion SetBoxPos() vorgegeben werden.
- Sie können die Position des Dialoges auch manuell ändern (z.B. durch Verschieben mit der Maus). Die Position wird für die Dauer der aktuellen Sitzung gemerkt.
- Multithreading: Die Funktion darf nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

```
fileName = DlgFileName("", "dat", "", 0)
FileLoad(fileName, "", 0)
IF VarExist?("y")
   y2 = sin(y)
   saveFileName = DlgFileName("c:\imc\dat", "", "Dateiname zum Sichern angeben", 0)
   FileSave(saveFileName, "", 0, y2)
END
```

Siehe auch:

FsDlgSelectDirectory, FsDlgSelectFiles, BoxValue?, BoxOutput, BoxMessage, SetBoxPos

DlgFindItem

Anwendungsbereich: Anwenderdefinierte Dialoge

Sucht nach einem Eintrag (Liste, Tabelle etc.) mit dem angegebenen Inhalt.

Deklaration:

```
DlgFindItem ( TxElementName, TxInhalt ) -> Index
```

Parameter:

TxElementName	Name des zu untersuchenden Dialogelementes.
TxInhalt	Text für den zu suchenden Eintrag.
Index	Index des gesuchten Eintrages (>=0), falls gefunden. 0 sonst.

Beschreibung:

Die Funktion unterscheidet nicht bezüglich Groß-/Kleinschreibung.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld

Beispiele:

In einem Listenfeld (mit Einfachselektion) wird ein Eintrag gesucht. Falls der Eintrag vorhanden ist, wird dieser selektiert und ggf. in die Sicht gerollt.

```
i = DlgFindItem("list", "100.0")
IF i > 0
   DlgSelectItem("list", i)
END
```

Siehe auch:

DlgGetItemCount, DlgGetItemText, DlgSetItemText, DlgInsertItem, DlgDeleteItem

DlgGetBarMax

Anwendungsbereich: Anwenderdefinierte Dialoge

Ermittelt die obere Grenze des Wertebereichs.

Deklaration:

```
DlgGetBarMax ( TxElementName ) -> Max
```

Parameter:

TxElementName	Name des anzufragenden Dialogelementes (Schieberegler).
Max	Maximum des Wertebereichs.

Anwendbar auf:

Schieberegler

Beispiele:

Ein Schieberegler soll auf einen neuen Wert gesetzt werden. Vorher wird geprügt, ob der gewünschte Wert im zulässigen Bereich liegt und ggf. eine Fehlermeldung angezeigt.

```
IF newValue >= DlgGetBarMin("Slider1") AND newValue <= DlgGetBarMax("Slider1")
   DlgSetValue( "Slider1", newValue)
ELSE
   BoxMessage("Fehler", "Bereichsüberschreitung", "1!")
END
```

Siehe auch:

DlgSetBarRange, DlgGetBarMin, DlgGetValue, DlgSetValue

DlgGetBarMin

Anwendungsbereich: Anwenderdefinierte Dialoge

Ermittelt die untere Grenze des Wertebereichs.

Deklaration:

```
DlgGetBarMin ( TxElementName ) -> Min
```

Parameter:

TxElementName	Name des anzufragenden Dialogelementes (Schieberegler).
Min	Minimum des Wertebereichs.

Anwendbar auf:

Schieberegler

Beispiele:

Ein Schieberegler soll auf einen neuen Wert gesetzt werden. Vorher wird geprügt, ob der gewünschte Wert im zulässigen Bereich liegt und ggf. eine Fehlermeldung angezeigt.

```
IF newValue >= DlgGetBarMin("Slider1") AND newValue <= DlgGetBarMax("Slider1")
   DlgSetValue( "Slider1", newValue)
ELSE
   BoxMessage("Fehler", "Bereichsüberschreitung", "1!")
END
```

Siehe auch:

DlgSetBarRange, DlgGetBarMax, DlgGetValue, DlgSetValue

DlgGetCellText

Anwendungsbereich: Anwenderdefinierte Dialoge

Der Inhalt der angegebenen Tabellenzelle wird abgefragt.

Deklaration:

```
DlgGetCellText ( TxElementName, Zeile, Spalte ) -> TxWert
```

Parameter:

TxElementName	Name des abzufragenden Dialogelementes (Tabelle).
Zeile	Zeilenindex. Die erste Zeile hat den Index 1. Der Spaltenkopf (sofern sichtbar) hat den Index 0.
Spalte	Spaltenindex. Die erste Spalte hat den Index 1. Der Zeilenkopf (sofern sichtbar) hat den Index 0.
TxWert	Aktueller Inhalt der Tabellenzelle

Anwendbar auf:

Tabelle

Beispiele:

In einer Tabelle werden diverse Kennwerte eines Datensatzes angezeigt, unter anderem auch die Y-Einheit. Wenn der Anwender die enstprechende Tabellenzelle ändert, wird die neue Einheit in den Datensatz übernommen.

--> Ereignis-Sequenz 'Dialog Initialisierung'

```
...
DlgSetCellText("Tab1", 1, 2, Unit?(MyData,1))
...
```

--> Ereignis-Sequenz 'Geändert' der Tabelle 'Tab1'

```
Row = PA2
Column = PA3
IF Row = 1 AND Column = 2
   TxUnit = DlgGetCellText(PA1, Row, Column)
   SetUnit(MyData, txUnit, 1)
END
```

Siehe auch:

DlgSetCellText, DlgGetCellValue, DlgSetCellValue

DlgGetCellValue

Anwendungsbereich: Anwenderdefinierte Dialoge

Der Wert des angegebenen Tabellenzelle wird abgefragt.

Deklaration:

```
DlgGetCellValue ( TxElementName, Zeile, Spalte ) -> Wert
```

Parameter:

TxElementName	Name des abzufragenden Dialogelementes.
Zeile	Zeilenindex. Die erste Zeile hat den Index 1. Der Spaltenkopf (sofern sichtbar) hat den Index 0.
Spalte	Spaltenindex. Die erste Spalte hat den Index 1. Der Zeilenkopf (sofern sichtbar) hat den Index 0.
Wert	Aktueller Wert der Tabellenzelle

Beschreibung:

Falls der aktuelle Inhalt der angesprochenen Tabellenzelle nicht in eine Zahl konvertierbar ist, wird eine 0 zurückgegeben.

Anwendbar auf:

Tabelle

Beispiele:

Nach Änderung des Inhalts einer Tabellenzelle durch den Anwender wird geprüft, ob der eingegebene Wert eine Zahl > 0 ist. Falls nicht, wird die Zelle rot markiert.

-->Ereignis-Sequenz 'Geändert' der Tabelle

```
Row = PA2
Column = PA3
newValue = DlgGetCellValue(PA1, Row, Column)
IF newValue <= 0
   DlgSetCellTextColor(PA1, Row, Column, RGB(255,0,0))
ELSE
   DlgSetCellTextColor(PA1, Row, Column, -2)
END
```

Siehe auch:

DlgSetCellValue, DlgGetCellText, DlgSetCellText

DlgGetItemCount

Anwendungsbereich: Anwenderdefinierte Dialoge

Ermittelt die Anzahl der Einträge in dem angegebenen Dialogelement (Liste, Tabelle etc.)

Deklaration:

```
DlgGetItemCount ( TxElementName ) -> Anzahl
```

Parameter:

TxElementName	Name des abzufragenden Dialogelementes.
Anzahl	Anzahl der Einträge.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, Baumansicht, Tabelle, Optionsgruppe

Beispiele:

In einem Listenfeld mit Mehrfachselektion werden alle selektierten Einträge gelöscht:

```
Count = DlgGetItemCount("list1")
i = Count
WHILE i > 0
   IF DlgIsItemSelected("list1", i)
      DlgDeleteItem("list1", i)
   END
   i = i - 1
END
```

Siehe auch:

DlgGetItemText, DlgSetItemText, DlgInsertItem, DlgFindItem, DlgDeleteItem

DlgGetItemLevel

Anwendungsbereich: Anwenderdefinierte Dialoge

Ermittelt die Ebene (den Einzug) eines Eintrages in einer Baumansicht.

Deklaration:

```
DlgGetItemLevel ( TxElementName, Index ) -> Ebene
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes.
Index	Position gewünschten Baumeintrages. Der erste Eintrag hat den Index 1.
Ebene	Ebene des Eintrages. Wurzeleinträge haben die Ebene 0.

Anwendbar auf:

Baumansicht

Beispiele:

Die Bedienbarkeit eines Knopfes sei abhängig von der aktuellen Selektion in einer Baumansicht. Der Knopf soll nur dann bedienbar sein, wenn ein Wurzelelement (Ebene = 0) selektiert ist.

```
i = DlgGetSelectedItem("Tree")
enable = (i > 0) AND (DlgGetItemLevel("Tree", i) = 0)
DlgEnable("button", enable)
```

Siehe auch:

DlgInsertTreeItem, DlgExpandTree

DlgGetItemText

Anwendungsbereich: Anwenderdefinierte Dialoge

Gibt den Text für einen Eintrag in einem Dialogelement (Listenfeld, Baumansicht etc.) zurück.

Deklaration:

```
DlgGetItemText ( TxElementName, Index ) -> TxInhalt
```

Parameter:

TxElementName	Name des abzufragenden Dialogelementes.
Index	Index des abzufragenden Eintrages. Der erste Eintrag hat den Index 1.
TxInhalt	Text für den angegebenen Eintrag.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, Baumansicht

Beispiele:

Aus einer Liste, welche Namen von Messwert-Dateien im FAMOS-Format enthält, wird der aktuell selektierte Eintrag ausgelesen und die entsprechende Datei geladen.

```
i = DlgGetSelectedItem("listFiles")
IF i > 0
   FileName$ = DlgGetItemText("listFiles", i)
   fh = FileOpenDSF( FileName$,0)
   IF fh > 0
      ;...
      FileClose(fh)
   END
END
```

Siehe auch:

DlgGetItemCount, DlgSetItemText, DlgInsertItem, DlgFindItem, DlgDeleteItem

DlgGetPath

Anwendungsbereich: Anwenderdefinierte Dialoge

Der komplette Pfad für die gerade ausgeführte Dialog-Datei wird ermittelt.

Deklaration:

```
DlgGetPath ( Option ) -> TxPfad
```

Parameter:

Option	Option
	0 : Kompletter Dateipfad
	1 : Nur Verzeichnis
TxPfad	Kompletter Pfadname bzw. Verzeichnis der momentan ausgeführten Dialogdatei.

Beschreibung:

Beispiele:

Wenn ein Dialog mit dem Befehl

```
ret = Dialog( "c:\imc\seq\MyDialog.dlg", "", 0)
```

gestartet wurde:

```
TxFullPath = DlgGetPath(0)
   ; TxFullPath enthält "c:\imc\seq\MyDialog.dlg"
TxDir = DlgGetPath(1)
   ; TxDir enthält "c:\imc\seq"
```

Siehe auch:

DlgSetTextColor, DlgSetCellTextColor

DlgGetSelectedItem

Anwendungsbereich: Anwenderdefinierte Dialoge

Ermittelt den aktuell selektierten Eintrag in einer Liste/Tabelle mit Einfachselektion.

Deklaration:

```
DlgGetSelectedItem ( TxElementName ) -> Index
```

Parameter:

TxElementName	Name des zu untersuchenden Dialogelementes.
Index	Index des selektierten Eintrages (>=0). 0, wenn kein Eintrag selektiert ist.

Anwendbar auf:

Listenfeld (mit Einfachselektion), Klappliste, Kombinationsfeld, Tabelle (mit Einfachselektion), Baumansicht, Optionsgruppe

Beispiele:

Aus einer Liste, welche Namen von Messwert-Dateien im FAMOS-Format enthält, wird der aktuell selektierte Eintrag ausgelesen und die entsprechende Datei geladen.

```
i = DlgGetSelectedItem("listFiles")
IF i > 0
   FileName$ = DlgGetItemText("listFiles", i)
   fh = FileOpenDSF( FileName$,0)
   IF fh > 0
      ;...
      FileClose(fh)
   END
END
```

Siehe auch:

DlgSelectItem, DlgIsItemSelected

DlgGetSelectedItemCount

Anwendungsbereich: Anwenderdefinierte Dialoge

Ermittelt die Anzahl der selektierten Einträge in einer Liste mit Mehrfachselektion.

Deklaration:

```
DlgGetSelectedItemCount ( TxElementName ) -> Anzahl
```

Parameter:

TxElementName	Name des zu untersuchenden Dialogelementes.
Anzahl	Anzahl der selektierten Einträge. 0, wenn kein Eintrag selektiert ist.

Beschreibung:

Anwendbar auf:

Listenfeld (Mehrfachselektion), Tabelle (Mehrfachselektion)

Beispiele:

Auf Knopfdruck sollen die selektierten Einträge in einem Listenfeldes mit Multiselektion ausgewertet werden. Der Knopf soll nur bedienbar sein, wenn mindestens 1 Eintrag selektiert ist. Dazu wird im Ereignis 'Ausgewählt' die Zahl der selektierten Einträge geprüft:

--> Ereignis-Sequenz 'Ausgewählt' des Listenfeldes

```
Count = DlgGetSelectedItemCount(PA1)
DlgEnable("Button1", Count > 0)
```

Siehe auch:

DlgIsItemSelected, DlgSetItemSelection, DlgGetSelectedItem

DlgGetText

Anwendungsbereich: Anwenderdefinierte Dialoge

Der Inhalt bzw. die Beschriftung des angegebenen Elements wird abgefragt.

Deklaration:

```
DlgGetText ( TxElementName ) -> TxText
```

Parameter:

TxElementName	Name des abzufragenden Dialogelementes.
TxText	Aktueller Inhalt bzw. Beschriftung des Dialogelements

Anwendbar auf:

Die Funktion kann nur auf solche Dialogelemente angewendet werden, die eine Beschriftung besitzen oder deren aktueller Zustand durch einen Text ausgedrückt werden kann. Die Interpretation des Textes ist abhängig vom Typ des Elementes.

Element	Bedeutung
Knopf	Beschriftung des Knopfes.
Label	Beschriftung des Labels.
Eingabefeld (einzeilig)	Inhalt des Eingabefeldes.
Eingabefeld (mehrzeilig)	Inhalt des Eingabefeldes.
Kontrollkästchen	Beschriftung des Elementes.
Gruppenfeld	Beschriftung des Elementes.
Optionsgruppe	Beschriftung des Rahmens um die Optionsfelder. Nur wirksam für "Rahmen" = "Mit Beschriftung".
Listenfeld (Einfachselektion)	Text des aktuell selektierten Eintrages.
Klappliste	Text des aktuell selektierten Eintrages.
Baumansicht	Text des aktuell selektierten Eintrages.
Kombinationsfeld	Inhalt des Eingabefeldes.
Dialog	Titelzeile des Dialoges. Für TxElementName ist ein leerer Text anzugeben.
Statuszeile	Inhalt der Statuszeile des Dialoges. Für TxElementName ist "#Status" anzugeben.

Für Elemente, die nicht in obenstehender Tabelle gelistet sind, ist die Funktion nicht anwendbar.

Beispiele:

In einem Dialog werden diverse Kennwerte eines Datensatzes angezeigt, unter anderem auch in einem Eingabefeld 'input_unit' die Y-Einheit. Beim Beenden des Dialoges mit dem Knopf 'OK' wird eine eventuelle Änderung der Einheit durch den Anwender in den Datensatz übernommen.

--> Ereignis-Sequenz 'Dialog Initialisierung'

```
...
DlgSetText("input_unit",  Unit?(MyData,1))
...
```

--> Ereignis-Sequenz 'Gedrückt' des 'OK'-Knopfes

```
TxUnit = DlgGetText("input_unit")
SetUnit(MyData, TxUnit, 1)
DlgCloseDialog(0)
```

Siehe auch:

DlgGetValue, DlgGetValue, DlgSetText

DlgGetValue

Anwendungsbereich: Anwenderdefinierte Dialoge

Der aktuelle numerische Wert des angegebenen Elements wird abgefragt.

Deklaration:

```
DlgGetValue ( TxElementName ) -> Wert
```

Parameter:

TxElementName	Name des abzufragenden Dialogelementes.
Wert	Aktueller Wert des Dialogelements

Anwendbar auf:

Die Funktion kann nur auf solche Dialogelemente angewendet werden, deren aktueller Zustand durch eine Zahl ausgedrückt werden kann. Die Interpretation des Wertes ist abhängig vom Typ des Elementes.

Element	Wert
Listenfeld (Einfachselektion)	Falls der aktuell selektierte Wert in der Liste in eine Zahl konvertierbar ist, wird diese zurückgegeben. Ansonsten 0.
Klappliste	..
Kombinationsfeld	Falls der Text im Eingabefeld in eine Zahl konvertierbar ist, wird diese zurückgegeben. Ansonsten 0.
Eingabefeld (einzeilig)	..
Kontrollkästchen	Liefert 1, falls das Kontrollkästchen 'angekreuzt' ist, ansonsten 0.
Optionsgruppe	Liefert den Index der aktuell ausgewählten Option. Die erste Option hat den Index 1.
Schieberegler	Liefert den aktuellen Wert des Schiebereglers.
Kalender	Liefert den aktuellen Wert im FAMOS-Zeitformat. Der Wert kann mit den Funktionen der Funktionsgruppe #18 "Datum/Uhrzeit" weiterverarbeitet werden
Uhrzeit	Liefert den aktuellen Wert im FAMOS-Zeitformat. Der Wert kann mit den Funktionen der Funktionsgruppe #18 "Datum/Uhrzeit" weiterverarbeitet werden

Für Elemente, die nicht in obenstehender Tabelle gelistet sind, ist die Funktion nicht anwendbar.

Beispiele:

Ein Dialog dient zur Einstellung von Filterparametern. Eine Klappliste 'CBoxType' enthält die Einträge 'Hochpass' und 'Tiefpass'. Ein Kombinationsfeld 'CBox_Order' dient zur Vorgabe der Filterordnung und ein Eingabefeld 'Input_Freq' zur Vorgabe der Grenzfrequenz. Auf Knopfdruck wird die Filterung dann ausgeführt:

```
Type  = DlgGetSelectedItem( "CBox_Type")
Order = DlgGetValue( "CBox_Order")
freq  = DlgGetValue("Input_Freq")
IF Order > 1 AND freq > 0
   IF Type = 1  ; Tiefpass
      filtrat = FiltLP( MyData, 0, 0, Order, freq)
   ELSE          ; Hochpass
      filtrat = FiltHP( MyData, 0, 0, Order, freq)
   END
END
```

Siehe auch:

DlgSetValue, DlgGetText

DlgInsertItem

Anwendungsbereich: Anwenderdefinierte Dialoge

Fügt einem Dialogelement (Liste, Tabelle etc.) einen neuen Eintrag hinzu.

Deklaration:

```
DlgInsertItem ( TxElementName, Index, TxInhalt, Option )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes.
Index	Einfügeposition. Der erste Eintrag hat den Index 1. Um den Eintrag am Ende anzuhängen, geben Sie eine 0 an.
TxInhalt	Text für den neuen Eintrag.
Option	Optionsparameter
	0 : Standard.
	1 : Der neue Eintrag wird gegebenenfalls ins Bild gerollt.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, Tabelle

Beispiele:

Ein Listenfeld wird mit den Namen aller Dateien gefüllt, die sich in einem vorgegebenen Verzeichnis befinden.

```
Dir$ = FsDlgSelectDirectory("Verzeichnis auswählen", "", 0)
FileListID = FsFileListNew( Dir$, "*.*", 0, 0, 0)
FileCount = FsFileListGetCount(FileListID)
i = 1
WHILE i <= FileCount
   File$ = FsSplitPath(FsFileListGetName(FileListID, i), 4)
   DlgInsertItem("listFiles", i, File$, 0)
   i = i + 1
END
FsFileListClose(FileListID)
```

Ein Listenfeld wird mit den Namen aller Variablen gefüllt, die sich beim Start des Pogramms in der FAMOS-Variablenliste befinden. Anschließend wird der erste Eintrag selektiert.

```
Count = VarGetInit(0)
i = 1
WHILE i <= Count
   DlgInsertItem("ListVariables", i, VarGetName?(i), 0)
   i = i + 1
END
DlgSelectItem("ListVariables", 1)
```

Ein Listenfeld wird mit allen Werten eines (kurzen) Datensatzes gefüllt.

```
Count = leng?(MyData)
i = 1
WHILE i <= Count
   TxVal = TForm( MyData[i], "")
   DlgInsertItem("list1", 0, TxVal, 0)
   i = i + 1
END
```

Siehe auch:

DlgGetItemCount, DlgGetItemText, DlgSetItemText, DlgFindItem, DlgDeleteItem

DlgInsertTreeItem

Anwendungsbereich: Anwenderdefinierte Dialoge

Fügt einem Dialogelement (Baumansicht) einen neuen Eintrag hinzu.

Deklaration:

```
DlgInsertTreeItem ( TxElementName, Index, TxInhalt, Ebene, Option )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes.
Index	Einfügeposition. Der erste Eintrag hat den Index 1. Um den Eintrag am Ende anzuhängen, geben Sie eine 0 an.
TxInhalt	Text für den neuen Eintrag.
Ebene	Einzug des neuen Eintrags. Wurzeleinträge haben den Einzug 0.
Option	Optionsparameter
	0 : Standard.
	1 : Der neue Eintrag wird gegebenenfalls ins Bild gerollt.

Anwendbar auf:

Baumansicht

Beispiele:

Am Beginn eines Baumansicht werden 2 neue Wurzeleinträge mit Kind-Elementen eingefügt.

```
DlgInsertTreeItem("Tree1", 1, "Wurzel1", 0, 0)
DlgInsertTreeItem("Tree1", 2, "Zweig1", 1, 0)
DlgInsertTreeItem("Tree1", 3, "Zweig2", 1, 0)
DlgInsertTreeItem("Tree1", 4, "Wurzel2", 0, 0)
DlgInsertTreeItem("Tree1", 5, "Zweig3", 1, 0)
```

Siehe auch:

DlgInsertItem, DlgDeleteItem

DlgIsItemSelected

Anwendungsbereich: Anwenderdefinierte Dialoge

Ermittelt, ob ein Eintrag in einer Liste/Tabelle mit Mehrfachselektion selektiert ist.

Deklaration:

```
DlgIsItemSelected ( TxElementName, Index ) -> IstSelektiert
```

Parameter:

TxElementName	Name des zu untersuchenden Dialogelementes.
Index	Index des zu prüfenden Eintrages. Der erste Eintrag hat den Index 1.
IstSelektiert	Selektionszustand des Eintrages
	0 : Nicht selektiert
	1 : Selektiert

Anwendbar auf:

Listenfeld(Mehrfachselektion), Tabelle (Mehrfachselektion)

Beispiele:

Ein Listenfeld in einem Dialog wird mit allen Werten eines (kurzen) Datensatzes gefüllt. Auf Knopdfruck werden alle selektierten Werte auf 0 gesetzt.

--> Ereignis-Sequenz 'Dialog Initialisierung'

```
;Füllen des Listenfeldes mit den Werten des Datensatzes:
Count = leng?(MyData)
i = 1
WHILE i <= Count
   TxVal = TForm( MyData[i], "")
   DlgInsertItem("list1", 0, TxVal, 0)
   i = i + 1
END
```

--> Ereignis-Sequenz 'Knopf gedrückt'

```
;Die selektierten Samples werden auf 0 gesetzt.
Count = DlgGetItemCount("list1")
i = 1
WHILE i <= Count
   IF DlgIsItemSelected("list1", i)
      MyData[i] = 0
      DlgSetItemText("0")
   END
   i = i + 1
END
```

Siehe auch:

DlgGetSelectedItemCount, DlgSetItemSelection, DlgGetSelectedItem

DlgSelectItem

Anwendungsbereich: Anwenderdefinierte Dialoge

Selektiert einen Eintrag in einer Liste/Tabelle mit Einfachselektion

Deklaration:

```
DlgSelectItem ( TxElementName, Index )
```

Parameter:

TxElementName	Name des anzusprechenden Dialogelementes.
Index	Index des zu selektierenden Eintrages. Der erste Eintrag hat den Index 1. Geben Sie eine 0 an, um die aktuelle Selektion zu entfernen.

Anwendbar auf:

Listenfeld mit Einfachselektion, Klappliste, Kombinationsfeld, Tabelle mit Einfachselektion, Baumansicht, Optionsgruppe

Beispiele:

In einem Listenfeld (mit Einfachselektion) wird ein Eintrag gesucht. Falls der Eintrag vorhanden ist, wird dieser selektiert und ggf. in die Sicht gerollt.

```
i = DlgFindItem("list", "100.0")
IF i > 0
   DlgSelectItem("list", i)
END
```

Siehe auch:

DlgGetSelectedItem, DlgSetItemSelection

DlgSetBackColor

Anwendungsbereich: Anwenderdefinierte Dialoge

Die Farbe des Hintergrundes für das gewählte Element wird eingestellt.

Deklaration:

```
DlgSetBackColor ( TxElementName, RGBColor )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes. Wenn ein leerer Text angegeben wird, wird der Dialoghintergrund gesetzt.
RGBColor	Farbe
	>=0 : RGB-Wert der gewünschten Farbe.
	-1 : Transparent
	-2 : Automatisch

Beschreibung:

Der als 2.Parameter übergebene RGB-Wert stellt den Anteil der Intensitäten der 3 Grundfarben Rot, Grün und Blau dar. Um einen solchen Farbwert zu erzeugen, können Sie die FAMOS-Funktion RGB() verwenden.

Der Wert (-1) setzt den Hintergrund auf transparent. Diese Option ist nur für einige Dialogelement-Typen sinnvoll und unterstützt.

Der Wert (-2) setzt den Hintergrund auf automatisch. Für den Dialog bedeutet dies, das die aktuelle Windows-Systemeinstellung verwendet wird. Für Dialogelemente bedeutet diese Option, dass die Hintergrundfarbe des Dialoges übernommen ('geerbt') wird.

Anwendbar auf:

Die Funktion ist nicht anwendbar auf: Datum, Uhrzeit, Kurvenfenster, Menü

Beispiele:

Das Maximum eines Datensatzes wird auf Überschreitung eines Grenzwertes geprüft und das Ergebnis in einem Label ("Label1") im Dialog angezeigt. Falls der Grenzwert überschritten wird, wird die Hintergrund- und Textfarbe des Labels geändert (Rot/Weiß) und eine entsprechende Warnung angezeigt.

```
_max = Max(Daten)
IF _max > 12
   DlgSetTextColor("Label1", RGB(255,255,255))
   DlgSetBackColor("Label1", RGB(255,0,0))
   DlgSetText("Label1", "Grenzwert überschritten!")
ELSE
   DlgSetTextColor("Label1", -2)
   DlgSetBackColor("Label1", -2)
   DlgSetText("Label1", "Datensatz ist OK.")
END
```

Siehe auch:

DlgSetTextColor, DlgSetCellTextColor

DlgSetBarRange

Anwendungsbereich: Anwenderdefinierte Dialoge

Konfiguriert einen Schieberegler bzgl. Bereichsminimum und -maximum sowie Schrittweite.

Deklaration:

```
DlgSetBarRange ( TxElementName, Minimum, Maximum, SchrittKlein, SchrittGross )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes (Schieberegler).
Minimum	Untere Grenze des Wertebereichs.
Maximum	Obere Grenze des Wertebereichs.
SchrittKlein	Legt die kleine Schrittweite fest (Navigations-Tasten 'Links', 'Rechts' ...).
SchrittGross	Legt die große Schrittweite fest (Tasten 'Bild auf' und 'Bild ab').

Anwendbar auf:

Schieberegler

Beispiele:

Der Wertebereich eines Schiebereglers wird festgelegt. Der Bereich wird auf 0 bis 100 (Prozent) festgelegt. Die Richtungs-Tasten verändern den Reglerwert um +/-1, die Tasten 'Bild auf' und 'Bild ab' um +/-10. Der Regler wird auf Mittelstellung initialisiert.

```
DlgSetBarRange("Slider1", 0, 100, 1, 10)
DlgSetValue("Slider1", 50)
```

Siehe auch:

DlgGetBarMin, DlgGetBarMax, DlgGetValue, DlgSetValue

DlgSetCellBackColor

Anwendungsbereich: Anwenderdefinierte Dialoge

Die Farbe des Hintergrundes für die angegebene Tabellenzelle wird eingestellt.

Deklaration:

```
DlgSetCellBackColor ( TxElementName, Zeile, Spalte, RGBColor )
```

Parameter:

TxElementName	Name der zu ändernden Tabelle.
Zeile	Zeilenindex. Die erste Zeile hat den Index 1. Falls 0, wird die Farbe für die gesamte Spalte gesetzt.
Spalte	Spaltenindex. Die erste Spalte hat den Index 1. Falls 0, wird die Farbe für die gesamte Zeile gesetzt.
RGBColor	RGB-Wert der gewünschten Farbe oder (-2) für automatisch.

Beschreibung:

Der als 2.Parameter übergebene RGB-Wert stellt den Anteil der Intensitäten der 3 Grundfarben Rot, Grün und Blau dar. Um einen solchen Farbwert zu erzeugen, können Sie die FAMOS-Funktion RGB() verwenden.

Die Einstellung 'automatisch' (-2) bedeutet hier, dass die Standard-Hintergrundfarbe der Spalte verwendet wird.

Anwendbar auf:

Tabelle

Beispiele:

In eine Tabelle werden diverse Kennwerte eines Datensatzes eingetragen. Wenn das Maximum einen Grenzwert überschreitet, wird die entsprechende Tabellenzelle farblich hervorgehoben.

```
DlgSetCellText("Table", 1, 1, "Länge ")
DlgSetCellValue("Table", 1, 2, leng?(MyData))
DlgSetCellText("Table", 2, 1, "Abtastzeit ")
DlgSetCellValue("Table", 2, 2, xdel?(MyData))
DlgSetCellText("Table", 3, 1, "Maximum ")
value = max(MyData)
DlgSetCellValue("Table", 3, 2, value)
IF value > 100
   DlgSetCellTextColor( "Table", 3, 2, RGB(255,0,0))
   DlgSetCellBackColor( "Table", 3, 2, RGB(127,127,127))
ELSE
   DlgSetCellTextColor( "Table", 3, 2, -2)
   DlgSetCellBackColor( "Table", 3, 2, -2)
END
```

Siehe auch:

DlgSetCellTextColor, DlgSetTextColor, DlgSetBackColor

DlgSetCellText

Anwendungsbereich: Anwenderdefinierte Dialoge

Der Inhalt der angegebenen Tabellenzelle wird neu gesetzt.

Deklaration:

```
DlgSetCellText ( TxElementName, Zeile, Spalte, TxInhalt )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes (Tabelle).
Zeile	Zeilenindex. Die erste Zeile hat den Index 1. Der Spaltenkopf (sofern sichtbar) hat den Index 0.
Spalte	Spaltenindex. Die erste Spalte hat den Index 1. Der Zeilenkopf (sofern sichtbar) hat den Index 0.
TxInhalt	Text, auf den die Zelle gesetzt werden soll.

Anwendbar auf:

Tabelle

Beispiele:

In eine Tabelle werden diverse Kennwerte eines Datensatzes eingetragen. Wenn das Maximum einen Grenzwert überschreitet, wird die entsprechende Tabellenzelle farblich hervorgehoben.

```
DlgSetCellText("Table", 1, 1, "Länge ")
DlgSetCellValue("Table", 1, 2, leng?(MyData))
DlgSetCellText("Table", 2, 1, "Abtastzeit ")
DlgSetCellValue("Table", 2, 2, xdel?(MyData))
DlgSetCellText("Table", 3, 1, "Maximum ")
value = max(MyData)
DlgSetCellValue("Table", 3, 2, value)
IF value > 100
   DlgSetCellTextColor( "Table", 3, 2, RGB(255,0,0))
   DlgSetCellBackColor( "Table", 3, 2, RGB(127,127,127))
ELSE
   DlgSetCellTextColor( "Table", 3, 2, -2)
   DlgSetCellBackColor( "Table", 3, 2, -2)
END
```

Siehe auch:

DlgSetCellValue, DlgGetCellText

DlgSetCellTextColor

Anwendungsbereich: Anwenderdefinierte Dialoge

Die Farbe der Schrift für die angegebene Tabellenzelle wird eingestellt.

Deklaration:

```
DlgSetCellTextColor ( TxElementName, Zeile, Spalte, RGBColor )
```

Parameter:

TxElementName	Name der zu ändernden Tabelle.
Zeile	Zeilenindex. Die erste Zeile hat den Index 1. Falls 0, wird die Farbe für die gesamte Spalte gesetzt.
Spalte	Spaltenindex. Die erste Spalte hat den Index 1. Falls 0, wird die Farbe für die gesamte Zeile gesetzt.
RGBColor	RGB-Wert der gewünschten Farbe oder (-2) für automatisch.

Beschreibung:

Der als 2.Parameter übergebene RGB-Wert stellt den Anteil der Intensitäten der 3 Grundfarben Rot, Grün und Blau dar. Um einen solchen Farbwert zu erzeugen, können Sie die FAMOS-Funktion RGB() verwenden.

Die Einstellung 'automatisch' (-2) bedeutet hier, dass die Standard-Textfarbe der Spalte verwendet wird.

Anwendbar auf:

Tabelle

Beispiele:

Nach Änderung des Inhalts einer Tabellenzelle durch den Anwender wird geprüft, ob der eingegebene Wert eine Zahl > 0 ist. Falls nicht, wird die Zelle rot markiert.

-->Ereignis-Sequenz 'Geändert' der Tabelle

```
Row = PA2
Column = PA3
newValue = DlgGetCellValue(PA1, Row, Column)
IF newValue <= 0
   DlgSetCellTextColor(PA1, Row, Column, RGB(255,0,0))
ELSE
   DlgSetCellTextColor(PA1, Row, Column, -2)
END
```

Siehe auch:

DlgSetCellBackColor, DlgSetTextColor, DlgSetBackColor

DlgSetCellValue

Anwendungsbereich: Anwenderdefinierte Dialoge

Der Wert der angegebenen Tabellenzelle wird neu gesetzt.

Deklaration:

```
DlgSetCellValue ( TxElementName, Zeile, Spalte, Wert )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes (Tabelle).
Zeile	Zeilenindex. Die erste Zeile hat den Index 1. Der Spaltenkopf (sofern sichtbar) hat den Index 0.
Spalte	Spaltenindex. Die erste Spalte hat den Index 1. Der Zeilenkopf (sofern sichtbar) hat den Index 0.
Wert	Zahlenwert, auf den die Zelle gesetzt werden soll.

Beschreibung:

Die Konvertierung des Zahlenwertes in den auszugebenden Text erfolgt entsprechend des für die Tabellenzelle festgelegten Datentyps (Eigenschaft "Daten-Anbindung / Datentyp". Für "Reelle Zahl" oder "automatisch" entspricht dies dem Ergebnis der Funktion TForm(..., "").

Anwendbar auf:

Tabelle

Beispiele:

In eine Tabelle werden diverse Kennwerte eines Datensatzes eingetragen. Wenn das Maximum einen Grenzwert überschreitet, wird die entsprechende Tabellenzelle farblich hervorgehoben.

```
DlgSetCellText("Table", 1, 1, "Länge ")
DlgSetCellValue("Table", 1, 2, leng?(MyData))
DlgSetCellText("Table", 2, 1, "Abtastzeit ")
DlgSetCellValue("Table", 2, 2, xdel?(MyData))
DlgSetCellText("Table", 3, 1, "Maximum ")
value = max(MyData)
DlgSetCellValue("Table", 3, 2, value)
IF value > 100
   DlgSetCellTextColor( "Table", 3, 2, RGB(255,0,0))
   DlgSetCellBackColor( "Table", 3, 2, RGB(127,127,127))
ELSE
   DlgSetCellTextColor( "Table", 3, 2, -2)
   DlgSetCellBackColor( "Table", 3, 2, -2)
END
```

Siehe auch:

DlgSetCellText, DlgGetCellValue

DlgSetItemSelection

Anwendungsbereich: Anwenderdefinierte Dialoge

Setzt oder entfernt die Selektion für einen Eintrag in einer Liste mit Mehrfachselektion.

Deklaration:

```
DlgSetItemSelection ( TxElementName, Index, AnAus )
```

Parameter:

TxElementName	Name des zu untersuchenden Dialogelementes.
Index	Index des zu ändernden Eintrages. Der erste Eintrag hat den Index 1. Um alle Einträge zu (de-)selektieren, geben Sie eine 0 an.
AnAus	Selektion an/aus
	0 : Selektion entfernen
	1 : Eintrag selektieren

Anwendbar auf:

Listenfeld (Mehrfachselektion), Tabelle (Mehrfachselektion)

Beispiele:

Die aktuelle Selektion in einer Liste mit Mehrfachselektion wird invertiert:

```
Count = DlgGetItemCount("list1")
i = 1
   WHILE i <= Count
   Sel = DlgIsItemSelected("list1", i)
   DlgSetItemSelection("list1", i, NOT(Sel))
   i = i + 1
END
```

Siehe auch:

DlgGetSelectedItemCount, DlgIsItemSelected, DlgSelectItem

DlgSetItemText

Anwendungsbereich: Anwenderdefinierte Dialoge

Ersetzt einen Eintrag (Listenfeld, Baumansicht etc.) mit den angegebenen Text.

Deklaration:

```
DlgSetItemText ( TxElementName, Index, TxNeuerInhalt )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes.
Index	Index des zu ändernden Eintrages. Der erste Eintrag hat den Index 1.
TxNeuerInhalt	Neuer Text für den zu ersetzenden Eintrag.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, Baumansicht

Beispiele:

Ein Listenfeld in einem Dialog wird mit allen Werten eines (kurzen) Datensatzes gefüllt. Auf Knopdfruck werden alle selektierten Werte auf 0 gesetzt.

--> Ereignis-Sequenz 'Dialog Initialisierung'

```
;Füllen des Listenfeldes mit den Werten des Datensatzes:
Count = leng?(MyData)
i = 1
WHILE i <= Count
   TxVal = TForm( MyData[i], "")
   DlgInsertItem("list1", 0, TxVal, 0)
   i = i + 1
END
```

--> Ereignis-Sequenz 'Knopf gedrückt'

```
;Die selektierten Samples werden auf 0 gesetzt.
Count = DlgGetItemCount("list1")
i = 1
WHILE i <= Count
   IF DlgIsItemSelected("list1", i)
      MyData[i] = 0
      DlgSetItemText("0")
   END
   i = i + 1
END
```

Siehe auch:

DlgGetItemCount, DlgGetItemText, DlgInsertItem, DlgFindItem, DlgDeleteItem

DlgSetText

Anwendungsbereich: Anwenderdefinierte Dialoge

Der Inhalt bzw. die Beschriftung des angegebenen Elements wird neu gesetzt.

Deklaration:

```
DlgSetText ( TxElementName, TxText )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes.
TxText	Text, auf den das Dialogelement gesetzt werden soll.

Anwendbar auf:

Die Funktion kann nur auf solche Dialogelemente angewendet werden, die eine Beschriftung besitzen oder deren aktueller Zustand durch einen Text ausgedrückt werden kann. Die Interpretation des Textes ist abhängig vom Typ des Elementes.

Element	Bedeutung
Knopf	Beschriftung des Knopfes.
Label	Beschriftung des Labels.
Eingabefeld (einzeilig)	Inhalt des Eingabefeldes.
Eingabefeld (mehrzeilig)	Inhalt des Eingabefeldes.
Kontrollkästchen	Beschriftung des Elementes.
Optionsgruppe	Setzt die Beschriftung für den Rahmen um die Optionsfelder. Nur wirksam für "Rahmen" = "Mit Beschriftung".
Gruppenfeld	Beschriftung des Elementes.
Listenfeld (Einfachselektion)	Falls ein Listeneintrag mit dem angegebenen Text existiert, wird dieser selektiert. Falls nicht vorhanden, wird die aktuelle Selektion beibehalten.
Klappliste	Wie Listenfeld.
Baumansicht	Wie Listenfeld.
Kombinationsfeld	Inhalt des Eingabefeldes.
Dialog	Titelzeile des Dialoges. Für TxElementName ist ein leerer Text anzugeben.
Statuszeile	Inhalt der Statuszeile des Dialoges. Für TxElementName ist "#Status" anzugeben.

Für Elemente, die nicht in obenstehender Tabelle gelistet sind, ist die Funktion nicht anwendbar.

Beispiele:

In einem Dialog werden diverse Kennwerte eines Datensatzes angezeigt, unter anderem auch in einem Eingabefeld 'input_unit' die Y-Einheit. Beim Beenden des Dialoges mit dem Knopf 'OK' wird eine eventuelle Änderung der Einheit durch den Anwender in den Datensatz übernommen.

--> Ereignis-Sequenz 'Dialog Initialisierung'

```
...
DlgSetText("input_unit",  Unit?(MyData,1))
...
```

--> Ereignis-Sequenz 'Gedrückt' des 'OK'-Knopfes

```
TxUnit = DlgGetText("input_unit")
SetUnit(MyData, TxUnit, 1)
DlgCloseDialog(0)
```

Siehe auch:

DlgGetValue, DlgGetValue, DlgGetText

DlgSetTextColor

Anwendungsbereich: Anwenderdefinierte Dialoge

Die Farbe der Schrift wird eingestellt.

Deklaration:

```
DlgSetTextColor ( TxElementName, RGBColor )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes. Wenn ein leerer Text angegeben wird, wird der Standard für den gesamten Dialog gesetzt.
RGBColor	RGB-Wert der gewünschten Farbe oder (-2) für automatisch.

Beschreibung:

Der als 2.Parameter übergebene RGB-Wert stellt den Anteil der Intensitäten der 3 Grundfarben Rot, Grün und Blau dar. Um einen solchen Farbwert zu erzeugen, können Sie die FAMOS-Funktion RGB() verwenden.

Wenn Sie die Textfarbe für dem gesamten Dialog ändern, ändert dies auch die Textfarbe aller derjenigen Dialogelemente, deren Textfarbe auf "automatisch" eingestellt ist.

Der Wert (-2) setzt die Textfarbe auf automatisch. Für den Dialog bedeutet dies, das die aktuelle Windows-Systemeinstellung verwendet wird. Für Dialogelemente bedeutet diese Option, dass die Textfarbe des Dialoges übernommen ('geerbt') wird.

Anwendbar auf:

Die Funktion ist nicht anwendbar auf: Schieberegler, Bild, Kurvenfenster, Menüelemente, Kalender, Uhrzeit

Beispiele:

Das Maximum eines Datensatzes wird auf Überschreitung eines Grenzwertes geprüft und das Ergebnis in einem Label ("Label1") im Dialog angezeigt. Falls der Grenzwert überschritten wird, wird die Hintergrund- und Textfarbe des Labels geändert (Rot/Weiß) und eine entsprechende Warnung angezeigt.

```
_max = Max(Daten)
IF _max > 12
   DlgSetTextColor("Label1", RGB(255,255,255))
   DlgSetBackColor("Label1", RGB(255,0,0))
   DlgSetText("Label1", "Grenzwert überschritten!")
ELSE
   DlgSetTextColor("Label1", -2)
   DlgSetBackColor("Label1", -2)
   DlgSetText("Label1", "Datensatz ist OK.")
END
```

Siehe auch:

DlgSetBackColor, DlgSetCellTextColor

DlgSetValue

Anwendungsbereich: Anwenderdefinierte Dialoge

Der numerische Wert des angegebenen Elements wird neu gesetzt.

Deklaration:

```
DlgSetValue ( TxElementName, Wert )
```

Parameter:

TxElementName	Name des zu setzenden Dialogelementes.
Wert	Zahlenwert, auf den das Dialogelement gesetzt werden soll.

Anwendbar auf:

Die Funktion kann nur auf solche Dialogelemente angewendet werden, deren aktueller Zustand durch eine Zahl ausgedrückt werden kann. Die Interpretation des Wertes ist abhängig vom Typ des Elementes.

Element	Wert
Label	Der Inhalt des Labels wird auf den angebenen Wert gesetzt, Die Formatierung erfolgt entsprechend dem für das Label festgelegten Datenformat.
Listenfeld (Einfachselektion)	Es wird versucht, einen korrespondierenden numerischen Eintrag in der Liste zu selektieren. Falls nicht vorhanden, wird die aktuelle Selektion beibehalten. Die Konvertierung der Zahl erfolgt analog zum Funktionsaufruf TForm(...,""). Praktikabel nur für ganze Zahlen einsetzbar.
Klappliste	Wie Listenfeld.
Kombinationsfeld	Die Zahl wird einen Text konvertiert und das Eingabefeld entsprechend gesetzt. Die Konvertierung der Zahl erfolgt analog zum Funktionsaufruf TForm(...,"").
Eingabefeld (einzeilig)	Wie Kombinationsfeld.
Kontrollkästchen	Nur 1 oder 0 erlaubt. Der Status des Kontrollkästchen wird entsprechend gesetzt. Andere Werte sind nicht erlaubt.
Optionsgruppe	Setzt die auszuwählende Option. Die erste Option hat den Index 1.
Schieberegler	Setzt den Schieberegler auf den angegebenen Wert. Falls der Wert außerhalb der für den Regler festgelegten Bereichsgrenzen liegt, wird der Regler auf das jeweilige Bereichsende gesetzt.
Kalender	Setzt das Element auf den angegebenen Datumswert. Der Wert muß im FAMOS-Zeitformat angegeben werden und kann mit den Funktionen der Funktionsgruppe #18 'Datum/Uhrzeit' erzeugt und verarbeitet werden.
Uhrzeit	Setzt das Element auf die angegebene Uhrzeit. Der Wert muß im FAMOS-Zeitformat angegeben werden und kann mit den Funktionen der Funktionsgruppe #18 'Datum/Uhrzeit' erzeugt und verarbeitet werden.

Für Elemente, die nicht in obenstehender Tabelle gelistet sind, ist die Funktion nicht anwendbar.

Beispiele:

Der Wertebereich eines Schiebereglers wird festgelegt und auf Mittelstellung initialisiert.

```
DlgSetBarRange("Slider1", 0, 100, 1, 10)
DlgSetValue("Slider1", 50)
```

In einem Dialog werden diverse Kennwerte eines Datensatzes angezeigt, unter anderem auch die Triggerzeit in einem 'Kalender'- und einem 'Uhrzeit'-Element. Beim Beenden des Dialoges mit dem Knopf 'OK' wird eine eventuelle Änderung der Zeit durch den Anwender in den Datensatz übernommen.

--> Ereignis-Sequenz 'Dialog Initialisierung'

```
...
Time = Time?(MyData)
DlgSetValue("time", time)
DlgSetValue("date", time)
...
```

--> Ereignis-Sequenz 'Gedrückt' des 'OK'-Knopfes

```
date = DlgGetValue("time")
time = DlgGetValue("date")
SetTime(Mydata, TimeAdd(date, time))
```

Siehe auch:

DlgGetValue, DlgGetText, DlgSetText

DlgShow

Anwendungsbereich: Anwenderdefinierte Dialoge

Der aktive Dialog bzw. ein spezielles Dialogelement wird versteckt bzw. (wieder) angezeigt.

Deklaration:

```
DlgShow ( TxElementName, AnAus )
```

Parameter:

TxElementName	Name des zu ändernden Dialogelementes. Wenn ein leerer Text angegeben wird, wird der gesamte aktive Dialog versteckt bzw. angezeigt.
AnAus	
	0 : Element verstecken
	1 : Element anzeigen

Anwendbar auf:

Die Funktion kann für alle Dialogelemente mit Ausnahme von Menü, Werkzeugleiste und Statuszeile angewendet werden.

Beispiele:

Das Ergebnis einer Berechnung wird in einer EXCEL-Datei gespeichert. Anschließend wird EXCEL geöffnet und die Datei zur Kontrolle angezeigt. Solange EXCEL geöffnet ist, wird der aktuelle Dialog versteckt.

```
Filtrat =  FiltLP(Daten, 0, 0, 6, 100)
fh = FileOpenXLS2("z:\dat\res.xls", "Template #2", 1, 1, 1)
IF fh > 0
   err = FileObjWrite( fh, Filtrat)
   err = FileClose(fh)
   DlgShow("", 0)
   Execute("Excel", "z:\dat\res.xls", "", 0, -1)
   DlgShow("", 1)
END
```

Siehe auch:

DlgEnable

DrDrucke

Anwendungsbereich: Reportgenerator

Ein fertig erstelltes Druckbild wird gedruckt.
Diese Funktion ist zur Kompatibilität noch enthalten. In neu zu erstellenden Sequenzen und Programmen sollte die Funktion RgDocPrint verwendet werden.

Deklaration:

```
DrDrucke ( )
```

Parameter:

Beschreibung:

Die im Druckbild aktuell enthaltene Grafik wird auf dem eingestellten Drucker ausgegeben.

- Der Reportgenerator REPORT.EXE muss als Applikation geladen sein.
- Den Drucker können Sie über den Menüpunkt "Datei: Drucker einrichten.." im Hauptfenster des Reportgenerators einrichten.
- Anstatt einen erstellten Report sofort zu drucken, können Sie auch Reporte sichern und alle zusammen während der Nacht ausdrucken.

Beispiele:

```
TxFehler$ = DrKonfig("report")
DrSetzen("Herbert Mustermann", "Name")
DrDrucke()
```

Ein Report wird üblicherweise erstellt, indem eine Maske geladen wird und dann Elemente auf aktuelle Werte gesetzt werden. Dieser fertige Report kann dann gedruckt werden.

Siehe auch:

RgDocPrint

DrFenst

Anwendungsbereich: Reportgenerator

Das Fenster des Report-Generators wird gesteuert.
Diese Funktion ist zur Kompatibilität noch enthalten. In neu zu erstellenden Sequenzen und Programmen sollte die Funktion RgWindow verwendet werden.

Deklaration:

```
DrFenst ( Auftrag )
```

Parameter:

Auftrag	Auftrag
	1 : Das Druckbild wird in normaler Größe gezeigt.
	2 : Das Druckbild wird als Sinnbild gezeigt.
	3 : Das Druckbild wird geschlossen.

Beschreibung:

Das Fenster wird entsprechend dem Steuer-Parameter EwAuftrag gesteuert.

- Wenn das Druckbild vorher noch nicht geladen war, wird es automatisch gestartet.
- Diese Funktion ist gegenüber einem Aufruf der ausführbaren Datei REPORT.EXE vorzuziehen, da sich dieser Name in zukünftigen Versionen ändern kann.

Beispiele:

```
DrFenst(1)
```

Das Druckbild wird vom Programm aus als normales Fenster gezeigt. Der Benutzer kann nun direkt im Reportgenerator arbeiten.

Siehe auch:

RgWindow

DrKonfig

Anwendungsbereich: Reportgenerator

Eine Konfigurationsdatei wird in den Reportgenerator geladen.
Diese Funktion ist zur Kompatibilität noch enthalten. In neu zu erstellenden Sequenzen und Programmen sollte die Funktion RgDocOpen verwendet werden.

Deklaration:

```
DrKonfig ( TxDateiname )
```

Parameter:

TxDateiname	Der Name der zu ladenden Datei.

Beschreibung:

In den Reportgenerator wird eine neue Konfiguration geladen. Diese geladene Konfiguration wurde vorher mit dem Reportgenerator erstellt und als Datei abgespeichert. I. a. werden Konfigurationen als Masken erstellt. In einer Maske werden Platzhalter benutzt, die erst noch durch aktuelle Werte (z. B. Meßwerte oder variierende Kommentare) gefüllt werden müssen.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird ".drb" angenommen.

Falls kein voller Pfadname angeben ist, wird das in der aufrufenden Applikation festgelegte Standardverzeichnis (Verzeichnis für Kurvenkonfigurationen) verwendet.

- Sie können Masken und fertige Reporte laden.
- Die Applikation REPORT.EXE wird als Sinnbild geladen, falls sie vorher noch nicht gestartet war. Dieses indirekte Aufrufen ist einem direkten Starten der Applikation stets vorzuziehen.
- Es gibt höchstens ein Fenster des Report-Generators auf dem Bildschirm.
- Soll ein Dateiname keine Dateinamen-Erweiterung haben, wird der Dateiname mit einem "." abgeschlossen.

Beispiele:

```
TxFehle$ = DrKonfig("report")
DrSetzen("Herbert Mustermann", "Name")
```

Nachdem eine Maske geladen wurde, werden mit der Funktion "DrSetzen" die Platzhalter durch wirkliche Werte ersetzt.

Siehe auch:

RgDocOpen

DrMove

Anwendungsbereich: Reportgenerator

Anordnung eines Objektes relativ zu einem anderen Objekt.
Diese Funktion ist zur Kompatibilität noch enthalten. In neu zu erstellenden Sequenzen und Programmen sollte die Funktion RgObjMove verwendet werden.

Deklaration:

```
DrMove ( TxTitel, TxReferenz, Xmm, Ymm, Null ) -> Null
```

Parameter:

TxTitel	Titel des anzuordnenden Objektes
TxReferenz	Titel des Referenz-Objektes
Xmm	x-Abstand der linken Kanten in mm
Ymm	y-Abstand der oberen Kanten in mm
Null	Reserviert, immer auf 0 zu setzen
Null	Ergebnis, immer 0

Beschreibung:

Ein Objekt mit dem Titel <TxTitel> wird relativ zum Objekt mit dem Titel <TxTitelRef> angeordnet. Dabei wird der Abstand der linken Kanten beider Objekte durch den Parameter <EwXmm> in mm angegeben. Der Abstand der oberen Kanten der Objekte wird durch <EwYmm> in mm festgelegt. Wenn das Referenz-Objekt mit dem Titel <TxTitelRef> nicht existiert oder ein leerer String ("") als <TxTitelRef> übergeben wird, stellt die linke obere Ecke des Blattes den Bezugspunkt dar.

- Die Abstände <EwXmm> und <EwYmm> können mit Nachkommastelle angegeben werden.
- Bei der Positionierung wird keine Rasterung durchgeführt.
- Werden Kurvenobjekte angesprochen, gilt die linke obere Ecke des Koordinatensystems als Bezugspunkt.

Beispiele:

```
Ew0 = DrMove("Text1", "Text2", 10, -15,0)
```

Das Objekt mit dem Titel "Text1" wird auf dem Blatt so angeordnet, dass sich dessen linke Kante 10mm rechts von der linken Kante des Objektes mit dem Titel "Text2" und dessen obere Kante 15mm über der oberen Kante dieses Objektes befindet.

```
Ew0 = DrMove("Kurve1", "", 5, 5,0)
```

Ein Kurvenobjekt mit dem Titel "Kurve1" wird auf dem Blatt so angeordnet, dass sich die linke obere Ecke seines Koordinatensystems 5mm rechts und 5mm unter der linken oberen Ecke des Blattes befindet.

Siehe auch:

DrRdClip

Anwendungsbereich: Reportgenerator

Kopieren des Inhalts der Zwischenablage in das Druckbild.

Deklaration:

```
DrRdClip ( TxTitelRef, Xmm, Ymm, Null, Null, Format, Attribut, Null ) -> Fehler
```

Parameter:

TxTitelRef	Titel des Referenz-Objektes
Xmm	x-Abstand der linken Kanten in mm
Ymm	y-Abstand der oberen Kanten in mm
Null	Reserviert, immer auf 0 zu setzen
Null	Reserviert, immer auf 0 zu setzen
Format	Datei-Format
	0 : automatische Auswahl des Datei-Formats
	1 : Text
	2 : Bitmap (Pixelgrafik)
	3 : Metadatei (Vektorgrafik)
Attribut	Attribut, Bedeutung siehe unten
Null	Reserviert, immer auf 0 zu setzen
Fehler	Fehler-Status
	0 : Kein Fehler aufgetreten
	1 : Falscher Parameter
	2 : Angegebenes Datei-Format und Datei-Format der Zwischenablage verschieden oder Zwischenablage leer
	3 : Druckbild nicht vorhanden oder nicht bedienbar
	4 : Nicht genügend Speicher oder Objekt zu groß

Beschreibung:

Bedeutung des Attribut-Parameters

EwAttrib	Eigenschaften des Objektes
0:	neues Objekt erscheint im Vordergrund
Addition von +1:	neues Objekt erscheint im Hintergrund
+2:	Inhalt der Zwischenablage wird in das Referenz-Objekt kopiert, wobei dessen Eigenschaften erhalten bleiben

Der Inhalt der Zwischenablage wird in das Druckbild kopiert, wobei das neue Objekt relativ zum Referenz-Objekt mit dem Titel <TxTitelRef> angeordnet wird. Der Abstand der linken Kanten beider Objekte wird durch den Parameter <EwXmm> in mm angegeben. Der Abstand der oberen Kanten der Objekte wird durch <EwYmm> in mm festgelegt. Wenn das Referenz-Objekt mit dem Titel <TxTitelRef> nicht existiert oder ein leerer String ("") als <TxTitelRef> übergeben wird, stellt die linke obere Ecke des Blattes den Bezugspunkt dar.

- Die Abstände <EwXmm> und <EwYmm> können mit Nachkommastelle angegeben werden.
- Bei der Positionierung wird keine Rasterung durchgeführt.
- Werden Kurvenobjekte angesprochen, gilt die linke obere Ecke des Koordinatensystems als Bezugspunkt.
- Wird ein neues Textobjekt angelegt und ein bestehendes Textobjekt als Referenz-Objekt angegeben, dann werden Schrift und Farbe dieses Objektes übernommen.
- Wird der Inhalt der Zwischenablage in das Referenz-Objekt kopiert (EwAttrib = 2 oder 3), wird der Typ des Referenz-Objektes entsprechend dem Datei-Format der Zwischenablage geändert. Für die Parameter <EwXmm> und <EwYmm> ist der Wert Null (0) zu übergeben.

Beispiele:

```
Fehler= DrRdClip("", 5, 10, 0, 0, 1, 1, 0)
```

Der Inhalt der Zwischenablage wird auf dem Blatt so angeordnet, dass sich dessen linke obere Ecke 5mm rechts und 10mm unter der linken oberen Ecke des Blattes befindet. Als Datei-Format wird Text angegeben. Das neue Objekt wird in den Hintergrund gesetzt.

```
Fehler= DrRdClip("Text1", 0, 0, 0, 0, 1, 3, 0)
```

Der Inhalt der Zwischenablage wird in das Referenz-Objekt mit dem Titel "Text1" kopiert. Eventuell vorhandener Text im Referenz-Objekt wird dabei überschrieben. Das Objekt wird in den Hintergrund gesetzt.

Siehe auch:

DrMove

DrSetzen

Anwendungsbereich: Reportgenerator

In einem Druckbild wird ein Platzhalter durch einen tatsächlichen Wert ersetzt.
Diese Funktion ist zur Kompatibilität noch enthalten. In neu zu erstellenden Sequenzen und Programmen sollten die objektspezifischen Funktionen (Menüs "Tabelle", "Kurve", "Text") verwendet werden.

Deklaration:

```
DrSetzen ( Inhalt, TxTitel )
```

Parameter:

Inhalt	Der neue Inhalt, der den Platzhalter ersetzen soll. Welcher Typ (Datensatz, Einzelwert, Text) sinnvoll ist, hängt vom Platzhalter ab.
TxTitel	Der Titel des Objektes im Druckbild, das verändert werden soll.

Beschreibung:

Im Druckbild gibt es Text-Objekte und Kurven-Objekte, die beide über die Funktion DrSetzen verändert werden können. Die Objekte werden dabei über ihren Titel ausgewählt.

Textobjekte:
Im Fall eines Text-Objektes können im Text selbst Platzhalter angegeben werden, die mit einem Doppelkreuz "#" eingeleitet werden. Wenn mit der Funktion "DrSetzen" eine Übertragung an ein Text-Objekt stattfindet, wird der erste geeignete Platzhalter ersetzt. Für reelle Zahlen sind die Platzhalter "#e" und "#f" geeignet, für Text "#s".

Folgende Platzhalter sind für Text-Objekte definiert:

#d	Aktuelles Datum im Windows-Format
#z	Aktuelle Uhrzeit im Windows-Format
#u	Einheit eines Einzelwertes bzw. y-Einheit eines Datensatzes
#e?	Zahl im Gleitkomma-Format, Angabe der Nachkomma-Stellen
#f?.?	Zahl im Festkomma-Format, Angabe der Vor- und Nachkommastellen
#s	Text

Dabei stehen "?" statt Ziffern.

Kurvenobjekte:
Im Fall eines Kurven-Objektes gibt es keinen Platzhalter im eigentlichen Sinn. Wenn ein Kurvenfenster an ein Kurven-Objekt übertragen wird, wird die alte Grafik durch die neue ersetzt. Falls das Kurven-Objekt leer war (eine Maske), dann wird nur die neue Grafik eingetragen.

Die Identifizierung des Kurvenfensters erfolgt über dessen Bezugsdatensatz. Der Bezugsdatensatz eines Kurvenfensters wird beim Erzeugen des Kurvenfensters mit den Cw*(..)-Funktionen des Kurven-Kits oder dem FAMOS-Befehl "SHOW" festgelegt. Oftmals entspricht er dem ersten angezeigten Datensatz im Fenster und dem Fenstertitel.

- Die Funktion kann nur angewendet werden, wenn eine Druckbild-Konfiguration geladen ist.
- Eine genaue Beschreibung der Platzhalter finden Sie im Kapitel "Reportgenerator" bei der Beschreibung der Textobjekte.
- Die Platzhalter "#d" (aktuelles Datum) und "#z" (aktuelle Uhrzeit) werden beim ersten Zugriff auf ein Text-Objekt mit der Funktion "DrSetzen" ersetzt.
- Wenn Datensätze an Text-Objekte mit Platzhaltern für reelle Zahlen übertragen werden, so wird für jedes Sample im Datensatz (jeden y-Wert) ein Platzhalter ersetzt. Sie können auf diese Weise eine ganze Tabelle (bzw. eine Spalte oder Zeile) mit einem einzigen Befehl übertragen.

Einmal ersetzte Platzhalter in Text-Objekten bleiben für immer ersetzt. Wenn Sie einen anderen Report erstellen möchten, müssen Sie zuerst wieder die Maske mit der Funktion DrKonfig laden!

Beispiele:

Zuerst wird eine Maske geladen.

```
TxFehler$ = DrKonfig("report")
```

Im Druckbild (in dieser Maske) sind 2 Text-Objekte definiert:

Titel	Inhalt
Name	#s
Zahl	x: #e2 und #f2.1

Die folgenden Zeilen werden ausgeführt:

```
DrSetzen("Herbert", "Name")
```

```
DrSetzen(27.5, "Zahl")
DrSetzen(28.9, "Zahl")
```

Danach enthalten die Text-Objekte folgende Texte:

Titel	Inhalt
Name	Herbert
Zahl	x: 2.75E+01 und 28.9

Siehe auch:

RgTextSet, RgTextGet, RgTextSetData, RgCurveSet, RgTableSetCell, RgTableSetRow, RgTableSetColumn

DrSicher

Anwendungsbereich: Reportgenerator

Die Konfiguration des Druckbildes wird in einer Datei gesichert.
Diese Funktion ist zur Kompatibilität noch enthalten. In neu zu erstellenden Sequenzen und Programmen sollte die Funktion RgDocSave verwendet werden.

Deklaration:

```
DrSicher ( TxDateiname ) -> TxFehler
```

Parameter:

TxDateiname	Der Name der zu erzeugenden Datei. Kann mit komplettem Verzeichnis und Dateinamen-Erweiterung angegeben werden.
TxFehler	Wenn die Funktion die Datei erfolgreich geschrieben hat, wird ein leerer Text zurückgegeben, ansonsten ein Fehlertext

Beschreibung:

Die aktuell im Reportgenerator REPORT.EXE vorliegende Konfiguration wird in einer Datei gesichert.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird ".drb" angenommen.

Falls kein voller Pfadname angeben ist, wird das in der aufrufenden Applikation festgelegte Standardverzeichnis (Verzeichnis für Kurvenkonfigurationen) verwendet.

- Sie können Masken und fertige Reporte sichern. Beachten Sie, dass Sie nicht versehentlich eine Maske mit einem komplett ausgefüllten Report überschreiben. Diese Datei ist dann als Maske nicht mehr benutzbar (z. B. sind alle Platzhalter ersetzt).
- Soll ein Dateiname keine Dateinamen-Erweiterung haben, wird der Dateiname mit einem "." abgeschlossen.

Beispiele:

```
;... Der Report wird erstellt.
DrSetzen("Herbert Mustermann", "Name")
;...
Fehler$ = DrSicher("report")
IF TComp(Fehler$,"")
   ok = BoxMessage("Achtung!", Fehler$, "!1")
END
```

Ein Report wird erstellt und in der Datei "report.drb" gesichert. Es wird abgefragt, ob ein Fehler beim Sichern aufgetreten ist. Wenn ja, wird eine Meldung ausgegeben.

Siehe auch:

RgDocSave

DrTitleI

Anwendungsbereich: Reportgenerator

Die Anzahl der Objekte mit Titel wird ermittelt.
Diese Funktion ist zur Kompatibilität noch enthalten. In neu zu erstellenden Sequenzen und Programmen sollte die Funktion RgDocSave verwendet werden.

Deklaration:

```
DrTitleI ( Null ) -> EwAnzahl
```

Parameter:

Null	Reserviert, immer auf 0 zu setzen
EwAnzahl	Anzahl der Objekte mit Titel.

Beschreibung:

Die Anzahl der im Druckbild vorhandenen Objekte mit Titel wird ermittelt.

Beispiele:

```
Anzahl = DrTitleI(0)
```

Siehe auch:

RgObjGetCount, RgObjGetTitle, RgObjGetType, DrTitleN

DrTitleN

Anwendungsbereich: Reportgenerator

Der Titel eines Objektes wird ermittelt.
Die Funktion "DrTitleN" ist zur Kompatibilität noch enthalten. In neu zu erstellenden Sequenzen und Programmen sollte die Funktion "RgObjGetTitle" verwendet werden.

Deklaration:

```
DrTitleN ( ObjektIndex, Null ) -> TxTitel
```

Parameter:

ObjektIndex	Mit diesem Index wird das Objekt ausgewählt.
Null	Reserviert, immer auf 0 zu setzen
TxTitel	Titel des angesprochenen Objektes.

Beschreibung:

Der Titel des durch EwIndex spezifizierten Objektes im Druckbild wird ermittelt. Hat das Objekt keinen Titel oder ist EwIndex größer als die Anzahl der im Druckbild vorhandenen Objekte, wird ein leerer String ("") zurückgeliefert.

- EwIndex beginnt immer bei 1.

Beispiele:

```
Anzahl = DrTitleI(0)
Titel$ = DrTitleN(Anzahl, 0)
```

Der Titel des letzten Objektes wird ermittelt.

Siehe auch:

RgObjGetTitle, RgObjGetCount, RgObjGetType, DrTitleI

DSPLOAD

Laden einer DIGISKOP(c)-Datei (Copyright remes GmbH)
Alternativer Name: **DSPLADEN**

Deklaration:

```
DSPLOAD EwKarte EwKanal Dateiname Variablenname
```

Parameter:

EwKarte	Zu ladende Karte (1..5)
EwKanal	Zu ladender Kanal (1...160)
Dateiname	Name der zu ladenden Datei
Variablenname	Variable, in die der Dateiausschnitt eingetragen wird

Beschreibung:

Eine DIGISKOP(c)-Dateiengruppe, definiert von der Firma remes, wird in imc FAMOS geladen. Eine DIGISKOP(c)-Dateiengruppe besteht aus einer *.DSP, *.PTR und *.DRD-Datei. Dabei unterscheiden sich diese jeweils drei Dateinamen nur durch ihre Erweiterung. Diese Dateien enthalten Informationen und Messwerte von 1 bis 5 Datenaufnahme-Karten und pro Karte von 1 bis 160 Kanälen. Über die Parameter [Karte] und [Kanal] kann ein gewünschter Dateiausschnitt eingelesen werden. Der Parameter [Karte] darf Werte zwischen 1 und 5 annehmen, der Parameter "Kanal" Werte zwischen 1 und 160.

Der ausgewählte Dateiausschnitt wird mit der unter [Variablenname] angegebenen Bezeichnung geladen.

- Der Dateiname kann ein vollständiger Pfadname inklusive Verzeichnis und Dateinamen-Erweiterung sein, darf aber auch ohne beides angegeben werden. Dann wird die Datei in dem Verzeichnis gesucht, aus dem FAMOS aktuell Daten lädt. Die Erweiterung wird automatisch gewählt.
- Der Dateiname darf auch in Anführungszeichen angegeben werden. Dies ist z.B. dann notwendig, wenn der Pfad Leerzeichen enthält.

Beispiele:

```
DSPLOAD 4 1 DSCP
DSPLOAD 4 1 c:\dig\DSCP.DSP karte4_1
```

In beiden Fällen wird aus der Dateiengruppe "DSCP.*" der erste Kanal der vierten Karte gelesen und in FAMOS unter der Bezeichnung "karte4_1" eingetragen.

```
kart = 4
kan = 1
DSPLOAD kart kan DSCP daten
```

Zwei Variablen definieren den gewünschten Bereich der Dateiengruppe "DSCP". Eingetragen wird er unter der Bezeichnung "daten".

```
DSPLOAD 4 1 "c:\Meine Daten\DSCP.DSP" karte4_1
```

Der Pfadname enthält Leerzeichen und muss deshalb in Anführungszeichen angegeben werden.

Siehe auch:

FileLoad, FileOpenFAS, LDIR

e

Eulersche Zahl e = 2.718...

Beispiele:

Der natürliche Logarithmus der vordefinierten Konstante e ist gleich 1:

```
one = ln(e)
```

Die beiden folgenden Formeln sind äquivalent:

```
y = exp(x)
y = e ^ x
```

Siehe auch:

ln, exp

eFit

Exponentielle Regression, Anpassung an Exponentialfunktion
Alternativer Name: **eRegr**

Deklaration:

```
eFit ( Daten ) -> eFunktion
```

Parameter:

Daten	Datensatz, zu dem eine anpassende e-Funktion gefunden werden soll. [ND],[XY]
eFunktion	An die e-Funktion angepasster Datensatz

Beschreibung:

Es wird ein Datensatz entsprechend der Gleichung

```
f(x) = A * exp(B*x)
```

bestimmt, der den gegebenen Datensatz am besten annähert.

Nach Beendigung der Funktion wird die ermittelte Gleichung in der Ausgabebox im FAMOS-Hauptfenster angezeigt. Die Gleichung wird mit Einheiten gezeigt und hat die Form:

```
f(x) = 13.81 [V] * exp (116.8 [Ohm] * x)
```

Ein Aufruf der Funktion mit einem leeren Text als Parameter liefert die zuletzt berechneten Koeffizienten B, A in Form eines Datensatzes mit 2 Werten.

Normaler Datensatz [ND]

Der erzeugte Datensatz hat die Länge und Einheit des übergebenen Datensatzes, aber maximal die Länge 1000. Bei Kürzung der Länge wird die Abtastzeit verändert, so dass der erzeugte Datensatz über demselben Intervall definiert ist.

XY-Datensatz [XY]

Der erzeugte Datensatz hat die Länge 1000. Er ist über demselben x-Intervall wie der Quelldatensatz definiert.

- Die Zahlenwerte des übergebenen Datensatzes müssen größer Null sein.
- Treten Zahlenwerte kleiner Null auf, erscheint mit der ermittelten Gleichung eine Warnung. Diese Gleichung sagt nichts aus und sollte auf keinen Fall als richtig angesehen werden. In einer Sequenz sollte man darum immer die ermittelten Koeffizienten prüfen. Wenn diese 1 und 0 sind, so gilt das Ergebnis als ungültig.
- Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Beispiele:

Die exponentielle Regression für einen Datensatz wird berechnet. Anschließend werden noch die beiden Koffizienten der e-Funktion ermittelt:

```
ExpRegression = eFit(Daten)
coeff = eFit("")
factorB = coeff[1]
factorA = coeff[2]
```

Siehe auch:

LFit, exp, ln, Appro, ApproNonLin, Poly

ELSE

Leitet in einer IF- Verzweigung jene Anweisungen ein, die ausgeführt werden sollen, falls weder die IF-Bedingung noch eventuell vorhandene ELSEIF-Bedingungen erfüllt waren.
Alternativer Name: **SONST**

Deklaration:

```
ELSE
```

Beschreibung:

Das Ende der zu diesem Block zugehörigen Anweisungen wird durch ein nachfolgendes END markiert.

Ein IF-Block kann beliebig viele ELSEIF-Verzweigungen enthalten. Eine eventuell ebenfalls angegebene ELSE-Verzweigung muss an letzter Stelle der Kette stehen.

Statt IF/ELSEIF/ELSE kann in vielen Anwendungsfällen auch bequemer die SWITCH/CASE/DEFAULT-Anweisung verwendet werden.

Beispiele:

Wenn das Maximum einer Variablen größer 0 ist, wird diese in einem Kurvenfenster angezeigt. Ansonsten wird die Variable gelöscht.

```
Maxi = Max(data)
IF Maxi
   SHOW data
ELSE
   DELETE data
END
```

Beim Laden einer Datei wird der Rückgabewert geprüft und ggf. eine Fehlermeldung ausgegeben.

```
fh = FileOpenDSF("Channel1.dat", 0)
IF fh = 0
   Pause ==> Fehler beim Laden der Datei <==
ELSE
 ; ...
   FileClose(fh)
END
```

Aus einem äquidistant abgetasteten Datensatz [data] soll ein Stück ausgeschnitten werden. Die beiden Grenzen [xmin] und [xmax]sind vorher vom Anwender eingegeben worden und werden nun auf Gültigkeit überprüft:

```
IF xmin < xoff?(data) OR xmin <= 0
   txError = "Illegal lower limit"
ELSEIF xmax > (xoff?(data)+(leng?(data)-1)*xdel?(data))
   txError = "Illegal upper limit"
ELSEIF xmin >= xmax
   txError = "Illegal limit relation"
ELSE
   result = Cut(data, xmin, xmax)
END
```

Siehe auch:

ELSEIF, ELSE, SWITCH

ELSEIF

Leitet in einer IF-Verzweigung einen weiteren Bedingungszweig ein. Dieser wird abgearbeitet, falls die vorherigen Bedingungszweige nicht durchlaufen wurden und die hier angegebene Bedingung erfüllt ist.

Deklaration:

```
ELSEIF Bedingung
```

Parameter:

Bedingung	Nur wenn die Bedingung erfüllt ist (Auswertung liefert einen Wert >0), werden die Anweisungen des folgenden Blocks durchlaufen.

Beschreibung:

Das Ende der zu diesem Block zugehörigen Anweisungen wird durch ein nachfolgendes weiteres ELSEIF, ein ELSE oder END markiert.

Als Bedingung kann z.B. eine Einzelwert-Variable oder ein komplexerer Ausdruck unter Verwendung von logischen Operatoren (AND,OR..) und/oder Vergleichsoperatoren (<, =, ...) angegeben werden.

Ein IF-Block kann beliebig viele weitere ELSEIF-Verzweigungen enthalten. Ein eventuell ebenfalls angegebene ELSE-Verzweigung muss an letzter Stelle der Kette stehen.

Statt IF/ELSEIF/ELSE kann in vielen Anwendungsfällen auch bequemer die SWITCH/CASE/DEFAULT-Anweisung verwendet werden.

Beispiele:

Aus einem äquidistant abgetasteten Datensatz [data] soll ein Stück ausgeschnitten werden. Die beiden Grenzen [xmin] und [xmax]sind vorher vom Anwender eingegeben worden und werden nun auf Gültigkeit überprüft:

```
IF xmin < xoff?(data) OR xmin <= 0
   txError = "Illegal lower limit"
ELSEIF xmax > (xoff?(data)+(leng?(data)-1)*xdel?(data))
   txError = "Illegal upper limit"
ELSEIF xmin >= xmax
   txError = "Illegal limit relation"
ELSE
   result = Cut(data, xmin, xmax)
END
```

Siehe auch:

IF, ELSE, SWITCH

EMPTY

Leerer Datensatz mit Länge 0.
Alternativer Name: **LEER**

Beispiele:

Ein typischer Einsatz dieser Konstanten ist die Initialisierung eines Datensatzes, der in einer nachfolgenden Schleife schrittweise konstruiert werden soll.

```
result = EMPTY
FOR i = 1 to ...
   newValue = ...
   result = Join(result, newValue)
END
```

Siehe auch:

Join

END

Der Befehl markiert das Ende einer Folge von Anweisungen in einer Schleife oder Verzweigung.
Alternativer Name: **ENDE**

Deklaration:

```
END
```

Beschreibung:

Letzter Befehl einer Schleife (WHILE, FOR, FOREACH), einer bedingten Verzweigung (IF, ELSEIF) oder einer Fallunterscheidung (SWITCH, CASE, DEFAULT).

Beispiele:

Wenn das Maximum einer Variablen größer 0 ist, wird diese in einem Kurvenfenster angezeigt.

```
Maxi = Max(data)
IF Maxi
   SHOW data
END
```

Zu einem Wert, der normalerweise im Bereich von 0 bis 100 liegt, wird ein beschreibender Text gebildet. Liegt er außerhalb dieses Bereichs, wird eine Fehlermeldung angezeigt.

```
SWITCH Round(value, 1)
CASE 0 TO 48
    Tx = "Lower half"
CASE 49,50,51
    Tx = "Center"
CASE 52 To 100
   Tx = "Upper half"
DEFAULT
   PAUSE Invalid Value
END
```

Siehe auch:

IF, SWITCH, FOR, WHILE, FOREACH, BREAK, CONTINUE

END_PARALLEL

Verfügbar ab: Professional Edition

Deklaration:

```
END_PARALLEL
```

Beschreibung:

Ein mit BEGIN_PARALLEL geöffneter paralleler Ausführungsblock wird geschlossen. Es wird gewartet, bis alle zuvor gestarteten parallelen Ausführungen beendet sind.

Siehe auch:

BEGIN_PARALLEL

Unterstützt ab:

Version 2022, Editionen: Professional, Enterprise, Runtime

Envelope1

Bestimmung der oberen und unteren Hüllkurve eines Datensatzes nach dem Intervall-Sekanten-Verfahren.

Deklaration:

```
Envelope1 ( Daten, EwZeitIntervall, EwOption, Null ) -> ErgHüllkurve
```

Parameter:

Daten	Datensatz, dessen Hüllkurve berechnet werden soll. [ND]
EwZeitIntervall	Zeitintervall, skaliert in der x-Einheit des Datensatzes
EwOption	Definiert die Art der Berechnung
	1 : Obere Hüllkurve
	2 : Untere Hüllkurve
	3 : Obere Hüllkurve des Betrages der Eingangsdaten
	4 : Obere Hüllkurve, aber zusätzlich interpolierte Werte an den Stellen, an denen die untere Hüllkurve Stützstellen hat.
	5 : Untere Hüllkurve, aber zusätzlich interpolierte Werte an den Stellen, an denen die obere Hüllkurve Stützstellen hat.
Null	Reservierter Parameter, immer 0.
ErgHüllkurve	Resultierende Hüllkurve

Beschreibung:

Algorithmus am Beispiel der oberen Hüllkurve:

Der erste Wert der Eingangsdaten wird gespeichert. Im Intervall wird ab dem Intervall-Start nach der Sekante der größten Steigung gesucht. Dieser 2. Schnittpunkt der Sekante wird gespeichert. Ab diesem Punkt beginnt ein neues Intervall. Der Vorgang wiederholt sich. Der letzte Wert der Eingangsdaten wird auch gespeichert.

- Steigungen werden als gleich angesehen, wenn sie um weniger als 1e-13 voneinander abweichen. Wenn im Intervall mehrmals die gleiche Steigung vorgefunden wird, wird das letzte Auftreten bevorzugt.
- Die Optionen 4 und 5 sind deutlich rechenintensiver als die einfachen Hüllkurven. Diese Optionen sind dann sinnvoll, wenn die obere und untere Hüllkurve miteinander verrechnet werden sollen. Dann sollten beide mit den Optionen 4 und 5 berechnet werden.
- Mit der Funktion Sub() u. ä. kann auch bei EwOption = 1, 2 direkt die Differenz gebildet werden, auch wenn die Stützstellen nicht an den selben Stellen liegen.

Beispiele:

Die Schwingungen im Datensatz haben eine Periode von bis zu 2.0 Sekunden. Das wird etwas aufgerundet auf 5.0. Die Funktion soll alle Spitzenwerte zu einer oberen Hüllkurve verbinden, zwischendurch aber nicht in die Täler abfallen.

```
Hüllkurve = Envelope1(zeitdaten, 5.0, 1, 0)
```

Schnelle Bestimmung der Differenz von oberer und unterer Hüllkurve:

```
ObereHüll = Envelope1(zeitdaten, 1e-3, 1, 0)
UntereHüll = Envelope1(zeitdaten, 1e-3, 2, 0)
Differenz = Sub(ObereHüll, UntereHüll, 0)
```

Rechenaufwendige Bestimmung der Differenz:

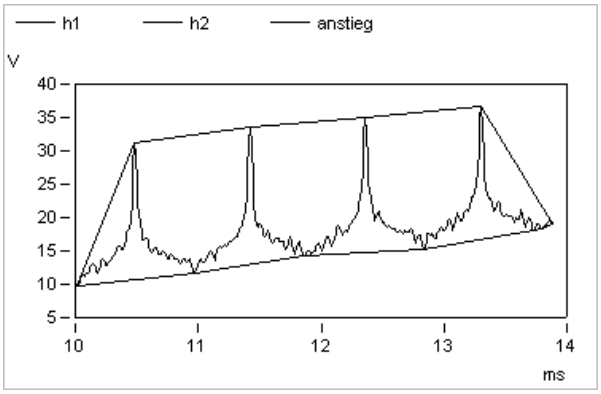
```
ObereHüll = Envelope1(zeitdaten, 1e-3, 4, 0)
UntereHüll = Envelope1(zeitdaten, 1e-3, 5, 0)
Differenz = XYof(ObereHüll.x, ObereHüll.y - UntereHüll.y)
```

Vergleich von Datensatz und Hüllkurve Sample für Sample:

```
ObereHüll = Envelope1(zeitdaten, 1e-3, 1, 0)
Oe = xydt(ObereHüll.x, ObereHüll.y, xdel?(zeitdaten))
; "Oe" ist nun ein Datensatz, der genauso wie "zeitdaten" abgetastet ist.
Abweichung = Oe - zeitdaten
```

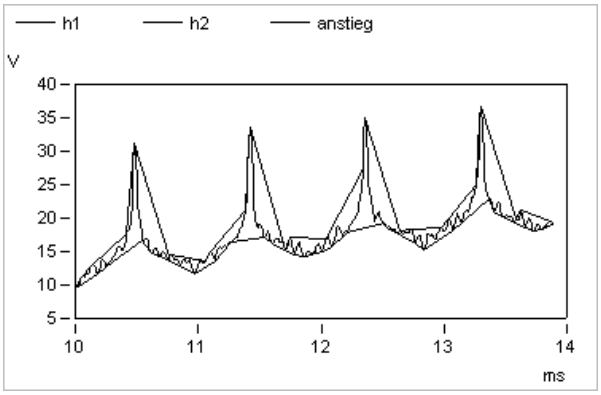
Der mit imc FAMOS gelieferte Datensatz "anstieg.dat" kann zum Ausprobieren auf folgende Weise behandelt werden:

```
ObereHülle = Envelope1(anstieg, 1e-3, 1, 0)
UntereHüll = Envelope1(anstieg, 1e-3, 2, 0)
```



Die Nutzung eines kürzeren Intervalls macht den Schlauch enger:

```
h1 = Envelope1(anstieg, 3e-4, 1 ,0)
h2 = Envelope1(anstieg, 3e-4, 2 ,0)
```



Siehe auch:

Envelope2, Sub, XYof

Envelope2

Bestimmung der oberen und unteren Hüllkurve eines Datensatzes nach dem Intervall-Sekanten-Verfahren mit Vorgabe von Stützstellen.

Deklaration:

```
Envelope2 ( Daten, EwZeitIntervall, EwOption, Null, Stützstellen ) -> ErgHüllkurve
```

Parameter:

Daten	Datensatz, dessen Hüllkurve berechnet werden soll. [ND]
EwZeitIntervall	Zeitintervall, skaliert in der x-Einheit des Datensatzes
EwOption	Definiert die Art der Berechnung
	1 : Obere Hüllkurve
	2 : Untere Hüllkurve
	3 : Obere Hüllkurve des Betrages der Eingangsdaten
	4 : Obere Hüllkurve, aber zusätzlich interpolierte Werte an den Stellen, an denen die untere Hüllkurve Stützstellen hat.
	5 : Untere Hüllkurve, aber zusätzlich interpolierte Werte an den Stellen, an denen die obere Hüllkurve Stützstellen hat.
Null	Reservierter Parameter, immer 0.
Stützstellen	Datensatz mit Zeiten bzw. x-Koordinaten, stets streng monoton aufsteigend sortiert! An diesen Stellen wird erzwungen, dass die Hüllkurve genau auf einem Eingangs-Sample liegt.
ErgHüllkurve	Resultierende Hüllkurve

Beschreibung:

Algorithmus am Beispiel der oberen Hüllkurve:

Der erste Wert der Eingangsdaten wird gespeichert. Im Intervall wird ab dem Intervall-Start nach der Sekante der größten Steigung gesucht. Dieser 2. Schnittpunkt der Sekante wird gespeichert. Ab diesem Punkt beginnt ein neues Intervall. Der Vorgang wiederholt sich. Der letzte Wert der Eingangsdaten wird auch gespeichert. Außerdem sind im Ergebnis alle Punkte enthalten, die durch die Liste der Stützstellen vorgegeben wird.

- Steigungen werden als gleich angesehen, wenn sie um weniger als 1e-13 voneinander abweichen. Wenn im Intervall mehrmals die gleiche Steigung vorgefunden wird, wird das letzte Auftreten bevorzugt.
- Die Optionen 4 und 5 sind deutlich rechenintensiver als die einfachen Hüllkurven. Diese Optionen sind dann sinnvoll, wenn die obere und untere Hüllkurve miteinander verrechnet werden sollen. Dann sollten beide mit den Optionen 4 und 5 berechnet werden.
- Mit der Funktion Sub() u. ä. kann auch bei EwOption = 1, 2 direkt die Differenz gebildet werden, auch wenn die Stützstellen nicht an den selben Stellen liegen.

Siehe auch:

Envelope1, Sub, XYof

Equal

Verfügbar ab: Professional Edition

Vergleicht zwei Datensätze auf Gleichheit. Der Vergleich erfolgt für jeden Messwert des einen Datensatzes gegen den entsprechenden Messwert des anderen Datensatzes. Gleichheit besteht nur, wenn genau alle gleich sind.

Deklaration:

```
Equal ( Wert1, Wert2 [, Toleranz] [, Berechnung] [, Toleranz2] [, Berechnung2] ) -> IstGleich
```

Parameter:

Wert1	1. Vergleichswert
Wert2	2. Vergleichswert
Toleranz	Beim Vergleich anzuwendende Toleranz. Beide zu vergleichenden Werte dürfen sich maximal um die Toleranz unterscheiden. >= 0. (optional , Standardwert: 0)
Berechnung	Wie wird die Toleranz berechnet? (optional , Standardwert: "absolute")
	"absolute" : Absolute Abweichung
	"relative" : Relative Abweichung
Toleranz2	Toleranz 2. Komponente. Nur nötig, wenn die zu vergleichenden Datensätze XY oder komplex sind. (optional , Standardwert: 0)
Berechnung2	Wie wird die Toleranz der 2. Komponente berechnet? Nur nötig, wenn die zu vergleichenden Datensätze XY oder komplex sind. (optional , Standardwert: "absolute")
	"absolute" : Absolute Abweichung
	"relative" : Relative Abweichung
IstGleich	IstGleich, also 1 bei Gleichheit, 0 bei Abweichung.

Beschreibung:

Die Funktion wird benutzt, um Selbstüberprüfungen kompakt in Sequenzen einzubauen.

Nur die Messwerte werden verglichen. Andere Eigenschaften wie Triggerzeit, Abtastzeit, Einheiten werden nicht verglichen.

Sind die zu vergleichenden Messwerte reelle Zahlen, muss i. Allg. mit einer Toleranz gearbeitet werden.

Eine relative Abweichung von 1e-7 ist nahe an der Auflösung von 4 byte reellen Zahlen, 1e-14 bei 8 byte reellen Zahlen. T

Gleichheit kann nur erreicht werden, wenn die Anzahl von Messwerten in beiden Datensätzen gleich ist.

Die Funktion kann auf Eingangsdaten mit Events oder Segmenten angewendet werden. Gleichheit wird nur erreicht, wenn die Anzahl der Segment und Events auch gleich ist.

Bei absoluter Abweichung darf die Differenz aus Wert1 und Wert2 vom Betrag nicht größer als die Toleranz sein.

Bei relativer Abweichung darf das Verhältnis von Wert1 und Wert2 nur um einen gewissen Anteil voneinander abweichen: Wert1 / (1+Toleranz) <= Wert2 <= Wert1 * (1+Toleranz)

Haben Daten 2 Komponenten (XY oder komplex), so werden beide Komponenten verglichen. Dann müssen auch Toleranz2 und Berechnung2 beachtet werden.

Beispiele:

Erfolgreiche Überprüfung zweier fast gleicher Zahlen, die sich um einen kleinen Prozentsatz unterscheiden

```
A=2
B=sqrt(2)^2 ; A <> B !
eq = equal( A, B, 1e-10, "relative" )
verify( eq )
```

Überprüfung zweier Zeitreihen, die eigentlich gleich sein sollten, aber es aufgrund der Numerik nur näherungsweise sind. Die Differenz zwischen beiden ist stets kleiner als 1e-3.

```
ra1=ramp(0,1,1024) ; test data
ra2 = ifft(fft(ra1))
eq = equal( ra1, ra2, 1e-3, "absolute" )
```

Siehe auch:

Verify

EventAppend

An einen eventierten Datensatz werden ein oder mehrere Events angehängt

Deklaration:

```
EventAppend ( EventDaten, NeueEventDaten, Null )
```

Parameter:

EventDaten	Eventierter Datensatz
NeueEventDaten	Anzuhängende Daten
Null	Reservierter Parameter, immer 0.

Beschreibung:

An einen eventierten Datensatz wird ein Datensatz angehängt.

Wenn der 2. Parameter selbst keine Eventstruktur besitzt, wird der Datensatz als neues Event angehängt.

Wenn der 2. Parameter bereits selbst eventiert ist, werden alle seine Events angehängt.

Der Datentyp der beiden Datensätze muss übereinstimmen (beide normal reell/ XY-Datensätze/ Komplex). Das Datenformat des Ergebnisses bleibt erhalten. Die Eigenschaften x-Offset (Pretrigger), x-Delta (Abtastzeit) und die Triggerzeit des 2. Parameters werden spezifisch für das neue Event übernommen, alle anderen Eigenschaften bleiben erhalten.

Zum Erzeugen eines eventierten Datensatzes verwenden Sie die Funktion EventNew() oder übergeben der EventAppend()-Funktion einen nicht-eventierten Datensatz als ersten Parameter. Dieser wird dann automatisch in einen eventierten Datensatz konvertiert, bevor das zweite Event angehängt wird.

Statt

```
a = EventNew(a, 0)
EventAppend(a, b, 0)
```

kann damit auch kürzer geschrieben werden:

```
EventAppend(a, b, 0)
```

Bei der Strukturierung eines Datensatzes in Events (Ereignisse) können die einzelnen Events verschiedene Längen besitzen, andere Eigenschaften, wie Triggerzeit und Pretrigger (bzw. x-Offset) sind ebenfalls eventspezifisch. Beispielsweise können aufeinander folgende Messungen auf einem physikalischen Messkanal in einem eventierten Datensatz zusammengefasst und so leichter verwaltet werden.

Einige Funktionen in imc FAMOS können eventierte Datensätze nicht direkt verarbeiten, sondern müssen in einer Schleife über alle Events aufgerufen werden.

Beispiele:

Aus drei einzelnen Datensätzen wird ein neuer Datensatz mit 3 Events erzeugt:

```
DataAllDay = EventNew(Data0_8h, 0)
EventAppend(DataAllDay, Data8_16h, 0)
EventAppend(DataAllDay, Data16_24h, 0)
```

Siehe auch:

EventNew, EventGet, EventDel, EventNum?, EventJoin, Join, JoinEx

EventDel

Aus einem eventierten Datensatz wird ein Event entfernt.

Deklaration:

```
EventDel ( EventDaten, EwEventIndex, Null )
```

Parameter:

EventDaten	Eventierter Datensatz
EwEventIndex	Index des zu löschenden Events, im Bereich 1.. Eventanzahl
Null	Reservierter Parameter, immer 0.

Beschreibung:

Das angegebene Event im Datensatz wird gelöscht. Die Gesamtlänge des Datensatzes verringert sich um die Länge des gelöschten Events.

Wenn Sie diese Funktion in einer Schleife über alle Events verwenden, beachten Sie bitte, dass sich nach Ausführung dieser Funktion die Anzahl der Events und das Eventgefüge ändert. Im Allgemeinen sollten darum die Events 'von hinten', d. h. mit dem letzten beginnend, aufgezählt werden (siehe Beispiel).

Beispiele:

In einem eventierten Datensatz werden alle Events gelöscht, deren Länge kleiner als 10 Messwerte ist.

```
count = EventNum?(signal)
index = count
WHILE index >= 1
   eventLeng = EventProp?(signal, index, 3)
   IF eventLeng < 10
      EventDel(signal, index, 0)
   END
   index = index -1
END
```

Siehe auch:

EventGet, EventSet, EventNum?

EventGet

Aus einem eventierten Datensatz wird ein Event herauskopiert

Deklaration:

```
EventGet ( EventDaten, EwEventIndex, Null ) -> EventKopie
```

Parameter:

EventDaten	Eventierter Datensatz
EwEventIndex	Index des gewünschten Events, im Bereich 1.. Eventanzahl
Null	Reservierter Parameter, immer 0.
EventKopie	Kopie des angegebenen Events

Beschreibung:

Die Funktion liefert eine Kopie des angegebenen Events. Alle Eigenschaften, einschließlich des Datenformates, werden übernommen. Der Ergebnisdatensatz besitzt keine Event-Struktur.

Die Funktion wird i. a. verwendet, um eine Aufzählung der einzelnen Events durchzuführen. Einige Funktionen in imc FAMOS können eventierte Datensätze nicht direkt verarbeiten, sondern müssen in einer Schleife über alle Events aufgerufen werden.

In imc FAMOS kann auf ein Event auch direkt über Indizierung mittels [..] zugegriffen werden:

```
singleEvent = EventData[ Index ]
```

Beispiele:

Eine Schleife wird über alle Events in einem Datensatz ausgeführt. Jedes Event wird geglättet und zurück in den Originaldatensatz kopiert.

```
count = EventNum?(signal)
FOR index = 1 TO count
   thisEvent = EventGet(signal, index, 0)
      ;; auch möglich:
      ;; thisEvent = signal[index]
   thisEvent = Smo5(thisEvent)
   EventSet(signal, thisEvent, index, 0)
      ;; auch möglich:
      ;; signal[index] = thisEvent
END
```

Hinweis: Solche Aufgaben können oft auch eleganter mit einer FOREACH EVENT-Schleife gelöst werden.

Siehe auch:

EventSet, EventNum?, EventDel, FOREACH

EventJoin

Die Event-Struktur eines Datensatzes wird aufgelöst.

Deklaration:

```
EventJoin ( EventDaten, Null )
```

Parameter:

EventDaten	Eventierter Datensatz
Null	Reservierter Parameter, immer 0.

Beschreibung:

Alle Events im Datensatz werden fortlaufend aneinander gebunden und die Strukturierung aufgelöst. Die eventspezifischen Kennwerte des ersten Events werden für den gesamten Datensatz übernommen.

ACHTUNG: Die eventspezifischen Eigenschaften der anderen Events gehen verloren, also insbesondere Triggerzeit, Pretrigger, Abtastzeit.

Die Gesamtlänge des Datensatzes bleibt unverändert.

Falls der erste Parameter nicht eventiert ist, bleibt der Datensatz unverändert. Es wird dann eine Warnung generiert.

Die Funktion wird i. a. verwendet, um eine anschließende Nachverarbeitung des gesamten Datensatzes einzuleiten. Einige Funktionen in imc FAMOS können eventierte Datensätze nicht direkt verarbeiten, sondern müssen entweder in einer Schleife über alle Events aufgerufen oder aber mit dieser Funktion vorbereitet werden.

Beispiele:

Das absolute Maximum über alle Events eines eventierten Datensatzes wird bestimmt:

```
EventJoin(signal, 0)
Maximum = Max(signal)
```

Siehe auch:

EventSet, EventGet, JoinEx

EventNew

Aus einem unstrukturierten Datensatz wird ein Datensatz mit Event-Eigenschaft erzeugt.

Deklaration:

```
EventNew ( ErstesEvent, Null ) -> EventDaten
```

Parameter:

ErstesEvent	Datensatz, der das erste Event bilden soll
Null	Reservierter Parameter, immer 0.
EventDaten	Eventierter Datensatz

Beschreibung:

Es wird eine Kopie der Quelldaten mit zusätzlicher Eventeigenschaft erzeugt. Mit nachfolgenden EventAppend()-Aufrufen können weitere Events zugefügt und so ein MultiEvent-Datensatz erzeugt werden.

Bei der Strukturierung eines Datensatzes in Events (Ereignisse) können die einzelnen Events verschiedene Längen besitzen, andere Eigenschaften, wie Triggerzeit und Pretrigger (bzw. x-Offset) sind ebenfalls eventspezifisch. Beispielsweise können aufeinander folgende Messungen auf einem physikalischen Messkanal in einem eventierten Datensatz zusammengefasst und so leichter verwaltet werden.

Beispiele:

Aus drei einzelnen Datensätzen wird ein neuer Datensatz mit 3 Events erzeugt:

```
DataAllDay = EventNew(Data0_8h, 0)
EventAppend(DataAllDay, Data8_16h, 0)
EventAppend(DataAllDay, Data16_24h, 0)
```

Siehe auch:

EventAppend, JoinEx

EventNum?

Abfrage der Zahl der Events (Ereignisse) in einem Datensatz.

Deklaration:

```
EventNum? ( EventDaten ) -> EwEventZahl
```

Parameter:

EventDaten	Eventierter Datensatz
EwEventZahl	EventZahl
	>=0 : Anzahl der enthaltenen Events
	-1 : Der Parameter ist kein eventierter Datensatz.

Beschreibung:

Die Anzahl der vorhandenen Events in einem eventierten Datensatz wird ermittelt. Falls es sich nicht um einen Datensatz mit Eventstrukturierung handelt, wird eine -1 zurückgegeben.

Die Funktion wird i. a. verwendet, um eine anschließende Aufzählung der einzelnen Events einzuleiten. Einige Funktionen in imc FAMOS können eventierte Datensätze nicht direkt verarbeiten, sondern müssen in einer Schleife über alle Events aufgerufen werden.

Beispiele:

Eine Schleife wird über alle Events in einem Datensatz ausgeführt. Jedes Event wird geglättet und zurück in den Originaldatensatz kopiert.

```
count = EventNum?(signal)
FOR index = 1 TO count
   thisEvent = signal[index]
   thisEvent = Smo5(thisEvent)
   signal[index] = thisEvent
END
```

Hinweis: Solche Aufgaben können oft auch eleganter mit einer FOREACH EVENT-Schleife gelöst werden.

Siehe auch:

EventSet, EventGet, Join

EventProp

Setzen von eventspezifischen Eigenschaften

Deklaration:

```
EventProp ( EventDaten, EwEventIndex, EwAuftrag, EwNeuerKennwert )
```

Parameter:

EventDaten	Zu ändernder eventierter Datensatz
EwEventIndex	Index des zu ändernden Events, im Bereich 1.. Eventanzahl
EwAuftrag	Auswahl des zu ändernden Kennwertes.
	0 : Triggerzeit des Events (im internen Zeitformat).
	1 : Abtastzeit bzw. x-Delta
	2 : Pretriggerzeit bzw. x-Offset
	5 : z-Offset
EwNeuerKennwert	Neuer Kennwert

Beschreibung:

Für ein angegebenes Event in einem Datensatz werden eventspezifische Kennwerte geändert.

Bei der Strukturierung eines Datensatzes in Events (Ereignisse) können die einzelnen Events verschiedene Längen besitzen, andere Eigenschaften, wie Triggerzeit und Pretrigger (bzw. x-Offset) sind ebenfalls eventspezifisch. Beispielsweise können aufeinander folgende Messungen auf einem physikalischen Messkanal in einem eventierten Datensatz zusammengefasst und so leichter verwaltet werden.

Beispiele:

In einem eventierten Datensatz werden alle Events mit der gleichen Triggerzeit versehen:

```
time = TimeJoin(11, 2, 1997, 15, 0, 0)
count = EventNum?(signal)
FOR index = 1 TO count
   EventProp(signal, index, 0, time)
END
```

Siehe auch:

EventSet, EventNum?, EventProp?

EventProp?

Abfrage von eventspezifischen Eigenschaften

Deklaration:

```
EventProp? ( EventDaten, EwEventIndex, EwAuftrag ) -> EwKennwert
```

Parameter:

EventDaten	Abzufragender eventierter Datensatz
EwEventIndex	Index des gewünschten Events, im Bereich 1.. Eventanzahl
EwAuftrag	Auswahl des abzufragenden Kennwertes
	0 : Triggerzeit des Events (im internen Zeitformat).
	1 : Abtastzeit bzw. x-Delta
	2 : Pretriggerzeit bzw. x-Offset
	3 : Länge (Zahl der Datenpunkte)
	4 : Index des ersten Punktes des Events in Bezug zum gesamten Datensatz. Für das erste Event ist dies eine 1, für alle weiteren Events die Summe der Längen der vorhergehenden Events, addiert mit 1.
	5 : z-Offset
EwKennwert	Abgefragter Kennwert

Beschreibung:

Zu einem angegebenen Event in einem Datensatz werden eventspezifische Kennwerte abgefragt.

Bei der Strukturierung eines Datensatzes in Events (Ereignisse) können die einzelnen Events verschiedene Längen besitzen, andere Eigenschaften, wie Triggerzeit und Pretrigger (bzw. x-Offset) sind ebenfalls eventspezifisch. Beispielsweise können aufeinander folgende Messungen auf einem physikalischen Messkanal in einem eventierten Datensatz zusammengefasst und so leichter verwaltet werden.

Beispiele:

In einem eventierten Datensatz werden alle Events gelöscht, deren Länge kleiner als 10 Messwerte ist.

```
count = EventNum?(signal)
index = count
WHILE index >= 1
   EventLeng = EventProp?(signal, index, 3)
   IF EventLeng < 10
      EventDel(signal, index, 0)
   END
   index = index - 1
END
```

Siehe auch:

EventProp, EventSet, EventNum?

EventSet

Ein ausgewähltes Event in einem Datensatz wird ersetzt.

Deklaration:

```
EventSet ( EventDaten, NeuesEvent, EwEventIndex, Null )
```

Parameter:

EventDaten	Eventierter Datensatz
NeuesEvent	Neue Daten für das angegebene Event.
EwEventIndex	Index des zu ändernden Events, im Bereich 1.. Eventanzahl
Null	Reservierter Parameter, immer 0.

Beschreibung:

In einem eventierten Datensatz wird ein gewähltes Event ersetzt. Der ersetzende Datensatz darf selbst nicht eventiert sein.

Die Datentypen der ersten beiden Parameter müssen gleich sein (beide reelle, komplexe oder XY-Datensätze). Das Datenformat des ersten Parameters wird beibehalten.

Die Funktion wird i. a. verwendet, um nach der Bearbeitung eines Events dieses wieder in den Originaldatensatz zurückzuschreiben. Einige Funktionen in imc FAMOS können eventierte Datensätze nicht direkt verarbeiten, sondern müssen in einer Schleife über alle Events aufgerufen werden.

In imc FAMOS kann auf ein Event auch direkt über Indizierung mittels [..] zugegriffen werden:

```
EventDaten[ Index ] = NeueDaten
```

Beispiele:

Eine Schleife wird über alle Events in einem Datensatz ausgeführt. Jedes Event wird geglättet und zurück in den Originaldatensatz kopiert.

```
count = EventNum?(signal)
FOR index = 1 TO count
   thisEvent = EventGet(signal, index, 0)
      ;; auch möglich:
      ;; thisEvent = signal[index]
   thisEvent = Smo5(thisEvent)
   EventSet(signal, thisEvent, index, 0)
      ;; auch möglich:
      ;; signal[index] = thisEvent
END
```

Hinweis: Solche Aufgaben können oft auch eleganter mit einer FOREACH EVENT-Schleife gelöst werden.

Siehe auch:

EventGet, EventNum?, EventDel, FOREACH

Execute

Ein ausführbares Programm wird gestartet und/oder eine Aktion mit einer Dokument-Datei ausgeführt.

Deklaration:

```
Execute ( TxDateiName, TxParameter, TxAktion, EwAnzeigeOption, EwWarteOption ) -> EwExitCode
```

Parameter:

TxDateiName	Name der ausführbaren Datei (*.exe, *.bat, *.com) bzw. Name der zu bearbeitenden Dokument-Datei. Eine Dokument-Datei muß über die Windows-Systemsteuerung mit einer Anwendung verknüpft sein.
TxParameter	Zu verwendende Parameter beim Starten der Anwendung, durch Leerzeichen getrennt. Für Dokument-Dateien ein leerer Text.
TxAktion	Gibt die Art der auszuführenden Aktion an. Die verfügbaren Aktionsverben hängen vom Typ der angegebenen Datei ab. Zum Ausführen einer Anwendung ein leerer Text oder "Open".
EwAnzeigeOption	Anzeige des Anwendungsfensters
	0 : Startet bzw. zeigt die Anwendung im normalen Fenster. Die Anwendung wird aktiviert.
	1 : Startet bzw. zeigt die Anwendung im normalen Fenster. Die Anwendung wird nicht aktiviert.
	2 : Startet bzw. zeigt die Anwendung in minimiertem Zustand (als Ikone).
	3 : Startet bzw. zeigt die Anwendung in maximiertem Zustand..
EwWarteOption	Warten auf Ende
	0 : Die Anwendung wird gestartet. Die Funktion kehrt sofort zurück.
	-1 : Die Anwendung wird gestartet. Die Funktion kehrt erst zurück, wenn die Anwendung beendet wurde. Sinnvoll z.B. für Aufrufe des Kommando-Interpreters und andere Anwendungen ohne explizierte Benutzerschnittstelle.
	>0 : Maximale Wartezeit (in Sekunden). Die Funktion kehrt zurück, wenn die gestartete Anwendung beendet ist ODER die Wartezeit abgelaufen ist.
EwExitCode	Status
	>=0 : Funktion ist erfolgreich ausgeführt und beendet worden. Der konkrete Wert ist anwendungsspezifisch und entspricht dem Code, der von der Anwendung an das Betriebssystem zurückgeliefert wird. Dieser kann z.B. den Erfolg der ausgeführten Aktion anzeigen.
	-2 : Anwendung wurde erfolgreich gestartet, ist aber nicht innerhalb der angegebenen Wartezeit beendet worden.
	-3 : Die gewünschte Aktion wurde durch eine bereits geöffnete Instanz der Anwendung ausgeführt und somit keine neue Anwendung gestartet.

Beschreibung:

Bearbeiten von Dokument-Dateien:

Wenn als erster Parameter eine Dokument-Datei angegeben wird, muß bekannt sein, mit welcher Anwendung diese Datei bearbeitet werden soll. Eine solche Verknüpfung ist in den Windows-Systemeinstellungen (Systemsteuerung / Ordneroptionen / Dateitypen) festgelegt und wird über die Dateinamens-Erweiterung identifiziert.

Neben Windows-Standardvorgaben werden solche Verknüpfungen normalerweise automatisch bei der Installation eines Anwendungsprogramms erzeugt, können aber auch manuell vom Anwender geändert werden. Hier werden sogenannte Aktions-Verben definiert, die verschiedene Bearbeitungsaktionen beschreiben. Eine solche definierte Aktion muß dann als 3. Parameter [TxAktion] übergegeben werden.

Die folgenden Aktions-Verben sind für Dokumente üblich:

"Open"	Startet die Datei (ausführbare Datei oder Dokument)
"Edit"	Öffnet einen Editor und lädt das Dokument.
"Explore"	Öffnet den Windows-Explorer und zeigt das angegebene Verzeichnis.
"Find"	Startet eine Suche im angegebenen Verzeichnis.
"Print"	Druckt die Dokument-Datei.
"Properties"	Öffnet den Eigenschaften-Dialog für die angegebene Datei.

Welche Verben tatächlich verfügbar sind, hängt von den Windows-Systemeinstellungen ab. Eine Anwendung kann selbst festlegen, welche Aktionen unterstützt werden, und zusätzliche Verben definieren (z.B. "play" für Video- und Sound-Dateien).

Anmerkungen

- Für ausführbare Dateien wird, wenn kein voller Pfadname angegeben ist, in den Verzeichnissen gesucht, die durch die Umgebungsvariable [PATH] aufgelistet sind. Für Dokument-Dateien sollte dagegen stets der komplette Pfadname angegeben werden.
- Im Fehlerfall (z.B. ungültiger Pfadname oder unbekannte Dateinamens-Erweiterung) wird eine Fehlermeldung angezeigt und die Sequenzabarbeitung abgebrochen.
- Während die Funktion auf das Ende der gestarteten Anwendung wartet ([WarteOption] = -1 oder >0) ist FAMOS nicht bedienbar (Ausnahme: Kurvenfenster). Allerdings kann eine laufende Sequenz auf die übliche Weise ("Sequenz"-Symbol mit Kontextmenü im Infobereich der Windows-Taskleiste oder [Strg+Break]) abgebrochen werden. Während der Wartezeit ist FAMOS per DDE teilweise erreichbar: Daten können empfangen, aber keine Anweisungen ausgeführt werden.
- Falls ein Parameter in [TxParameter] Leerzeichen enthält, muß dieser in Anführungszeichen eingeschlossen werden. Siehe Beispiel.
- Die Abfrage des Rückgabewertes ist in FAMOS optional.
- Der Parameter [WarteOption] wird an die gestartete Anwendung übergeben. Ob dieser tatsächlich beachtet wird, ist jedoch anwendungsspezifisch.
- Wenn eine Instanz der gewünschten Anwendung bereits geöffnet ist, ist das resultierende Verhalten ebenfalls anwendungsspezifisch. Entweder wird eine neue zusätzliche Programminstanz gestartet oder die bereits vorhandene wiederverwendet. In letzteren Fall wird unabhängig vom Parameter [WarteOption] nicht auf das Ende gewartet, die Funktion kehrt sofort zurück (erkennbar am Rückgabewert [-3]).

Beispiele:

Starten des Editors [Notepad] und Laden der angegebenen Datei. Funktion kehrt sofort zurück.

```
ret = Execute("notepad.exe", "c:\tmp\MyTest.txt", "", 0, 0)
```

Parameter mit Leerzeichen müssen in Anführungszeichen eingeschlossen werden:

```
ret = Execute("notepad.exe", """c:\tmp\My Text.txt""", "", 0, 0)
```

Das gleiche Resultat erhält man durch den folgenden Aufruf (Annahme: Erweiterung "txt" ist mit [Notepad] verknüpft):

```
ret = Execute("c:\tmp\My Text.txt", "", "open", 0, 0)
```

Starten von MS-EXCEL und Laden der angebenen Tabelle. Funktion kehrt zurück, wenn EXCEL wieder geschlossen wurde, spätestens aber nach 60s. Beide Aufrufe sind gleichwertig.

```
ret = Execute("excel.exe", "c:\tmp\1.xls", "", 0, 60)
```

```
ret = Execute("c:\tmp\1.xls", "", "open", 0, 60)
```

Drucken der angegebenen MS-EXCEL-Datei.

```
ret = Execute("c:\tmp\1.xls", "", "print", 2, 0)
```

Aufruf des Pack-Programms "7-Zip" zum Entpacken einer Datei:

```
fullname$ = "c:\files\archive.zip" ; input file
dir$ = "c:\files\unpacked" ; output folder

para$ = "e """ + fullname$ +""" -o""" + dir$ + """"
; Bitte beachten Sie die zusätzlichen Gänsefüßchen um die Dateinamen. Diese sind notwendig für den Fall, dass die Dateinamen Leerzeichen enthalten.
ret$ = Execute("c:\Program Files\7-Zip\7z", para$, "", 0, -1)
```

Starten von MS-EXCEL und Laden der angebenen Tabelle. Funktion kehrt zurück, wenn EXCEL wieder geschlossen wurde, spätestens aber nach 60s. Beide Aufrufe sind gleichwertig.

```
ret = Execute("excel.exe", "c:\tmp\1.xls", "", 0, 60)
ret = Execute("c:\tmp\1.xls", "", "open", 0, 60)
```

Drucken der angegebenen MS-EXCEL-Datei.

```
ret = Execute("c:\tmp\1.xls", "", "print", 2, 0)
```

Zum Ausführen eines Betriebsystems-Kommandos kann der Kommandoprozessor mit der Option "/C", gefolgt vom gewünschten Befehl, gestartet werden. Im Beispiel wird ein Verzeichnis erzeugt. Das Konsolenfenster wird im minimierten Zustand angezeigt. Die Funktion kehrt erst nach dem Ende des Kommando-Interpreters, also nachdem das Verzeichnis tatsächlich erzeugt wurde, zurück.

```
ret = Execute("cmd.exe", "/C MkDir z:\NewFolder", "", 2, -1)
```

Aufruf des "Eigenschaften"-Dialoges zu der angegebenen Datei:

```
ret = Execute("c:\tmp\data.raw", "", "properties", 0, 0)
```

Anmerkung: Hier wird der Standard-Windows-Explorer verwendet, um den Eigenschaften-Dialog anzuzeigen. Da dieser stets aktiv ist, wird hier eine (-3) zurückgeliefert, der letzte Parameter [WarteOption] wird ignoriert.

Siehe auch:

APPLICATION

EXITSEQUENCE

Sequenzabarbeitung beenden
Alternativer Name: **EXITSEQUENZ**

Deklaration:

```
EXITSEQUENCE EwOption
```

Parameter:

EwOption	Option
	0 : Die Sequenzabarbeitung wird komplett beendet. Standard bei klassischen Sequenzen, nicht erlaubt innerhalb von Sequenzfunktionen.
	1 : Sprung zum Ende der aktuellen Sequenz. Falls es sich um eine Untersequenz handelt, wird die Abarbeitung in der aufrufenden Sequenz fortgesetzt. Standard bei Sequenzfunktionen.
	2 : Sprung zum Ende der aktuellen Sequenz. Falls es sich um eine zyklische Untersequenz handelt (Aufruf mit Parametern mit Jokerzeichen), wird mit dem nächsten Zyklus fortgesetzt, ansonsten wird mit der Abarbeitung der aufrufenden Sequenz fortgefahren. Nicht erlaubt innerhalb von Sequenzfunktionen.

Beschreibung:

Die Sequenzabarbeitung wird beendet. Je nach Option wird nur die laufende Untersequenz/Sequenzfunktion beendet (Rücksprung in die aufrufende Sequenz) oder die gesamte Sequenzabarbeitung terminiert.

Der Parameter ist optional. Wenn nicht angegeben, wird im Falle einer klassischen Sequenz eine 0 (Komplett-Abbruch), im Falle einer Sequenzfunktion eine 1 (Sprung zum Sequenzende) angenommen.

Beispiele:

Abbrechen einer Sequenz nach Benutzereingabe:

```
Stop = BoxMessage("Auswertung", "Auswertung abbrechen ?", "?4")
IF Stop
   EXITSEQUENZ
END
```

Der Aufruf der folgenden Sequenz mit

```
SEQUENCE Auswertung *.dat
```

führt zum Auswerten aller Dateien "*.dat" aus dem aktuellen Verzeichnis, die erfolgreich im imc/FAMOS-Format geladen werden konnten.

```
; Sequenz "Auswertung.seq"
fh = FileOpenDSF("PA1", 0)
IF fh < 1
   EXITSEQUENCE 2
END
...
; Umfangreiche Auswertung folgt
...
```

Wenn dagegen der Parameter von EXITSEQUENCE in eine 1 geändert wird, stoppt die Abarbeitung, sobald das Öffnen einer Datei fehlschlägt (z.B. weil die Datei nicht im imc/FAMOS-Format vorliegt).

Siehe auch:

PAUSE

exp

Exponentialfunktion (Exponent zur Basis e)

Deklaration:

```
exp ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Parameter. Erlaubte Typen: [ND],[XY].
Ergebnis	Ergebnis aus der Bildung der e-Funktion.

Beschreibung:

Die Exponentialfunktion erhebt den Parameter in den Exponenten zur Basis e (Eulersche Zahl = 2.718...)

Die Exponentialfunktion kann auch mit Hilfe des Operators ^ (hoch) ausgedrückt werden. Folgende Formeln erfüllen dieselbe Funktion:

```
y = exp(x)
y = e ^ x
```

Anmerkungen

- Die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Die Exponentialfunktion steigt bei positivem Argument sehr stark an. Bei Argumenten von über etwa 46 ist schon der zulässige Zahlenbereich überschritten.
- Ein Aufruf ln(exp(x)) stellt eine Identität dar, solange nicht im Zwischenergebnis der zulässige Wertebereich überschritten wird.
- Der Parameter der Exponentialfunktion sollte keine Einheit haben.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Die Werte eines Messgebers werden entsprechend einer exponentiellen Kennlinie korrigiert:

```
data_corrected = 2 * exp(0.1 * data)
```

Siehe auch:

ln, ^(hoch)

ExpoRMS

Gleitender Effektivwert mit exponentieller Mittelung

Deklaration:

```
ExpoRMS ( Daten, EwZeitkonstante, EwReduktion ) -> GleitEff
```

Parameter:

Daten	Zu mittelnder Datensatz. [ND]
EwZeitkonstante	Zeitkonstante/Bewertung des Filters
	>=0 : Zeitkonstante des Filters
	-1 : FAST-Bewertung
	-2 : SLOW-Bewertung
	-3 : IMPULSE-Bewertung
	-4 : PEAK-Bewertung
EwReduktion	Breite des Reduktionsintervalls (in x-Einheiten).
GleitEff	Gleitender Effektivwert

Beschreibung:

Es wird der gleitende Effektivwert mit exponentiell gewichteter Mittelung und anschließender Nachabtastung berechnet.

Die hier benutzte exponentielle Mittelung kann als Filterung mit einem Tiefpass 1. Ordnung verstanden werden. Die Zeitkonstante kann größer oder kleiner als die Abtastzeit sein.

Der gleitende Effektivwert wird hier auf folgende Weise berechnet: Das Signal wird quadriert, dann exponentiell gemittelt, dann wird wieder die Wurzel gezogen. Quadrierung, Mittelung und anschließende Radizierung sind die Merkmale einer Effektivwert-Bildung. Beim gewöhnlichen Effektivwert wird ein gleichmäßig gewichtetes Mittel aller Quadrate gebildet, während hier beim gleitenden Effektivwert eine zeitliche Bewertung durchgeführt wird.

Im Gegensatz zur Funktion MvRMS() wird hier also nicht über eine feste Fensterbreite gleichmäßig gemittelt. 'Ältere' Messwerte werden hier geringer gewichtet.

Nach der Filterung wird das Ergebnis noch über die Intervallbreite [EwReduktion] nachabgetastet. Wenn [EwReduktion] kleiner oder gleich der Abtastzeit ist, findet keine Datenreduktion statt.

Die Funktion arbeitet analog zur der in den Funktionen KBT() (KB-Bewertung), OctA() (Terzanalyse), und ABCRating() enthaltenen Effektivwertbildung.

Die vordefinierten Zeitbewertungen entsprechen der Norm IEC 651 und haben folgende Bedeutung:

FAST	Zeitkonstante = 0.125 s.
SLOW	Zeitkonstante = 1 s.
IMPULSE	Bei ansteigender Amplitude beträgt die Zeitkonstante 35ms, bei abfallender Amplitude 1.5s. Damit werden impulsförmige Signale schnell erfasst, die Anzeige klingt langsam ab.
PEAK	Extreme Anzeige für ganz kurze Impulse, wobei garantiert der Spitzenwert gezeigt wird. Bei ansteigender Amplitude Zeitkonstante Null (ist im Computer exakt machbar, in Analogschaltung nur näherungsweise). Bei abfallender Amplitude 3s.

Beispiele:

```
resultRMS = ExpoRMS(timeData, 0.125, 0.05)
```

Es wird mit der Zeitkonstante 0.125 s (FAST-Bewertung) der gleitende Effektivwert bestimmt. Vom Ergebnis wird alle 0.05s ein Wert verwendet.

```
resultRMS = ExpoRMS(timeData, -2, 0)
```

Effektivwertbildung mit Zeitkonstante 1s (SLOW-Bewertung). Eine Datenreduktion findet nicht statt.

Siehe auch:

MvRMS, MvMean, KBT, RMS, ABCRating

FAMOS

Einstellen des imc/FAMOS-Formats zum anschließenden Laden von Dateien mit dem Befehl LOAD

Deklaration:

```
FAMOS
```

Beschreibung:

Dieser Befehl ist veraltet, statt der Befehlskombination FAMOS/LOAD können Sie auch die leistungsfähigeren und bequemer anzuwendenden Funktionen FileLoad() oder FileOpenDSF() verwenden.

Das Dateiformat für den Befehl LOAD wird auf das imc/FAMOS-Format gestellt.

Multithreading: Der Befehl wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
FAMOS
LOAD DATEN
```

Das Dateiformat zum Laden von Dateien wird auf das imc/FAMOS-Format gestellt. Anschließend die Datei DATEN.DAT geladen und unter dem Namen DATEN in die Variablenliste eingetragen.

Siehe auch:

LOAD, FileLoad, FileOpenDSF

FASLOAD

Datei in einem nutzerdefinierten Format laden (Datei-Assistent oder Import/Export- Erweiterungsbibliothek)
Alternativer Name: **FASLADEN**

Deklaration:

```
FASLOAD TxDateiname TxFormat Variablenname
```

Parameter:

TxDateiname	Name der zu ladenden Datei. Als Textkonstante in "..." oder als Textvariable zu übergeben.
TxFormat	Spezifikation des Dateiformats.
Variablenname	Name, unter dem die Variable in die Variablenliste aufgenommen wird. Dieser Name wird nur benutzt, wenn die Datei genau 1 Datenobjekt enthält. Ansonsten werden die bei der Formatdefinition angegebenen Namen benutzt. Als Textkonstante in "..." oder Textvariable zu übergeben.

Beschreibung:

Dieser Befehl ist veraltet, statt dessen können Sie auch die leistungsfähigere und bequemer anzuwendende Funktion FileLoad() verwenden.

Für die Spezifikation des Formats im Parameter [TxFormat] gibt es 2 Varianten:

Importfilter (Datei-Assistent)

Das Format ist durch eine Import-Beschreibungsdatei (Importfilter) festgelegt, die mit dem imc-Datei- Assistenten erstellt wurde. Eine solche Importfilter-Datei hat üblicherweise die Erweiterung ".fas". Der Name dieser Datei wird dann als Parameter übergeben. Falls kein kompletter Pfadname angegeben ist, wird im voreingestellten Definitionsverzeichnis ("Extra" / "Optionen" / "Verzeichnisse") gesucht.

Import/Export-Erweiterungsbibliothek

Die notwendige Funktionalität wird durch eine Import/Export-Erweiterungsbibliothek zur Verfügung gestellt. Solche Bibliotheken ("Dynamic link library", DLL) werden von imc oder unseren Partnern angeboten und integrieren sich nahtlos in die FAMOS-Bedienoberfläche und die Funktionsbibliothek. Damit ist eine flexible Anpassung von FAMOS an zusätzlich zu unterstützende Dateiformate möglich. Ein Beispiel ist die DLL zum Import und Export des MATLAB-Dateiformates, die im Standard-Lieferumfang von imc FAMOS enthalten ist.

Die Syntax des Parameters [TxFormat] ist dann wie folgt:

```
"#DLLName|FormatName|Parameter"
```

[DLLName] gibt den Dateinamen der Erweiterungsbibliothek an, die Erweiterung ".DLL" kann weggelassen werden.

[FormatName] gibt den Namen des gewünschten Formats an (eine Erweiterungsbibliothek kann durchaus mehrere Formate unterstützen). Kann weggelassen werden, wenn die DLL nur ein einziges Format zur Verfügung stellt und keine weiteren Parameter benötigt.

[Parameter] dient zur optionalen Übergabe weiterer Parameter an die Bibliothek. Ob Parameter benötigt werden sowie deren Syntax hängt von der jeweiligen Erweiterungsbibliothek ab. Oft nicht benötigt, kann dann weggelassen werden.

Tipp:Verwenden Sie den Funktions-Assistent für die Parametrierung der Funktion. Sie können hier bequem in einer Liste aller installierten Dateifilter auswählen.

Beispiele

```
FASLOAD "1.txt" "#MyFormat.DLL"
FASLOAD "1.txt" "#MyFormat.DLL|Txt"
FASLOAD "1.asc" "#MyFormat.DLL|Asc|Header=yes"
```

Eine Übersicht über die im FAMOS Lieferumfang enthaltenen Importfilter und ihre Parametrierung finden Sie in der Online-Hilfe, "FAMOS Bedienerhandbuch"=>"Import/Export Filter"

- Falls der Dateiname keinen vollen Verzeichnispfad enthält, wird das momentan eingestellte Ladeverzeichnis benutzt. Dieser Pfad steht nach dem Aufstart von imc FAMOS auf dem im Dialog "Optionen"/ "Verzeichnisse" eingestellten Standardpfad und kann z.B. mit dem Befehl LDIR oder der Funktion SetOption() umgestellt werden.
- Sie können im Dateinamen Jokerzeichen (Wildcards) angeben, um eine Reihe von Dateien zu laden. Das Jokerzeichen '?' steht dabei für genau ein beliebiges Zeichen, das Jokerzeichen '*' für eine unbestimmte Anzahl von beliebigen Zeichen.
- Der Variablenname ist optional. Er wird nur verwendet, wenn die zu lesende Datei genau ein Datenobjekt enthält. Ansonsten werden die Variablennamen verwendet, die bei der Definition mit dem Datei-Assistenten festgelegt worden sind.
- Zum Laden von nutzerdefinierten Dateien können Sie auch die Funktion FileOpenFAS() verwenden. Diese ist besonders bei der Verarbeitung von Multikanal-Dateien überlegen.

Beispiele:

```
FASLOAD "Daten.dat" "Oszi2311.fas" "Kanal1"
```

Die Datei DATEN.DAT wird unter Verwendung der angegebenen Formatbeschreibungsdatei gelesen und als Variable mit dem Namen "kanal1" in imc FAMOS angelegt.

```
FASLOAD "Daten.mat" "#MatLabImportExport.DLL"
```

Die Datei DATEN.MAT im MatLab® -Format wird geladen.

```
TxFormatFile = "myformat.fas"
FASLOAD "c:\imc\*.*" TxFormatFile
```

Im Verzeichnis c:\imc werden alle Dateien geladen.

```
FASLOAD "a??.*" TxFormatFile
```

Alle Dateien im aktuellen Verzeichnis werden geladen, deren Name aus 3 Zeichen, beginnend mit einem 'a', besteht. Die Dateinamens-Erweiterung ist beliebig.

```
FASLOAD "*a1*.dat" TxFormatFile
```

Alle Dateien mit der Erweiterung 'dat', in deren Namen die Zeichenfolge '1a' (auch zu Beginn oder am Ende) vorkommt, werden geladen.

Siehe auch:

FileLoad, FileOpenFAS, LDIR

FFT

Spektrum nach FFT-Algorithmus

Deklaration:

```
FFT ( Daten [, EwFenster] [, EwModus] ) -> Spektrum
```

Parameter:

Daten	Datensatz, von dem das Spektrum berechnet werden soll [ND].
EwFenster	Fenster-Funktion (optional , Standardwert: -1)
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman/Harris
	5 : Flat-Top
	-1 : Die globale Voreinstellung (Dialog 'Optionen'/'Funktionen', SetOption(), FFTOPTION) wird verwendet. Kompatibel zu älteren FAMOS-Versionen (< V2022).
EwModus	Verfahren (optional , Standardwert: -1)
	0 : Radix 2: Die Länge des Datensatzes wird vor der Berechnung auf die nächstkleinere 2er-Potenz verringert (abgeschnitten).
	1 : Radix 2: Die Länge des Datensatzes wird vor der Berechnung auf die nächstgrößere 2er-Potenz erhöht (mit Nullen aufgefüllt).
	2 : Radix 2: Der Datensatz wird vor der Berechnung so nachabgetastet, dass die neue Datensatzlänge eine Zweierpotenz wird. Entspricht einer vorherigen Anwendung der Funktion Red2().
	3 : Mixed-Radix: Die Länge des Datensatzes muss ein Produkt von Potenzen der Zahlen 2, 3 und 5 sein. Andere Datensatzlängen können nicht verarbeitet werden.
	-1 : Die globale Voreinstellung (Dialog 'Optionen'/'Funktionen', SetOption(), FFTOPTION) wird verwendet. Kompatibel zu älteren FAMOS-Versionen (< V2022).
Spektrum	Diskretes Spektrum des Datensatzes nach FFT-Algorithmus [BP].

Beschreibung:

Die FFT-Funktion berechnet das diskrete Spektrum eines Datensatzes nach dem sehr schnellen FFT-Algorithmus. In der vorliegenden Implementierung wird der klassische Algorithmus nach Cooley und Tukey in der Radix-2- oder Mixed-Radix-Form benutzt. Die FFT (Fast Fourier-Transformation) ist ein Spezialfall der allgemeinen diskreten Fourier-Transformation (DFT). In diesem Spezialfall wird ausgenutzt, dass die Länge des vorliegenden Datensatzes eine Zweierpotenz (Radix 2) oder ein Produkt von Potenzen der Zahlen 2, 3 und 5 (Mixed-Radix) ist.

Das von der FFT erzeugte Spektrum gibt die Größen und Phasen der im Signal enthaltenen Harmonischen an. Die Werte des Spektrums sind nach ihrer Frequenz geordnet. Der allererste Wert gehört dabei zur Frequenz 0, er gibt den Mittelwert des Signals wieder.

Beachten Sie, dass die Größen der Harmonischen nicht direkt als Amplituden interpretierbar sind. Wenn Sie auf eine Interpretierbarkeit der Amplituden Wert legen, benutzen Sie die Funktion Spec().

Es ist wichtig zu beachten, dass kein kontinuierliches sondern nur ein diskretes Spektrum berechnet wird. Das bedeutet, dass das Spektrum nur für einige diskrete Frequenzen berechnet wird. Das sind alle die Frequenzen, bei denen eine ganzzahlige Anzahl von Perioden innerhalb des Berechnungsintervalls liegen. Wird zum Beispiel die FFT auf einen Datensatz der Länge 0.5s angewendet, wird das Spektrum an den Frequenzen 0, 2Hz, 4Hz, 6Hz, 8Hz ... bestimmt.

Ein wichtiges Prinzip in diesem Zusammenhang ist, dass ein diskretes Spektrum das Spektrum eines periodischen Signals ist. Bei der Berechnung des Spektrums eines Signals wird das Signal so interpretiert, als wäre es viele Male nach vorn und hinten aneinandergereiht. Stellt das Signal einen einzigen Impuls dar, so wird streng genommen nicht das Spektrum dieses einmaligen Impulses, sondern das einer Kette von Impulsen berechnet. Dieses periodische Aneinanderreihen macht sich besonders bemerkbar, wenn eine nichtganze Zahl von Perioden eines Signals vorliegen.

Ein Datensatz enthält zum Beispiel 3.5 Perioden eines sinusförmigen Signals. Als Spektrum wird eigentlich eine einzige Harmonische erwartet. Nun aber stelle man sich das Signal periodisch fortgesetzt vor. Das ergibt nun insgesamt keinen sinusförmigen Verlauf, sondern nur eine Reihe sinusförmiger Stücke, die nicht richtig aneinandergereiht zu sein scheinen. Es wird verständlich, dass das Spektrum nicht eine einzige, sondern viele Harmonische enthält, die in der Nähe der wirklichen Frequenz besonders ausgeprägt sind. Diesen offensichtlichen Nachteil kann man auf zwei Arten beheben. Die erste Möglichkeit ist, vor der Berechnung des Spektrums einen geeigneten Ausschnitt auszuwählen, der eine ganze Anzahl von Perioden des Signals enthält. Das wird man meistens ohnehin bei periodischen Signalen machen. Die andere Möglichkeit ist die Anwendung einer Fensterfunktion. Es können auch beide Möglichkeiten kombiniert werden.

Eine Fensterfunktion gewichtet die Werte eines Datensatzes unterschiedlich stark. Die Randwerte werden besonders schwach, die mittleren Werte besonders stark bewertet.

Sechs verschiedene Fensterfunktionen stehen zur Auswahl:

- Rechteck
- Hanning
- Hamming
- Blackman
- Blackman-Harris.
- Flat Top

Das Rechteckfenster bewertet alle Werte gleich. Es ist die Standardeinstellung, bei der keine Extra-Berechnung erforderlich ist. Das Flat Top - Fenster am Ende der Liste ist das andere Extrem. Die Wirkung der Fensterung steigt in dieser Liste von oben nach unten. Der Effekt der Fensterung wird am Beispiel der 3.5-periodigen Sinusfunktion erläutert.

Um die eigentliche Frequenz herum erscheinen im Spektrum weitere Harmonische, deren Amplitude mit zunehmender Entfernung von der eigentlichen Frequenz abnimmt. Das Rechteckfenster erzeugt nun eine deutliche, recht dünne Spitze bei 3 bzw. 4. Aber auch in größerer Entfernung ist noch ein sehr stark verfälschender Einfluss auf das Spektrum vorhanden. Bei Anwendung der anderen Fenster wird die Spitze zunehmend breiter, aber in größerer Entfernung von der eigentlichen Frequenz klingt der Fehler immer schneller ab. Jede Fensterfunktion stellt also nur einen Kompromiss dar.

Um sehr nahe beieinander liegende Harmonische unterscheiden zu können, ist ein Rechteckfenster angemessen. Um mehrere nicht so nahe beieinander liegende Harmonische genau in ihren Amplituden und Phasen zu bestimmen, ist es ratsam, das Blackman-Fenster zu benutzen.

- Beachten Sie, dass gegenüber der Funktion Spec() die mittleren Harmonischen noch nicht mit einem Faktor 2 beaufschlagt sind, um als Amplituden interpretiert werden zu können.
- Wenn Sie auf eine direkte Interpretierbarkeit der Amplituden des Spektrums Wert legen, benutzen Sie die Funktion Spec(). Beachten Sie aber, dass Sie sinnvolle Berechnungen im Frequenzbereich wie Filterung oder Faltung nur durchführen können, wenn Sie die Funktion FFT() zur Berechnung des Spektrums herangezogen haben.
- Im **Spektralpaket** (enthalten in Professional/Enterprise-Edition) finden Sie noch weitere, wesentlich leistungsfähigere Funktionen und auf die Bedürfnisse der Signalanalyse zugeschnittene Funktionen zur Berechnung von Spektren. Beispiel: **AmpSpectrumRMS()** berechnet ein Amplitudenspektrum (Harmonische als Effektivwerte), mit gleitendem Fenster und linearer Mittelung.
- Die optionalen Parameter für Fenster-Funktion und Modus werden seit ab FAMOS 2022 unterstützt. Wenn diese Parameter weggelassen werden, gelten wie in früheren Versionen die globalen Einstellungen, die über den Dialog 'Optionen'/'Funktionen', mit dem Kommando FFTOPTION oder der Funktion SetOption() eingestellt wurden.
- Wenn die Länge des als Parameter übergebenen Datensatzes 2^27 überschreitet, wird eine entsprechende Fehlermeldung ausgegeben. Es ist dann keine Berechnung möglich. Verkürzen Sie den Datensatz mit den Funktionen Leng() oder Red2(). Die maximal bearbeitbare Punktezahl ist 134.217.728. Dann wird ein komplexer Datensatz der Länge 67.108.865 erzeugt.
- Die hier implementierte FFT liefert nur die positive Seite des Spektrums. Bei den hier vorliegenden reellen Datensätzen ist das Spektrum stets konjugiert komplex symmetrisch, so dass die andere Hälfte keine zusätzlichen Informationen enthält. Zudem ist die Darstellung von negativen Frequenzen oder von Frequenzen oberhalb der halben Abtastfrequenz sehr fragwürdig.
- Hat der übergebene Datensatz N Punkte und ist N gerade, so hat das erzeugte Spektrum N/2 + 1 Punkte. Da der erste und letzte Wert (Mittelwert und höchste Harmonische) stets nur reell sind, die anderen Spektrallinien aber komplex, folgt, dass im Spektrum genauso viele signifikante Zahlen enthalten sind wie im Datensatz, also keine Information verloren gegangen ist. Ist N ungerade (nur möglich beim Mixed-Radix-Verfahren), so hat das erzeugte Spektrum (N-1)/2 + 1 Punkte. Die höchste Harmonische ist dann komplex und nicht mit dem Faktor 2 beaufschlagt.
- Das **Mixed-Radix-Verfahren** hat den Vorteil, dass man zu Datensätzen mit technisch üblichen Abtastfrequenzen (z.B. 1kHz und Vielfache im 1/2/5-Raster) und geeigneter Wahl der Eingangslänge Spektren mit "runden" Frequenzlinienabständen erhält. Ein mit 1kHz abgetasteter Datensatz liefert z.B. bei einer Länge von 1000 Samples (1000 = 2^3 * 5^3) ein Spektrum mit 1Hz Frequenzlinienabstand, bei einer Länge von 20000 Samples (= 2^5 * 5^4) einen Frequenzlinienabstand von 50mHz.
- Aus dem Spektrum kann der Datensatz wieder konstruiert werden, wenn ein Rechteckfenster bei der Berechnung des Spektrums gewählt war.
- Als x-Einheit des Spektrums wird stets Hz gewählt. Die y-Einheit des Betrages entspricht der des Datensatzes. Die Phase wird in Grad angegeben.
- Der x-Offset des übergebenen Datensatzes sollte Null sein. Ist er nicht Null, wird eine entsprechende Warnung erzeugt. Der x-Offset wird stets zu Null angenommenen. Möchten Sie den Einfluss eines x-Offsets dennoch in der Phase berücksichtigt haben, müssen Sie nachträglich die Phase mit einer linearen Funktion zur Korrektur beaufschlagen.

Beispiele:

```
MPfft = FFT(NDdata, 1, 0)
```

Einfache Anwendung zur Berechnung des Spektrums eines Datensatzes, dessen Länge eine Zweierpotenz ist, wobei auf eine direkte Interpretierbarkeit der Amplituden kein Wert gelegt wird (Hammingfenster).

```
DPTransfer=dB(FFT(NDout, 0, 2)/FFT(NDin, 0, 2))
; ist äquivalent zu:
DPTransfer=dB(FFT(Red2(NDout), 0, 0)/FFT(Red2(NDin), 0, 0))
```

Berechnung einer Übertragungsfunktion in dB von zwei gleichlangen Datensätzen, deren Länge keine Zweierpotenz ist.

```
First1000Samples = CutIndex(data1kHz, 1, 1000)
FFT_with_1Hz_distance = FFT(First1000Samples, 0, 3)
```

Die FFT der ersten 1000 Samples eines mit 1kHz abgetasteten Datensatzes wird berechnet. Das Ergebnis hat 501 Spektrallinien an den Frequenzen 0, 1, 2, ... 500 Hz.

```
; Achtung bei ungerader Eingangslänge:
First125amples = CutIndex(data1kHz, 1, 125)
FFT_with_8Hz_distance = FFT(First125Samples, 0, 3)
```

Die FFT der ersten 125 Samples eines mit 1kHz abgetasteten Datensatzes wird berechnet. Das Ergebnis hat 63 Spektrallinien an den Frequenzen 0, 8, 16,... 496 Hz. Die letzte Spektrallinie ist komplex und liegt nicht bei der halben Abtastfrequenz.

```
NDwindow = iFFT(FFT(Ramp(0, 1, 256) * 0 + 1, 2, 0))
```

Es wird ein Datensatz, der konstant gleich 1 ist, erzeugt. Die aktuell eingestellte Fensterfunktion gewichtet diesen Datensatz bei der Berechnung der FFT, die iFFT zeigt dann die Fensterfunktion (hier: Hanning).

Siehe auch:

AmpSpectrumRMS_1, AmpSpectrumRMS, Spec, iFFT, Red2, dB, ACF

Unterstützt ab:

Optionale Parameter und Mixed-Radix-Verfahren unterstützt ab Version 2022.

FFTOPTION

FFT-Einstellungen setzen

Deklaration:

```
FFTOPTION EwFenster EwModus
```

Parameter:

EwFenster	Fenster-Funktion
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman/Harris
	5 : Flat-Top
EwModus	Modus
	0 : Datensatzlänge auf nächste 2er Potenz verringern
	1 : Datensatzlänge auf nächste 2er Potenz vergrößern und mit Nullen auffüllen

Beschreibung:

Das Kommando FFTOPTION ist veraltet, statt dessen sollte in neu zu erstellenden Sequenzen die entsprechenden optionalen Parameter der Funktionen FFT() und Spec() bzw. die Funktion SetOption() verwendet werden.

Der Befehl stellt eine Automatisierung der entsprechenden Einstellungen im Dialogfeld 'Extra'/'Optionen'/'Funktionen' dar.

Die Einstellungen wirken auf die Funktionen FFT(), Spec(), CCF() und ACF().

Multithreading: Der Befehl wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
FFTOPTION 4 1
FFTresult = FFT(data)
```

Es wurde eine Blackman/Harris-Fensterfunktion ausgewählt. Der Datensatz wird bis zur nächsten 2er- Potenz erweitert und mit Nullen aufgefüllt.

Siehe auch:

FFT, Spec, SetOption

FileClose

Schließen einer geöffneten Datei

Deklaration:

```
FileClose ( EwFileID ) -> EwStatus
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit einer FileOpen*()-Funktion zurückgegeben wurde-
EwStatus	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt.
	-1 : Fehler beim Ausführen der Funktion.

Beschreibung:

Die Funktion schließt eine Datei, die mit einer der Funktionen FileOpenDSF(), FileOpenFAS(), FileOpenASCII(), FileOpenASCII2(), FileOpenXLS() oder FileOpenXLS2() geöffnet wurde. Falls die Datei zum Schreiben geöffnet war, wird erst hier die eventuell geänderte Datei wirklich geschrieben.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden
- Die Abfrage des Rückgabewertes ist optional, dieser sollte aber besonders beim Schreiben von Dateien ausgewertet werden.

Beispiele:

Eine Datei "xxx.dat" wird zum Schreiben im FAMOS-Format geöffnet. Zwei Variablen werden dieser Datei hinzugefügt. Mit dem Aufruf von FileClose() wird die Datei gespeichert.

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 1)
IF idFile > 0
   data = Ramp(0, 1, 100)
   sinData = sin(data)
   status= FileObjWrite(idFile, data)
   status= FileObjWrite(idFile, sinData)
   status= FileClose(idFile)
END
```

Siehe auch:

FileOpenDSF, FileOpenFAS, FileOpenASCII, FileOpenXLS, FileResetAll

FileComm?

Dateikommentar einer Datei im imc/FAMOS-Format abfragen

Deklaration:

```
FileComm? ( EwFileID ) -> TxKommentar
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit der Funktion FileOpenDSF() zurückgegeben wurde.
TxKommentar	Dateikommentar. Leerer Text, wenn kein Kommentar vorhanden oder im Fehlerfall.

Beschreibung:

Die Funktion erfragt den Dateikommentar einer geöffneten Datei im imc/FAMOS-Format. Die Datei muss mit der Funktion FileOpenDSF() geöffnet worden sein.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird ein leerer Text zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.

Beispiele:

Eine FAMOS-Datei "xxx.dat" wird zum Lesen und Schreiben im imc FAMOS-Format geöffnet. Der Kommentar wird abgefragt, falls er nicht vorhanden ist, wird ein Kommentar eingetragen und die Datei wieder geschlossen:

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 2)
txComment = FileComm?(idFile)
IF TLeng(txComment) = 0
   FileSetComm(idFile, "Checked")
END
FileClose(idFile)
```

Siehe auch:

FileOpenDSF, FileSetComm

FileErrCode?

Fehlercode der zuletzt ausgeführten Dateifunktion holen

Deklaration:

```
FileErrCode? ( ) -> EwFehlercode
```

Parameter:

EwFehlercode	Letzter Fehlercode

Beschreibung:

Die Funktion liefert den Fehlercode der zuletzt ausgefuhrten Funktion aus der Gruppe der File*()-Funktionen.

Allgemeine Fehlercodes:

0	Funktion war erfolgreich.
1	Nicht genügend Speicher.
2	Zu viele geöffnete Dateien.
3	Der Identifikator ist ungültig.
4	Zu diesem Identifikator existiert keine geöffnete Datei.
5	Abbruch durch Nutzer.
6	Ungültiger Objekt-Index.
7	Versuch, eine im "Nur Lesen"-Modus geöffnete Datei zu ändern.
8	Funktion ist nur für DSF-Dateien erlaubt.
9	Ungültige Option beim Öffnen der Datei.
10	Interner Fehler.
11	Datei kann nicht geöffnet werden.
12	Diese Funktion kann auf ASCII-Dateien nicht angewendet werden.
13	Diese Funktion kann nur auf ASCII-Dateien angewendet werden.
14	Fehler beim Schreiben der Datei (Datenträger voll ?)
15	Fehler beim Lesen der Datei.
16	Versuch, eine im 'Schreiben'-Modus geöffnete Datei zu lesen.
17	Die Bibliothek 'im22form.dll' ist nicht vorhanden oder kann nicht geladen werden.
18	Diese Funktion kann nur auf XLS-Dateien angewendet werden.

Die folgenden Fehlermeldungen beziehen sich auf Dateien im imc FAMOS-Format, die mit FileOpenDSF() geöffnet wurden.

21	Nicht ausreichender temporärer Speicherplatz.
22	Fehler beim Schreiben, Platte voll
23	Datei kann nicht angelegt werden.
24	Zu viele und zu große Objekte in der Datei, Dateigröße nicht mehr verwaltbar.
25,26	Fehler beim Schreiben der temporären Datei.
27	Datei ist schreibgeschützt.
28	Die Datei kann nicht geöffnet werden.
29..69	Formatfehler in der Datei.

Die folgenden Fehlermeldungen beziehen sich auf nutzerdefinierte Dateiformate, die mit FileOpenFAS() geöffnet wurden.

71	Abbruch durch Nutzer.
72	Nicht ausreichend Speicher.
73	Fehler beim Öffnen der Formatdefinitions-Datei.
74	Fehler beim Öffnen der Messwert-Datei.
75	Die Datei-Assistent-Dll kann nicht gefunden werden.

76	Falsche Version der Definitions-Datei.
77	Fehler beim Lesen der Formatdefinitions-Datei
78	Fehler beim Lesen der Messwert-Datei.

Die folgenden Fehlermeldungen beziehen sich auf Dateien, die mit FileOpenXLS() oder FileOpenXLS2() geöffnet worden sind.

121	Versionskonflikt mit der Bibliothek 'im22form.dll'. Die Datei ist zu alt.
122	Versionskonflikt mit der Bibliothek 'im22form.dll'. Die Datei ist zu neu.
123	Implementierungsproblem.
124	Unbekannter interner Fehler.
136	Mindestens einer der zu speichernden Datensätze hat einen nicht erlaubten Datentyp.
137	Nicht genügend Speicher.
138	Es können keine Daten gespeichert werden. Alle übergebenen Datensätze haben einen nicht erlaubten Typ (z.B. Text) oder die Länge 0.
139	Daten vom Typ 'Zeitstempel-ASCII' können nur einzeln und nicht zusammen mit anderen Datensätzen in der Datei gespeichert werden. Sie dürfen außerdem weder Events noch Segmente besitzen.
140	Für den Modus 'Gemeinsame Skalierungsspalte' müssen alle Datensätze eine monotone x- bzw. Zeitspur aufweisen.
152	Die angegebene Vorlagen-Datei enthält keine gültige Exportvorlage.
153	Die angegebene Vorlagen-Datei hat eine ungültige bzw. nicht passende Version.
154	Die angegebene Vorlagen-Datei hat den falschen Typ.
168	Die angegebene Datei kann nicht geöffnet werden. Bitte überprüfen Sie den Dateinamen.
169	Fehler beim Schreiben in die Ausgabedatei. Eventuell ist der Datenträger voll.
184	Fehler bei der EXCEL-Ansteuerung.
186	Die Skalierungsspalte ist leer oder enthält ungültige Werte.
187	An der angegebenen Tabellen-Position sind keine lesbaren Werte vorhanden.
188	Excel kann nicht gestartet werden.

Beispiele:

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 0)
IF idFile >= 1
   status = FileObjWrite(idFile, data)
   IF status = -1
      BoxOutput("Fehler beim Schreiben: " + FileErrText?(), FileErrCode?(), "", 0)
   END
END
```

Falls die Variable nicht in die Datei geschrieben werden kann, wird eine Ausgabebox mit Fehlertext und Fehlercode erzeugt.

Siehe auch:

FileErrText?

FileErrText?

Fehlertext der zuletzt ausgeführten Dateifunktion holen.

Deklaration:

```
FileErrText? ( ) -> TxFehlerText
```

Parameter:

TxFehlerText	Letzter Fehlertext

Beschreibung:

Die Funktion liefert den Fehlertext der zuletzt ausgeführten Funktion aus der Gruppe der File*()-Funktionen.

Beispiele:

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 0)
IF idFile >= 1
   status = FileObjWrite(idFile, data)
   IF status = -1
      BoxOutput("Fehler beim Schreiben: " + FileErrText?(), FileErrCode?(), "", 0)
   END
END
```

Falls die Variable nicht in die Datei geschrieben werden kann, wird eine Ausgabebox mit Fehlertext und Fehlercode erzeugt.

Siehe auch:

FileErrCode?

FileLineRead

Die nächste Zeile(n) aus einer geöffneten ASCII-Datei lesen

Deklaration:

```
FileLineRead ( EwFileID, TxZeile, Null ) -> EwStatus
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit der Funktion FileOpenASCII() zurückgegeben wurde
TxZeile	Textvariable oder Textfeld zur Aufnahme des gelesenen Textes
Null	Reservierter Parameter, immer 0.
EwStatus	Erfolg der Funktion (optional).
	0 : Funktion erfolgreich ausgeführt.
	1 : Das Dateiende ist erreicht.
	-1 : Ein Fehler ist aufgetreten.

Beschreibung:

Die Funktion liest die nächste Zeile(n) aus einer geöffneten ASCII-Datei. Die Datei muss vorher mit der Funktion FileOpenASCII() geöffnet worden sein.

Das Zeilenende wird durch die Kombination 'CarriageReturn'/'LineFeed' (ASCII-Zeichen 13 und 10) oder nur 'LineFeed' erkannt. Im gelesenen Text sind diese Zeichen nicht enthalten.

Die als 2. Parameter übergebene Variable muss zum Zeitpunkt des Aufrufs bereits existieren. Sie kann z. B. über eine einfache Zuweisung

```
TxZeile = ""
```

bzw. für Textfelder:

```
TxAlleZeilen = TxArrayCreate(0)
```

erzeugt werden.

Wenn als 2. Parameter ein Textfeld übergeben wird, so werden alle verbleibenden Zeilen bis zum Dateiende gelesen und diese als neue Elemente an das Textfeld angehängt.

- Die Funktion ist beispielsweise dazu geeignet, Protokolldateien oder auch kurze Zahlenreihen im ASCII-Format (mit anschließender Konvertierung Text in Zahl) einzulesen. Für lange Messwertdateien im ASCII-Format eignet sich dieser Befehl aus Geschwindigkeitsgründen i. a. nicht.
- Zum Lesen solcher Dateien empfiehlt sich die Verwendung des ASCII-Import-Assistenten oder die Erstellung eines Dateifilters mit dem Datei-Assistenten.
- Im Fehlerfall (Rückgabe von -1) kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.

Beispiele:

Aus einer mehrspaltigen Textdatei mit Zahlenwerten (Spalten separiert durch Komma) soll die erste Spalte ausgelesen werden:

```
idFile = FileOpenASCII("z:\000.txt", 0)
TxLine = ""
channel1 = EMPTY ; one blank channel
WHILE 1
   isEOF = FileLineRead(idFile, TxLine, 0) ; read one text line from file
   IF isEOF = 1
      BREAK  ; End Of File
   END
   TxLineSplit = TxSplit(TxLine, ",") ; split comma separated values from line
   TxChannel1 = TxLineSplit[1] ; get 1. separated value for channel 1
   SvChannel1 = TtoSV(TxChannel1, "f") ; convert text to float value
   AppendLoop(channel1, SvChannel1) ; attach value to value
END
AppendLoopEnd(channel1)
FileClose(idFile) ; close file processing
```

Anmerkung: Solche Import-Aufgaben können oft bequemer mit dem ASCII-Import-Assistenten erledigt werden.

Die selbe Aufgabe kann mit einem Textfeld wesentlich eleganter und schneller gelöst werden:

```
idFile = FileOpenASCII("z:\000.txt", 0)
TxaLines = TxArrayCreate(0)
ret = FileLineRead(idFile, TxaLines, 0) ; read all text lines from file
```

```
IF ret = 0
   TxaLines = TxRegexMatch(TxaLines, "^[^,]+", " ", 0, 0) ; split 1st column (comma separated) from line
   ; To get the 3rd column, you could use the following line (note the '{2}' for the the zero based column index):
   ; TxaLines = TxRegexMatch(TxaLines, "^(?:[^,]+,){2}([^,]+)", " ", 0, 1)
   channel = TxArrayToChannel(TxaLines, 0);
END
FileClose(idFile) ; close file processing
```

In einer Protokolldatei sind Ereignisse in der Form "12.2.95 13:00:42 Alarm Nr.123 Datei::c12_34" zeilenweise abgelegt. Diese Meldungen werden nacheinander angezeigt. Der Anwender kann wahlweise den angegebenen Datensatz laden und anzeigen.

```
idFile = FileOpenASCII("c:\dat\prot.dat", 0)
txRow = ""
ok = FileLineRead(idFile, txRow, 0)
WHILE ok = 0
   yes = BoxMessage("Anzeigen?", txRow, "?4")
   IF yes
      txFile= TPart(txRow, TxWhere(txRow, "::")+2, 100)
      FileLoad(txFile + ".dat, "", 0)
      SHOW <txFile>
   END
   ok = FileLineRead(idFile, txRow, 0)
END
FileClose(idFile)
```

Siehe auch:

FileOpenASCII, FileOpenASCII2, FileLineWrite, FileClose

FileLineWrite

Zeile(n) an eine geöffnete ASCII-Datei anhängen

Deklaration:

```
FileLineWrite ( EwFileID, TxZeilen, EwOption ) -> EwStatus
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit der Funktion FileOpenASCII() zurückgegeben wurde
TxZeilen	Zu schreibender Text. Erlaubt sind die Datentypen Text und Textfeld.
EwOption	Optionen
	0 : Zeilenumbruch anhängen
	1 : Kein Zeilenumbruch
EwStatus	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt.
	-1 : Ein Fehler ist aufgetreten.

Beschreibung:

Die Funktion hängt eine (wenn als 2. Parameter ein Text übergeben wird) oder mehrere Zeilen (wenn der 2. Parameter ein Textfeld ist) an eine geöffnete ASCII-Datei an. Die Datei muss vorher mit der Funktion FileOpenASCII() zum Schreiben geöffnet worden sein.

An den Zeilentext wird (optional) ein 'CarriageReturn'- (ASCII 13) und ein 'LineFeed'-Zeichen (ASCII 10) angehängt, um für den Zeilenumbruch nach dem geschriebenen Text zu sorgen.

Wenn ein Textfeld angegeben wird, wird jedes Element als neue Zeile geschrieben. Der Optionsparameter wird dann ignoriert.

Der Text wird zunächst intern zwischengespeichert. Das eigentliche Schreiben der Datei erfolgt beim abschließenden Aufruf von FileClose().

- Die Funktion ist beispielsweise dazu geeignet, Protokolldateien oder auch kurze Zahlenreihen im ASCII-Format (mit vorhergehender Konvertierung Zahl in Text) zu schreiben. Für lange Messwertdateien im ASCII-Format eignet sich dieser Befehl aus Geschwindigkeitsgründen i. a. nicht. Für solche Aufgaben können Sie die Funktion FileOpenASCII2() verwenden.
- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlermitteilung angezeigt.
- Die Abfrage und Auswertung des Rückgabewertes ist zwar nicht zwingend notwendig, wird aber empfohlen.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.

Beispiele:

Bei einer komplexen Auswertung werden anfallende Fehler- oder Statusmeldungen zusammen mit der aktuellen Zeit in eine ASCII-Datei gespeichert, z. B. "12.10.23 13:00:02 Messung xyz: Verletzung des Toleranzbandes".

```
idFile = FileOpenASCII("c:\dat\prot.dat", 2)
;; ... Auswertung ...
;; ...
IF TLeng(TxStatus) > 0
   txTime = TimeToText(TimeSystem?(), 0)
   txRow = txTime + TxStatus
   ret= FileLineWrite(idFile, txRow, 0)
END
;; ... weitere FileLineWrite ...
;; ...
ret= FileClose(idFile)
```

Eine Text-Datei mit mehreren, durch Komma abgetrennten, Spalten wird gelesen. Die 3. Spalte wird extrahiert und in eine neue Datei gespeichert.

```
; Read file with at least 3 comma separated columns
idFile = FileOpenASCII("z:\000.txt", 0)
TxaLines = TxArrayCreate(0)
FileLineRead(idFile, TxaLines, 0) ; read all text lines from file
TxaLines = TxRegExMatch(TxaLines, "^(?:[^,]+,){2}([^,]+)", " ", 0, 1) ; split 3rd column (comma separated) from line
FileClose(idFile) ;
; Write new file with 1 column
idFile = FileOpenASCII("z:\000_col3.txt",1)
FileLineWrite(idFile, TxaLines, 0) ; write all text lines to file
FileClose(idFile) ;
```

Siehe auch:

FileOpenASCII, FileOpenASCII2, FileLineRead, FileClose

FileLoad

Lädt die Messwertdatei im angegebenen Format und trägt alle enthaltenen Datenobjekte in die FAMOS Variablenliste ein.

Deklaration:

```
FileLoad ( TxDatei, TxFormat, EwOptionen ) -> EwErfolg
```

Parameter:

TxDatei	Name der Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet.
TxFormat	Spezifikation des Dateiformats.
EwOptionen	Optionen
	0 : Standardverhalten
	1 : Eine eventuell durch das Importfilter festgelegte Messungszugehörigkeit wird ignoriert.
EwErfolg	Erfolg der Funktion
	-1 : Fehler beim Laden der Datei. Die Fehlerursache kann mit der Funktion GetLastError() erfragt werden.
	>=0 : Anzahl der geladenen Datenobjekte (Optional).

Beschreibung:

Der Inhalt der angegebenen Datei(en) wird komplett geladen und die erzeugten Datenobjekte in die FAMOS-Variablenliste übernommen. Die Funktion liefert keine Information, welche Variablen (unter welchem Namen) erzeugt worden sind und eignet sich damit für Anwendungen, in denen die Struktur der zu ladenden Dateien und damit das Ergebnis des Ladevorgangs vorhersagbar ist.

Für Dateien mit unbekannten oder variablen Inhalt oder wenn Sie nur ausgewählte Kanäle aus einer Multikanaldatei laden möchten, verwenden Sie besser die leistungsfähigeren Funktionen FileOpenDSF() bzw. FileOpenFAS().

Sie können im Dateinamen Jokerzeichen (Wildcards) angeben, um eine Reihe von Dateien zu laden. Das Jokerzeichen '?' steht dabei für genau ein beliebiges Zeichen. Das Jokerzeichen '*' steht für eine unbestimmte Anzahl von beliebigen Zeichen.

Tipp: Verwenden Sie den Funktions-Assistent für die Parametrierung der Funktion. Sie können hier bequem alle möglichen Importformate in einer Liste auswählen.

Für die Spezifikation des Formats sind mehrere Varianten möglich:

Laden im Standard-imc/FAMOS-Format
Für <TxFormat> ist dann entweder ein leerer Text oder die Kennung "imc/FAMOS" anzugeben.

Falls der übergebene Dateiname keine Erweiterung besitzt, wird automatisch ".dat" angehängt.

Sie können außerdem noch spezifizieren, dass die Option 'Schnelles Laden' verwendet werden soll, in diesem Fall ist als Formatkennung "imc/FAMOS|Quick" zu verwenden. Diese Option bewirkt, dass die Daten beim Laden nicht kopiert, sondern intern als Verweis auf die Originaldatei verwaltet werden. Dies kann das Laden sehr großer Datensätze beschleunigen, ist aber nur sinnvoll, wenn Sie die geladenen Datensätze später nicht verändern wollen.

Laden einer Textvariablen aus einer Textdatei
Um den Inhalt einer Textdatei in eine Textvariable zu übertragen, geben Sie die Formatkennung "imc/Text" an.

Falls der übergebene Dateiname keine Erweiterung besitzt, wird automatisch ".txt" angehängt. Die erzeugte Textvariable trägt denselben Namen wie die Datei.

Sie können zusätzlich die Zeichensatz-Kodierung (Code page) der Datei angeben. Standard ist die aktuelle Windows-Codepage.

"imc/Text"	Die aktuelle Windows-ANSI-Codepage wird angenommen.
"imc/Text/auto"	Die Kodierung wird automatisch ermittelt. Dazu wird geprüft, ob die Datei mit einem definierten BOM (Byte order mark) beginnt. Erkannt werden die BOM für UTF-8 und UTF-16 ("Big endian" und "Little endian" byte order). Wenn kein solcher BOM erkannt wird, wird die aktuelle Windows-Codepage angenommen.
"imc/Text/UTF8"	UTF-8-Kodierung
"imc/Text/UTF16_LE"	UTF-16 LE (Byte-Reihenfolge "Little endian", niederwertiges Byte zuerst)
"imc/Text/UTF16_BE"	UTF-16 BE (Byte-Reihenfolge "Big endian", höherwertiges Byte zuerst)

Import mit Format-Beschreibungsdatei (*.FAS, imc-Dateiassistent).
Das Format ist durch eine Import-Beschreibungsdatei (Importfilter) festgelegt, die mit dem imc- Dateiassistenten erstellt wurde. Eine solche Importfilter-Datei hat üblicherweise die Erweiterung ".fas" und muss sich im voreingestellten Definitionsverzeichnis ("Extras" / "Optionen" / "Verzeichnisse") befinden. Der Name dieser Datei wird dann als Parameter übergeben, die Erweiterung ".fas" kann weggelassen werden.

Import mit einer entsprechenden Erweiterungsbibliothek (*.DLL).
Die Syntax von <TxFormat> lautet dann:

```
"#DLLName|FormatName|Parameter"
```

- DLLName: gibt den Dateinamen der Erweiterungsbibliothek an, die Erweiterung ".DLL" kann weggelassen werden.
- FormatName: gibt den Namen des gewünschten Formats an (eine Erweiterungsbibliothek kann durchaus mehrere Formate unterstützen). Kann weggelassen werden, wenn die DLL nur ein einziges Format zur Verfügung stellt und keine weiteren Parameter benötigt.

- Parameter: dient zur optionalen Übergabe weiterer Parameter an die Bibliothek. Ob Parameter benötigt werden sowie deren Syntax hängt von der jeweiligen Erweiterungsbibliothek ab. Oft nicht benötigt, kann dann weggelassen werden.Suchen Sie ggf. nach dem Namen des verwendenden Formats in der Hilfe, wo die formatspezifische Verwendung des Parameters beschrieben wird.

Im Lieferumfang von FAMOS sind bereits eine Reihe von Erweiterungsbibliotheken zum Import verbreiteter Dateiformate (wie z.B. MatLab) enthalten.

Der in FAMOS integrierte ASCII-Import-Assistent ist ebenfalls eine Erweiterungsbibliothek (ImportAscii1.dll). Mit ihm erstellte Filter können so hier ebenfalls verwendet werden (siehe Beispiel).

Import mit einem abgeleiteten Importfilter

Der Import erfolgt durch ein abgeleitetes Importfilter. Die Syntax des Parameters <TxFormat> lautet dann:

```
"$FormatName"
```

[FormatName] entspricht dem vom Anwender selbst vergebenen Namen, wie er z.B. im FAMOS-Dialog "Optionen"/"Importfilter" angezeigt wird.

Sonderformat "ByteBlob": Uninterpretierte Binär-Daten.

Der komplette Dateiinhalt wird binär und uninterpretiert in einen Datensatz eingelesen. Geben Sie dafür die Formatkennung "imc/ByteBlob" an.

Das resultierende Datenformat ist '1 Byte ohne Vorzeichen'. Wird im Allgemeinen nur für Spezialanwendungen benötigt, z.B. im Zusammenspiel mit dem FAMOS-Datenbankkit, um beliebige Dateien (z.B. PDF, Bilder, Videos) zusammen mit den eigentlichen Messdaten in Datenbanken ablegen zu können.

- Bestehende Variablen mit gleichem Namen werden ohne Warnung überschrieben.
- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im "Optionen"/ "Funktionen"-Dialog oder mit der Funktion SetOption). Wenn diese Option ausgeschaltet ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine 0 zurückgegeben und keine Fehlermitteilung angezeigt. Ggf. kann die Fehlerursache mit der Funktion GetLastError() abgefragt werden.

Beispiele:

Laden einer Datei im FAMOS-Format:

```
FileLoad("test","", 0)
```

..oder mit 'Quickload'-Option:

```
FileLoad("test","imc/Famos|Quick", 0)
```

Laden einer Text-Datei und Zuweisung des Inhalts als anwenderderfinierte Eigenschaft an einen Datensatz:

```
FileLoad("mycomment.txt","imc/Text", 0)
UserPropSet(Channel1, "UserComment", mycomment, 1, 0)
```

Laden einer MatLab-Datei (mit Erweiterungs-DLL)

```
FileLoad("1.mat","#MatLabImportExport", 0)
```

Mit dem ASCII Import Assistenten definierte Import-Filter können ebenfalls verwendet werden.

```
FileLoad("1.txt","#ImportAscii1.dll|FilterName", 0)
```

[FilterName] entspricht dem Namen, unter dem das Filter im ASCII-Import-Dialog gespeichert wurde und ist identisch mit dem angezeigten Formatnamen im "Datei laden"-Dialog.

Laden einer Datei im LeCroy-Format (mit Dateiassistent, 'lecroy.fas')

```
FileLoad("1.lec","lecroy", 0)
```

Laden einer EXCEL-Datei mit einem abgeleiteten Importfilter. Dieses wurde auf Basis des EXCEL-Formats erstellt. Die privaten Einstellungen sind so gesetzt, dass die Zellen B2 bis C300 gelesen werden. Das Importfilter wurde unter dem Namen 'EXCEL (B2-C300)' gespeichert.

```
FileLoad("c:\test\result.xls","$EXCEL (B2-C300)", 0)
```

Siehe auch:

FileSave, FileOpenDSF, FileOpenFAS

FileName?

Der Name der Datei, aus der ein Datenobjekt geladen worden ist, wird ermittelt.
Alternativer Name: **DateiName?**

Deklaration:

```
FileName? ( Datenobjekt ) -> TxDateiname
```

Parameter:

Datenobjekt	Datenobjekt, dessen Herkunft ermittelt werden soll.
TxDateiname	Dateiname (kompletter Pfad)

Beschreibung:

Falls das Datenobjekt in imc FAMOS durch Laden aus einer Datei erzeugt worden ist, wird der komplette Pfadname der Datei zurückgegeben.

Falls das Datenobjekt nicht durch Laden aus einer Datei erzeugt worden ist, wird ein leerer Text zurückgegeben.

Beispiele:

```
FileLoad("c:\imc\dat\slope.dat", "", 0)
RENAME slope TestData
TxFileName = FileName?(TestData)
```

TxFileName enthält nun den Text "c:\imc\dat\slope.dat".

Siehe auch:

Comm?

FileObjDel

Die Funktion entfernt ein Objekt aus dem Inhaltsverzeichnis einer geöffneten Datei.

Deklaration:

```
FileObjDel ( EwFileID, EwObjektIndex ) -> EwStatus
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit der Funktion FileOpenDSF() zurückgegeben wurde.
EwObjektIndex	Der Index des zu entfernenden Objektes. Das erste Objekt hat den Index 1.
EwStatus	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt.
	-1 : Fehler beim Ausführen der Funktion.

Beschreibung:

Diese Funktion entfernt das angegebene Objekt aus der Inhaltsliste der Datei. Physisch geschrieben wird die geänderte Datei erst beim Aufruf von FileClose().

Die Datei muss mit der Funktion FileOpenDSF() zum Schreiben geöffnet worden sein (Modus 2='Anhängen/Ändern').

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden
- Die Abfrage und Auswertung des Rückgabewertes ist war nicht zwingend notwendig, wird aber empfohlen.

Beispiele:

Die Datei 'Versuch.dat' im FAMOS-Format wird geladen. Alle Kanäle, deren Maximum größer als 20 ist, werden gelöscht. Bitte beachten Sie, dass die Kanäle von hinten beginnend aufgezählt werden. Dies ist einfacher, da das Löschen eines Kanals die Indizes der nachfolgenden Kanäle ändert.

```
idFile = FileOpenDSF("c:\dat\versuch.dat", 2)
IF idFile >= 1
   num = FileObjNum?(idFile)
   index = num
   WHILE index >= 1
      data = FileObjRead(idFile, index)
      IF max(data) > 20
         err = FileObjDel(idFile, index)
      END
      index = index-1
   END
   err = FileClose(idFile)
END
```

Siehe auch:

FileOpenDSF, FileObjWrite, FileObjRead

FileObjFind

Suche in einer geöffneten Datei nach einem Datenobjekt bestimmten Namens.

Deklaration:

```
FileObjFind ( EwFileID, TxSuchName, EwStartIndex ) -> EwIndex
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit den Funktionen FileOpenDSF() oder FileOpenFAS() zurückgegeben wurde.
TxSuchName	Name des gesuchten Objektes
EwStartIndex	Beginn der Suche, 1= von Anfang an
EwIndex	Index des Objektes bei Erfolg
	>0 : Gefundener Objekt-Index
	0 : Kein Objekt mit dem gesuchten Namen vorhanden
	-1 : Fehler beim Ausführen der Funktion.

Beschreibung:

Die Funktion sucht in der Inhaltsliste einer geöffneten Datei nach einem Datenobjekt mit dem angegebenen Namen. Zurückgegeben wird der Index bei Erfolg oder eine 0, falls das Objekt nicht gefunden werden konnte.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.
- Der Index des gefundenen Objektes ist nur gültig, bis der nächste schreibende Zugriff auf diese Datei erfolgt. Wenn z. B. ein Datensatz entfernt wird, der vor dem hier gefundenen liegt, würde dieser Index danach auf das nachfolgende Objekt in der Datei zeigen.

Beispiele:

In der Datei "Versuch.dat" im imc FAMOS-Format wird nach einem Datensatz mit dem Namen "Kanal1" gesucht. Falls dieser vorhanden ist, wird er geladen und angezeigt:

```
idFile = FileOpenDSF("c:\dat\versuch.dat", 0)
IF idFile >= 1
   index = FileObjFind(idFile, "Kanal1", 1)
   IF index > 0
      data = FileObjRead(idFile, index)
      SHOW data
   END
END
```

Siehe auch:

FileOpenDSF, FileOpenFAS, FileObjRead

FileObjName?

Ermittlung des Namens eines Datenobjektes in einer geöffneten Datei.

Deklaration:

```
FileObjName? ( EwFileID, EwObjektIndex ) -> TxName
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit den Funktionen FileOpenDSF() oder FileOpenFAS() zurückgegeben wurde.
EwObjektIndex	Der Index des zu prüfenden Objektes. Das erste Objekt hat den Index 1.
TxName	Name des Datenobjektes. Leerer Text im Fehlerfall.

Beschreibung:

Die Funktion liefert den Namen eines Datenobjektes in einer mit FileOpenFAS() oder FileOpenDSF() geöffneten Datei.

Der Index muss zwischen 1 und der Gesamtzahl der vorhandenen Objekte liegen.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird ein leerer Text zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.

Beispiele:

Eine Datei im FAMOS-Format wird geöffnet. Mit einer Schleife über alle Datenobjekte in der Datei wird eine Liste aller Namen in das Ausgabefeld im FAMOS-Hauptfenster ausgegeben:

```
idFile = FileOpenDSF("c:\dat\versuch1.dat", 0)
IF idFile >= 1
   count = FileObjNum?(idFile)
   index = 1
   WHILE index <= count
      TxName = FileObjName?(idFile, index)
      BoxOutput(TxName, EMPTY, "", 1)
      index = index +1
   END
   ret=FileClose(idFile)
END
```

Siehe auch:

FileOpenDSF, FileOpenFAS, FileObjType?, FileObjFind, FileObjRead

FileObjNum?

Die Anzahl der Datenobjekte in einer geöffneten Datei wird ermittelt.

Deklaration:

```
FileObjNum? ( EwFileID ) -> EwAnzahl
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit den Funktionen FileOpenDSF() oder FileOpenFAS() zurückgegeben wurde.
EwAnzahl	Zahl der Datenobjekte (Datensätze, Datengruppen, Texte) in der Datei. -1 im Fehlerfall.

Beschreibung:

Die Funktion liefert die Anzahl der Objekte (Gruppen, Texte, Datensätze) in einer geöffneten Datei.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.

Beispiele:

Eine Datei wird im imc/FAMOS-Format zum Lesen geöffnet. In einer Schleife über alle Objekte werden alle Datensätze unter dem Originalnamen angelegt. Texte und Datengruppen werden ignoriert.

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 0)
IF idFile >= 1
   count = FileObjNum?(idFile)
   index = 1
   WHILE index <= count
      type = FileObjType?(idFile, index)
      IF type = 1
         TxName = FileObjName?(idFile, index)
          <TxName> = FileObjRead(idFile, index)
      END
      index = index + 1
   END
   ret = FileClose(idFile)
END
```

Siehe auch:

FileOpenDSF, FileOpenFAS, FileObjRead, FileObjName?

FileObjRead

Ein Datenobjekt wird aus einer geöffneten Datei gelesen

Deklaration:

```
FileObjRead ( EwFileID, EwObjektIndex ) -> Variable
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit den Funktionen FileOpenDSF() oder FileOpenFAS() zurückgegeben wurde.
EwObjektIndex	Der Index des zu lesenden Objektes. Das erste Objekt hat den Index 1.
Variable	Das gelesene Datenobjekt. Datensatz, Text oder Datengruppe.

Beschreibung:

Die Funktion liest ein Objekt aus einer geöffneten Datei und weist es der Zielvariablen zu. Im Fehlerfall wird die Funktion nicht ausgeführt, eine bestehende Variable wird also nicht verändert

Beispiele:

Eine Datei wird im imc/FAMOS-Format zum Lesen geöffnet. In einer Schleife über alle Objekte werden alle Datensätze unter dem Originalnamen angelegt. Texte und Datengruppen werden ignoriert.

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 0)
IF idFile >= 1
   count = FileObjNum?(idFile)
   index = 1
   WHILE index <= count
      type = FileObjType?(idFile, index)
      IF type = 1
         TxName = FileObjName?(idFile, index)
          <TxName> = FileObjRead(idFile, index)
      END
      index = index + 1
   END
   ret = FileClose(idFile)
END
```

Siehe auch:

FileOpenDSF, FileOpenFAS, FileObjNum?, FileObjWrite

FileObjType?

Typ eines Datenobjektes in einer geöffneten Datei bestimmen.

Deklaration:

```
FileObjType? ( EwFileID, EwObjektIndex ) -> EwTyp
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit den Funktionen FileOpenDSF() oder FileOpenFAS() zurückgegeben wurde.
EwObjektIndex	Der Index des zu prüfenden Objektes. Das erste Objekt hat den Index 1.
EwTyp	Typ des Datenobjektes
	0 : Datengruppe
	1 : Datensatz
	2 : Text
	-1 : Fehler beim Ausführen der Funktion.

Beschreibung:

Die Funktion liefert den Typ eines Objektes in einer geöffneten Datei. Unterschieden wird dabei nach Datensatz, Text und Datengruppe.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.

Beispiele:

Eine Datei wird im FAMOS-Format zum Lesen geöffnet. In einer Schleife über alle Objekte werden alle Texte in unter dem Originalnamen angelegt. Datensätze und Datengruppen in der Datei werden übergangen

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 0)
IF idFile >= 1
   count = FileObjNum?(idFile)
   index = 1
   WHILE index <= count
      type = FileObjType?(idFile, index)
      IF type = 2
         TxName = FileObjName?(idFile, index)
         <TxName>= FileObjRead(idFile, index)
      END
      index = index + 1
   END
   ret=FileClose(idFile)
END
```

Siehe auch:

FileOpenDSF, FileOpenFAS, FileObjName?, FileObjFind, FileObjRead

FileObjWrite

Ein Datenobjekt wird in eine geöffnete Datei geschrieben.

Deklaration:

```
FileObjWrite ( EwFileID, Variable ) -> EwStatus
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit der Funktion FileOpenDSF(), FileOpenFAS(), FileOpenASCII2() oder FileOpenXLS2() ermittelt wurde.
Variable	Zu speichernder Datensatz, Text oder Datengruppe.
EwStatus	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt.
	-1 : Fehler beim Ausführen der Funktion.

Beschreibung:

Die Funktion fügt eine Variable zu der Inhaltsliste einer geöffneten Datei hinzu. Die Datei muss mit der Funktion FileOpenDSF() oder FileOpenFAS() zum Schreiben (Option 1 oder 2) oder mit den Funktionen FileOpenASCII2() bzw. FileOpenXLS2() geöffnet worden sein.

Dem Inhaltsverzeichnis wird eine Kopie der übergebenen Variablen hinzugefügt. Tatsächlich gespeichert wird die Datei erst beim Aufruf von FileClose().

Beim Speichern eines einzelnen Elementes aus einer Gruppe geht die Gruppeninformation verloren.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden
- Die Abfrage und Auswertung des Rückgabewertes ist war nicht zwingend notwendig, wird aber empfohlen.

Beispiele:

Eine Datei "xxx.dat" wird zum Schreiben im FAMOS-Format geöffnet. Zwei Variablen werden dieser Datei hinzugefügt. Mit dem Aufruf von FileClose() wird die Datei gespeichert.

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 1)
IF idFile > 0
   data = Ramp(0, 1, 100)
   sinData = sin(data)
   status= FileObjWrite(idFile, data)
   status= FileObjWrite(idFile, sinData)
   status= FileClose(idFile)
END
```

Siehe auch:

FileOpenDSF, FileOpenFAS, FileOpenASCII2, FileOpenXLS2, FileLineWrite

FileOpenASCII

Öffnen einer Text-Datei im ASCII-Format zum zeilenweisen Schreiben oder Lesen

Deklaration:

```
FileOpenASCII ( TxDatei, EwModus [, EwKodierung] ) -> EwDateiID
```

Parameter:

TxDatei	Name der Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet.
EwModus	Modus
	0 : Datei wird zum Lesen geöffnet.
	1 : Datei wird zum Schreiben geöffnet. Eine eventuell bestehende Datei gleichen Namens wird überschrieben.
	2 : Datei wird zum Schreiben geöffnet. Wenn die Datei schon existiert, wird der neue Inhalt angehängt.
EwKodierung	Falls die Datei mit der Option 0 zum Lesen geöffnet werden soll, kann hier zusätzlich die Zeichensatz-Kodierung (Code page) der Datei angegeben werden. Standard ist die aktuelle Windows-Codepage. (optional , Standardwert: 0)
	0 : Aktuelle Windows-ANSI-Codepage
	1 : Die Kodierung wird automatisch ermittelt. Dazu wird geprüft, ob die Datei mit einem definierten BOM (Byte order mark) beginnt. Erkannt werden die BOM für UTF-8 und UTF-16 ("Big endian" und "Little endian" byte order). Wenn kein solcher BOM erkannt wird, wird die aktuelle Windows-Codepage angenommen.
	2 : UTF-8 Kodierung.
	3 : UTF-16 mit "Little Endian"-Byte-Reihenfolge (niederwertiges Byte zuerst).
	4 : UTF-16 mit "Big Endian"-Byte-Reihenfolge (höherwertiges Byte zuerst).
EwDateiID	ID der geöffneten Datei bzw. Fehlercode
	0 : Fehler beim Öffnen der Datei. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.
	>0 : Der Identifikator für die geöffnete Datei. Dieser wird allen nachfolgenden Funktionen zur Bearbeitung dieser Datei als Parameter übergeben.

Beschreibung:

Eine Datei im ASCII-Format wird geöffnet. Anschließend können die Funktionen FileLineRead() und FileLineWrite() zum zeilenweisen Lesen bzw. Schreiben der Datei verwendet werden.

Falls kein voller Dateiname angegeben ist, wird im aktuellen Ladeverzeichnis gesucht. Das aktuelle Ladeverzeichnis wird beim Starten von imc FAMOS auf die Voreinstellung (Dialog 'Optionen'/ 'Verzeichnisse') gesetzt. Es kann mit dem Befehl LDIR oder der Funktion SetOption() umgesetzt werden. Ansonsten wird das Verzeichnis benutzt, das beim letzten Laden einer Datei angegeben wurde.

Der Identifikator der geöffneten Datei muss bei jedem folgenden Zugriff auf die Datei mit einer Dateifunktion angegeben werden. Er ist gültig bis zum Aufruf von FileClose().

Jede mit FileOpenASCII() geöffnete Datei muss durch einen Aufruf von FileClose() wieder geschlossen werden.

FAMOS kann maximal 10 geöffnete Dateien gleichzeitig verwalten. Durch fehlende Aufrufe von FileClose() kann diese Zahl schnell erreicht werden. Sie erhalten dann eine entsprechende Fehlermeldung.

Die Funktion FileResetAll() schließt alle momentan geöffneten Dateien. Dies kann nützlich sein, wenn z.B. beim Austesten von Sequenzen im Einzelschrittmodus Dateien offen geblieben sind, deren Identifikatoren Sie nicht mehr kennen.

- Die Funktion ist im Zusammenspiel mit FileLineRead() und FileLineWrite() beispielsweise dazu geeignet, Protokolldateien oder auch kurze Zahlenreihen im ASCII-Format (mit vorhergehender Konvertierung von Zahl in Text bzw. umgekehrt) zu schreiben oder zu lesen. Für lange Messwertdateien im ASCII-Format eignet sich dieser Befehl aus Geschwindigkeitsgründen im Allgemeinen nicht.
- Zum Schreiben solcher Dateien sollten Sie die Funktion FileOpenASCII2() verwenden.
- Zum Lesen solcher Dateien empfiehlt sich die Verwendung des ASCII-Import-Assistenten oder die Erstellung eines Dateifilters mit dem Datei-Assistenten.
- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine 0 zurückgegeben und keine Fehlermitteilung angezeigt. Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.
- Multithreading: Die von der Funktion zurückgegebene Datei-ID ist nur für den aktuellen Ausführungs-Thread gültig. Dateien werden am Thread-Ende (also z.B. am Ende einer mittels BEGIN_PARALLEL ausgeführten Sequenzfunktion) automatisch geschlossen.

Beispiele:

Bei einer komplexen Auswertung werden anfallende Fehler- oder Statusmeldungen zusammen mit der aktuellen Zeit in eine ASCII-Datei

gespeichert, z. B. "12.10.23 13:00:02 Messung xyz: Verletzung des Toleranzbandes".

```
idFile = FileOpenASCII("c:\dat\prot.dat", 2)
;; ... Auswertung ...
;; ...
IF TLeng(txStatus) > 0
   txTime = TimeToText(TimeSystem?(), 0)
   txRow = txTime + txStatus
   ret= FileLineWrite(idFile, txRow, 0)
END
;; ... weitere FileLineWrite ...
;; ...
ret = FileClose(idFile)
```

Siehe auch:

FileOpenASCII2, FileLineRead, FileLineWrite, FileClose, FileOpenXLS, FileOpenXLS2

FileOpenASCII2

Eine ASCII-Datei wird zum spaltenweisen Schreiben von Datensätzen geöffnet. Die Formatspezifikation erfolgt mittels einer vordefinierten ASCII-Exportvorlage.

Deklaration:

```
FileOpenASCII2 ( TxDatei, TxExportVorlage, EwModus ) -> EwDateiID
```

Parameter:

TxDatei	Name der ASCII-Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet.
TxExportVorlage	Name der zu verwendenden ASCII-Exportvorlage.
EwModus	Modus
	1 : Datei wird zum Schreiben geöffnet. Eine eventuell bestehende Datei gleichen Namens wird überschrieben.
	2 : Datei wird zum Schreiben geöffnet. Wenn die Datei schon existiert, wird der neue Inhalt angehängt.
EwDateiID	ID der geöffneten Datei bzw. Fehlercode
	0 : Fehler beim Öffnen der Datei. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.
	>0 : Der Identifikator für die geöffnete Datei. Dieser wird allen nachfolgenden Funktionen zur Bearbeitung dieser Datei als Parameter übergeben.

Beschreibung:

Diese Funktion ermöglicht das Schreiben von ASCII-Dateien, in denen Datensätze spaltenweise abgelegt werden.

Nachdem eine Datei mit dieser Funktion geöffnet wurde, können die zu speichernden Datensätze mit der Funktion FileObjWrite(..) zugefügt werden.

Jede mit FileOpenAscii2() geöffnete Datei muß durch einen abschließenden Aufruf von FileClose() wieder geschlossen werden. Die Datei wird erst dann physisch auf den Datenträger geschrieben.

FileObjWrite() und FileClose() sind die einzigen Funktionen, die auf mit FileOpenAscii2() geöffnete Dateien anwendbar sind. Zusätzliche Kopf- und Fußzeilen können durch vor- bzw. nachgeschaltete Aufrufe von FileOpenASCII()/FileClose() in Kombination mit FileLineWrite() eingefügt werden.

Die konkrete Definition des Dateiformats (z.B. Spaltenköpfe, Skalierungsspalten, Zahlenformat..) erfolgt durch die angegebene ASCII-Exportvorlage. Das Erstellen und die Verwaltung solcher Vorlagen erfolgt über den Menüpunkt "Extra / Optionen / Dateiformate / ASCII-Speichern"

Tipp: Verwenden Sie den Funktions-Assistent für die Parametrierung der Funktion. Sie können hier bequem in einer Liste aller ASCII-Exportvorlagen auswählen.

Falls kein voller Dateiname angegeben ist, wird im aktuellen Ladeverzeichnis gesucht. Das aktuelle Ladeverzeichnis wird beim Starten von imc FAMOS auf die Voreinstellung (Dialog "Optionen"/"Verzeichnisse") gesetzt. Es kann mit dem Befehl LDIR oder der Funktion SetOption umgesetzt werden. Ansonsten wird das Verzeichnis benutzt, das beim letzten Laden einer Datei angegeben wurde.

Wenn sich die Vorlagendatei (Erweiterung "*.aet") nicht im aktuell eingestellten Definitionsverzeichnis befindet, muß der komplette Pfadname inklusive Dateinamenserweiterung angegeben werden.

Diese Funktion prüft nicht die Gültigkeit des übergebenen Datei- und Vorlagennamen. Dies geschieht erst im korrespondierenden FileClose(), wenn die Datei tatsächlich geschrieben wird.

Das Verhalten der Funktion im Fehlerfall ist abhängig von der Einstellung "Keine Fehlerboxen bei Dateifunktionen" im "Optionen"/"Funktionen" -Dialog (bzw. der korrespondierenden Funktion SetOption("Func.ErrorBoxes", ..). Wenn diese Option ausgeschaltet ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und die Sequenzausführung abgebrochen (Standardverhalten der FAMOS-Funktionen). Wenn dagegen die Option eingeschaltet ist, wird eine 0 zurückgegeben und keine Fehlernachricht angezeigt.

Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() und FileErrCode?() angefragt werden.

FAMOS kann maximal 10 geöffnete Dateien gleichzeitig verwalten. Durch fehlende Aufrufe von FileClose() kann diese Zahl schnell erreicht werden. Sie erhalten dann eine entsprechende Fehlermeldung. Die Funktion FileResetAll() schließt alle momentan geöffneten Dateien. Dies kann nützlich sein, wenn z.B. beim Austesten von imc FAMOS-Sequenzen im Einzelschrittmodus Dateien offen geblieben sind, deren Identifikatoren Sie nicht mehr kennen.

Texte/Textfelder: Ab Version 2023 können bei FileObjWrite() auch Text- oder Textfeld-Variablen angegeben werden. Texte werden beim Speichern auf maximal 32767 Zeichen abgeschnitten.

Multithreading: Die von der Funktion zurückgegebene Datei-ID ist nur für den aktuellen Ausführungs-Thread gültig. Dateien werden am Thread-Ende (also z.B. am Ende einer mittels BEGIN_PARALLEL ausgeführten Sequenzfunktion) automatisch geschlossen.

Beispiele:

Ein Datensatz wird gefiltert und anschließend zusammen mit dem Ergebnis der Filterung in einer ASCII-Datei gespeichert. Das genaue Format wird durch die Exportvorlage "Vorlage #2" definiert, die vorher über "Extra / Optionen / Dateiformate / ASCII-Speichern" angelegt worden ist.

```
signal = ...
filtered =  FiltLP(signal, 0, 0, 6, 100)
fh = FileOpenAscii2("z:\dat\result.txt", "Template #2", 1)
IF (fh > 0)
   err = FileObjWrite(fh, signal)
   err = FileObjWrite(fh, filtered)
   err = FileClose(fh)
END
```

Die folgende Sequenz speichert die selektierten Variablen spaltenweise in einer ASCII-Datei ab. Anschließend wird eine zusätzliche Fußzeile an die Datei angehängt.

```
TxFileName = "c:\dat\protocol.txt"

fh = FileOpenAscii2(TxFileName, "Vorlage #2", 1)
IF (fh > 0)
   count = VarGetInit(1)
   n = 1
   WHILE (n <= count)
      TxVarName = VarGetName?(n)
      err = FileObjWrite(fh, <TxVarName>)
      n = n+1
   END
   err = FileClose(fh)
END

fh = FileOpenAscii(TxFileName, 2) ; Option 2: Anhängen
IF (fh > 0)
   err = FileLineWrite(fh, "Bearbeiter: Hans Müller", 0)
   err = FileClose(fh)
END
```

Siehe auch:

FileOpenAscii, FileObjWrite, FileClose, FileOpenXLS, FileOpenXLS2

FileOpenDSF

Eine Meßwertdatei im imc/FAMOS-Format wird geöffnet.

Deklaration:

```
FileOpenDSF ( TxDatei, EwModus ) -> EwDateiID
```

Parameter:

TxDatei	Name der Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet.
EwModus	Modus
	0 : Datei wird zum Lesen geöffnet.
	1 : Datei wird zum Schreiben geöffnet. Eine eventuell bestehende Datei gleichen Namens wird überschrieben.
	2 : Datei wird zum Schreiben geöffnet. Wenn die Datei schon existiert, wird der neue Inhalt angehängt.
EwDateiID	ID der geöffneten Datei bzw. Fehlercode
	0 : Fehler beim Öffnen der Datei. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.
	>0 : Der Identifikator für die geöffnete Datei. Dieser wird allen nachfolgenden Funktionen zur Bearbeitung dieser Datei als Parameter übergeben.

Beschreibung:

Eine Messwertdatei im imc/FAMOS-Format wird geöffnet. Falls kein voller Dateiname angegeben ist, wird im aktuellen Ladeverzeichnis gesucht.

Das aktuelle Ladeverzeichnis wird beim Starten von imc FAMOS auf die Voreinstellung (Dialog 'Optionen' / 'Verzeichnisse') gesetzt. Es kann mit dem Befehl LDIR oder der Funktion SetOption() umgesetzt werden. Ansonsten wird das Verzeichnis benutzt, das beim letzten Laden einer Datei angegeben wurde.

Der Identifikator der geöffneten Datei muss bei jedem folgenden Zugriff auf die Datei mit einer Dateifunktion angegeben werden. Er ist gültig bis zum Aufruf von FileClose().

Jede mit FileOpenDSF() geöffnete Datei muss durch einen Aufruf von FileClose() wieder geschlossen werden.

imc FAMOS kann maximal 10 geöffnete Dateien gleichzeitig verwalten. Durch fehlende Aufrufe von FileClose() kann diese Zahl schnell erreicht werden. Sie erhalten dann eine entsprechende Fehlermeldung.

Die Funktion FileResetAll() schließt alle momentan geöffneten Dateien. Dies kann nützlich sein, wenn z.B. beim Austesten von Sequenzen im Einzelschrittmodus Dateien offen geblieben sind, deren Identifikatoren Sie nicht mehr kennen.

Falls die Datei mit der Option 0 zum Lesen geöffnet werden soll, können weitere Flags mit [Modus] angegeben werden:

Addieren Sie +10:	Schnelles Laden. Bei einem späteren Aufruf der Funktion FileObjRead werden die Daten nicht kopiert, sondern als Verweis auf die Datei verwaltet. Nur sinnvoll, wenn Sie die geladenen Datensätze später nicht verändern wollen.
Addieren Sie +100:	Keine Datengruppen. Die in Datengruppen enthaltenen Datensätze oder Texte werden expandiert.

Weitere Optionen zur Kompatibilität mit DSF-Dateien, die von Programmen oder Messgeräten erzeugt wurden, die nicht vom imc stammen:

Addieren Sie +1000:	Offsetkorrektur. Notwendig, wenn Sie nach dem Laden eine Verschiebung des Datensatzes in y-Richtung feststellen.
Addieren Sie +2000:	Vertauschen von Tag und Monat beim Lesen der Triggerzeit. Wählen Sie diese Option, wenn Sie eine Fehlermeldung der Form 'Ungültiger Eintrag im |NT-Key' erhalten oder wenn bei der Triggerzeit des geladenen Datensatzes Tag und Monat vertauscht sind.
Addieren Sie +20000:	Diese Option kann verwendet werden, um Dateien zu laden, die bei der Messung nicht korrekt abgeschlossen wurden. FAMOS versucht dann, solche Dateien teilweise zu laden und die gültige Teilmenge an Messdaten zu lesen. Es wird empfohlen, die geladenen Daten anschließend gründlich zu prüfen. Diese Option wirkt nur auf Dateien im "imc3"-Dateiformat.

Falls die Datei mit der Option 1 oder 2 zum Schreiben geöffnet werden soll, können weitere Flags mit [Modus] angegeben werden:

Addieren Sie +10000:	Ab FAMOS 7.4 wird standardmäßig ein neues, verbessertes Dateiformat (Version 3) verwendet, welches von älteren imc Programmen (imc FAMOS 7.3, imc STUDIO 5.x) möglicherweise nicht gelesen werden kann. Mit dieser Option wird das 'alte' Dateiformat (Version 2) erzwungen.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der imc FAMOS-Funktionen. Wenn dagegen die Option gewählt wurde, wird eine 0 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.
- Multithreading: Die von der Funktion zurückgegebene Datei-ID ist nur für den aktuellen Ausführungs-Thread gültig. Dateien werden am

Thread-Ende (also z.B. am Ende einer mittels BEGIN_PARALLEL ausgeführten Sequenzfunktion) automatisch geschlossen.

Beispiele:

Eine Datei wird im imc/FAMOS-Format zum Lesen geöffnet. In einer Schleife über alle Objekte werden alle Texte in unter dem Originalnamen angelegt. Datensätze und Datengruppen in der Datei werden übergangen.

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 0)
IF idFile >= 1
   count = FileObjNum?(idFile)
   index = 1
   WHILE index <= count
      Typ = FileObjType?(idFile, index)
      IF Typ = 1
         TxName  = FileObjName?(idFile, index)
          <TxName> = FileObjRead(idFile, index)
      END
      index = index + 1
   END
   ret = FileClose(idFile)
END
```

Siehe auch:

FileLoad, FileSave, FileOpenFAS, FileOpenASCII, FileOpenXLS

FileOpenFAS

Eine Meßwertdatei in einem nutzerdefinierten Format wird geöffnet.

Deklaration:

```
FileOpenFAS ( TxDatei, TxFormat, EwModus ) -> EwDateiID
```

Parameter:

TxDatei	Name der Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet.
TxFormat	Spezifikation des Dateiformats.
EwModus	Modus
	0 : Datei wird zum Lesen geöffnet.
	100 : Datei wird zum Lesen geöffnet. Enthaltene Gruppen werden expandiert.
	1 : Datei wird zum Schreiben geöffnet. Eine eventuell bestehende Datei gleichen Namens wird überschrieben.
	2 : Datei wird zum Schreiben geöffnet. Wenn die Datei schon existiert, wird der neue Inhalt angehängt (sofern 'Anhängen' vom verwendeten Exportfilter unterstützt wird).
EwDateiID	ID der geöffneten Datei bzw. Fehlercode
	0 : Fehler beim Öffnen der Datei. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.
	>0 : Der Identifikator für die geöffnete Datei. Dieser wird allen nachfolgenden Funktionen zur Bearbeitung dieser Datei als Parameter übergeben.

Beschreibung:

Eine Datei in einem nutzerdefinierten Format (Datei-Assistent, Import/Export- Erweiterungsbibliothek oder abgeleitetes Importformat) wird zum Lesen oder Schreiben geöffnet.

Für die Spezifikation des Formats sind 3 Varianten möglich:

Import mit Format-Beschreibungsdatei (*.FAS, imc-Dateiassistent).

Das Format ist durch eine Import-Beschreibungsdatei (Importfilter) festgelegt, die mit dem imc- Dateiassistenten erstellt wurde. Eine solche Importfilter-Datei hat üblicherweise die Erweiterung ".fas".

<TxFormat> gibt dann den Namen der Formatbeschreibungsdatei an. Falls kein kompletter Pfadname angegeben ist, wird im voreingestellten Definitionsverzeichnis ("Extras"/"Optionen"/"Verzeichnisse") gesucht.

Mit dieser Variante können Messwertdateien nur gelesen, aber nicht geschrieben werden, <Modus> ist daher stets auf 0 zu setzen.

Import oder Export mit einer entsprechenden Erweiterungsbibliothek (*.DLL).

Die notwendige Funktionalität wird durch eine Import/Export-Erweiterungsbibliothek zur Verfügung gestellt. Solche Bibliotheken ("Dynamic link library", DLL) werden von imc oder unseren Partnern angeboten und integrieren sich nahtlos in die imc FAMOS-Bedienoberfläche und die Funktionsbibliothek. Damit ist eine flexible Anpassung von imc FAMOS an zusätzlich zu unterstützende Dateiformate möglich. Ein Beispiel ist die DLL zum Import und Export des MATLAB-Dateiformates, die im Standard-Lieferumfang von imc FAMOS enthalten ist.

Die Syntax von <TxFormat> lautet dann wie folgt:

```
"#DLLName|FormatName|Parameter"
```

- DLLName: gibt den Dateinamen der Erweiterungsbibliothek an, die Erweiterung ".DLL" kann weggelassen werden.
- FormatName: gibt den Namen des gewünschten Formats an (eine Erweiterungsbibliothek kann durchaus mehrere Formate unterstützen). Kann weggelassen werden, wenn die DLL nur ein einziges Format zur Verfügung stellt und keine weiteren Parameter benötigt.

- Parameter: dient zur optionalen Übergabe weiterer Parameter an die Bibliothek. Ob Parameter benötigt werden sowie deren Syntax hängt von der jeweiligen Erweiterungsbibliothek ab. Oft nicht benötigt, kann dann weggelassen werden.Suchen Sie ggf. nach dem Namen des verwendenden Formats in der Hilfe, wo die formatspezifische Verwendung des Parameters beschrieben wird.

Import mit einem abgeleiteten Importfilter

Der Import erfolgt durch ein abgeleitetetes Importfilter. Die Syntax des Parameters <TxFormat> ist dann wie folgt:

```
"$FormatName"
```

"FormatName" entspricht dem vom Anwender selbst vergebenen Namen, wie er z.B. im FAMOS-Dialog "Optionen"/"Importfilter" angezeigt wird.

Hinweis: Eine in der Definition des abgeleiteten Importfilters festgelegte Messungszugehörigkeit wird beim Laden ignoriert.

Laden einer Textvariablen aus einer Textdatei

Um den Inhalt einer Textdatei in eine Textvariable zu übertragen, geben Sie die Formatkennung "imc/Text" an.

Sie können zusätzlich die Zeichensatz-Kodierung (Code page) der Datei angeben. Standard ist die aktuelle Windows-Codepage.

"imc/Text"	Die aktuelle Windows-ANSI-Codepage wird angenommen.
"imc/Text/auto)"	Die Kodierung wird automatisch ermittelt. Dazu wird geprüft, ob die Datei mit einem definierten BOM (Byte order mark) beginnt. Erkannt werden die BOM für UTF-8 und UTF-16 ("Big endian" und "Little endian" byte order). Wenn kein solcher BOM erkannt wird, wird die aktuelle Windows-Codepage angenommen.
"imc/Text/UTF8"	UTF-8-Kodierung
"imc/Text/UTF16_LE"	UTF-16 LE (Byte-Reihenfolge "Little endian", niederwertiges Byte zuerst)
"imc/Text/UTF16_BE"	UTF-16 BE (Byte-Reihenfolge "Big endian", höherwertiges Byte zuerst)

Tipp: Verwenden Sie den Funktions-Assistent für die Parametrierung der Funktion. Sie können hier bequem in einer Liste aller installierten Dateifilter auswählen.

Beispiele:

```
FileOpenFAS("1.txt","#MyFormat.DLL", 0)
FileOpenFAS("1.txt","#MyFormat.DLL|Txt", 0)
FileOpenFAS("1.asc","#MyFormat.DLL|Asc|Head=y", 1)
FileOpenFAS("1.lec","$Lecroy #1", 0)
```

Import von MatLab-Dateien:

```
FileOpenFAS("1.mat","#MatLabImportExport.DLL", 0)
```

Export von MatLab-Dateien:

```
FileOpenFAS("1.mat","#MatLabImportExport.DLL|Matlab 4 Format", 1)
  ; oder
FileOpenFAS("1.mat","#MatLabImportExport.DLL|Matlab 5 Format", 1)
```

ASCII-Import-Assistent

Mit dem ASCII-Import-Assistenten definierte Import-Filter können ebenfalls mit der Funktion FileOpenFAS() verwendet werden.

```
FileOpenFAS("1.txt","#ImportAscii1.dll|FilterName", 0)
```

"FilterName" entspricht dem Namen, unter dem das Filter im ASCII-Import-Dialog gespeichert wurde und ist identisch mit dem angezeigten Formatnamen im "Datei laden"-Dialog.

- Falls für die Messwertdatei kein kompletter Pfadname angegeben ist, wird im aktuellen Ladeverzeichnis gesucht. Das aktuelle Ladeverzeichnis wird beim Starten von imc FAMOS auf die Voreinstellung (Dialog "Optionen"/ "Verzeichnisse") gesetzt. Es kann mit dem Befehl LDIR oder der Funktion SetOption() umgesetzt werden. Ansonsten wird das Verzeichnis benutzt, das beim letzten Laden einer Datei angegeben wurde.
- Der Identifikator der geöffneten Datei muss bei jedem folgenden Zugriff auf die Datei mit einer Dateifunktion angegeben werden. Er ist gültig bis zum Aufruf von FileClose().
- Jede mit FileOpenFAS() geöffnete Datei muss durch einen Aufruf von FileClose() wieder geschlossen werden.
- imc FAMOS kann maximal 10 geöffnete Dateien gleichzeitig verwalten. Durch fehlende Aufrufe von FileClose() kann diese Zahl schnell erreicht werden. Sie erhalten dann eine entsprechende Fehlermeldung.
- Die Funktion FileResetAll() schließt alle momentan geöffneten Dateien. Dies kann nützlich sein, wenn z. B. beim Austesten von imc FAMOS-Sequenzen im Einzelschrittmodus Dateien offen geblieben sind, deren Identifikatoren Sie nicht mehr kennen.
- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung "Keine Fehlerboxen bei Dateifunktionen" (einstellbar im "Optionen"/ "Funktionen"-Dialog oder mit der Funktion SetOption). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der imc FAMOS-Funktionen. Wenn dagegen die Option gewählt wurde, wird eine 0 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText? oder FileErrCode? abgefragt werden.
- Multithreading: Die von der Funktion zurückgegebene Datei-ID ist nur für den aktuellen Ausführungs-Thread gültig. Dateien werden am Thread-Ende (also z.B. am Ende einer mittels BEGIN_PARALLEL ausgeführten Sequenzfunktion) automatisch geschlossen.

Beispiele:

Eine Datei wird in einem nutzerdefinierten Format zum Lesen geöffnet. Das Format wird durch die Datei 'format1.fas' beschrieben. In einer Schleife über alle Objekte werden alle Datensätze in FAMOS unter dem Originalnamen angelegt. Texte und Datengruppen in der Datei werden übergangen.

```
idFile = FileOpenFAS("c:\imc\dat\xxx.dat", "c:\imc\def\format1.fas", 0)
IF idFile >= 1
   count = FileObjNum?(idFile)
   index = 1
   WHILE index <= count
      Typ = FileObjType?(idFile, index)
      IF Typ = 1
         TxName = FileObjName?(idFile, index)
          <TxName>= FileObjRead(idFile, index)
      END
```

```
      index = index + 1
   END
   ret = FileClose(idFile)
END
```

Die Variable 'MyChannel' wird im MatLab5-Format gespeichert.

```
MyChannel = ...
idFile = FileOpenFAS("z:\tmp\export.mat", "#MatlabImportExport.dll|Matlab 5 Format", 1)
IF idFile >= 1
   TxName  = FileObjWrite(idFile, MyChannel)
   ret=FileClose(idFile)
END
```

Siehe auch:

FileLoad, FileOpenASCII, FileOpenXLS2, FileXLSColumnRead, FileXLSCellRead

FileOpenXLS

Eine Excel-Tabelle im XLS-Format wird zum Lesen und/oder zellwenweisen Schreiben geöffnet.

Deklaration:

```
FileOpenXLS ( TxDatei, EwModus ) -> EwDateiID
```

Parameter:

TxDatei	Name der XLS-Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet.
EwModus	Modus
	0 : Datei wird nur zum Lesen geöffnet.
	1 : Datei wird zum Schreiben geöffnet. Eine eventuell bestehende Datei gleichen Namens wird überschrieben.
	2 : Datei wird zum Schreiben geöffnet. Wenn die Datei schon existiert, wird der neue Inhalt angehängt.
EwDateiID	ID der geöffneten Datei bzw. Fehlercode
	0 : Fehler beim Öffnen der Datei. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.
	>0 : Der Identifikator für die geöffnete Datei. Dieser wird allen nachfolgenden Funktionen zur Bearbeitung dieser Datei als Parameter übergeben.

Beschreibung:

Diese Funktion ermöglicht das Lesen und Schreiben von Dateien im EXCEL-XLS-Format. Nach dem erfolgreichen Öffnen der Datei mit dieser Funktion kann anschließend mit den Funktion FileXLSColumnRead(), FileXLSCellRead() oder FileXLSCellWrite() auf die Tabelle zugegriffen werden.

Mit dieser Funktion wird eine versteckte EXCEL-Instanz gestartet und diese per Fernsteuerung (der sogenannten OLE-Automation) aufgefordert, die Datei zu laden. Nachfolgende angewendete Lesefunktionen importieren die gewünschten Daten ebenfalls mittels EXCEL-Fernsteuerung.

Die Funktion kann also nur verwendet werden, wenn auf dem Rechner eine EXCEL-Version installiert ist (Versionen Excel95 .. Excel2016).

Jede mit FileOpenXLS() geöffnete Datei muß durch einen abschließenden Aufruf von FileClose() wieder geschlossen werden. Hierbei wird ggf. die geänderte Datei gespeichert und die versteckte Excel-Programminstanz wieder beendet.

Mit dieser Funktion kann immer nur eine EXCEL-Datei gleichzeitig geöffnet werden. Ist bei einem FileOpenXLS()-Aufruf eine zuvor geöffnete XLS-Datei noch geöffnet (also kein passender FileClose()-Aufruf erfolgt), wird die zuvor geöffnete Datei automatisch geschlossen.

imc FAMOS kann maximal 10 geöffnete Dateien gleichzeitig verwalten. Durch fehlende Aufrufe von FileClose() kann diese Zahl schnell erreicht werden. Sie erhalten dann eine entsprechende Fehlermeldung. Die Funktion FileResetAll() schließt alle momentan geöffneten Dateien. Dies kann nützlich sein, wenn z. B. beim Austesten von imc FAMOS-Sequenzen im Einzelschrittmodus Dateien offen geblieben sind, deren Identifikatoren Sie nicht mehr kennen.

FileXLSColumnRead(), FileXLSCellRead(), FileXLSCellWrite(), FileXLSSelectSheet() und FileClose() sind die einzigen Funktionen, die auf mit FileOpenXLS() geöffnete Dateien anwendbar sind.

Zum spaltenweisen Schreiben von XLS-Dateien können Sie die Funktion FileOpenXLS2() verwenden.

Falls kein voller Dateiname angegeben ist, wird im aktuellen Ladeverzeichnis gesucht. Das aktuelle Ladeverzeichnis wird beim Starten von imc FAMOS auf die Voreinstellung (Dialog "Optionen" /"Verzeichnisse") gesetzt. Es kann mit dem Befehl LDIR oder der Funktion SetOption() umgesetzt werden. Ansonsten wird das Verzeichnis benutzt, das beim letzten Laden einer Datei angegeben wurde.

Das Verhalten der Funktion im Fehlerfall ist abhängig von der Einstellung "Keine Fehlerboxen bei Dateifunktionen" im "Optionen"/"Funktionen" -Dialog (bzw. der korrespondierenden Funktion SetOption("Func.ErrorBoxes", ..). Wenn diese Option ausgeschaltet ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und die Sequenzausführung abgebrochen (Standardverhalten der FAMOS-Funktionen). Wenn dagegen die Option eingeschaltet ist, wird ein leerer Datensatz zurückgegeben und keine Fehlernachricht angezeigt.

Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() und FileErrCode?() angefragt werden.

Multithreading: Die von der Funktion zurückgegebene Datei-ID ist nur für den aktuellen Ausführungs-Thread gültig. Dateien werden am Thread-Ende (also z.B. am Ende einer mittels BEGIN_PARALLEL ausgeführten Sequenzfunktion) automatisch geschlossen.

Beispiele:

Ein XLS-Datei wird geöffnet. Diese enthält eine Tabelle mit 2 Spalten, die jeweils Wertepaare (Zeit, Messwert) enthalten. Die Werte beginnen in der 2. Zeile. In der 1. Zeile stehen der Name des Datensatzes und daneben die y-Einheit.

```
fh = FileOpenXLS("c:\dat\tabelle.xls", 0)
IF (fh > 0)
   x = FileXLSColumnRead(fh, 1, 2, 65336, 0)
   y = FileXLSColumnRead(fh, 2, 2, 65336, 0)
   TxName = FileXLSCellRead(fh, 1, 1, 0)
   TxUnit = FileXLSCellRead(fh, 2, 1, 0)
   err = FileClose(fh)
   data = XYof(x,y)
   ; 1.Wert der Zeitspur = Triggerzeit
```

```
   first_time = data[1].x
   SetTime(data, first_time)
   ; Zeitspur bzgl. Triggerzeit normieren
   data.x = data.x-first_time
   ; Einheit und Name setzen
   SetUnit(data, TxUnit, 1)
   RENAME data <TxName>
END
```

Ein Datensatz wird in einer XLS-Datei ab der 2. Zeile gespeichert. Anschließend wird in die erste Zeile der Mittelwert der gespeicherten Daten eingetragen.

```
signal = ...
fh = FileOpenXLS2("z:\dat\result.xls", "Vorlage #2", 1, 1, 2)
IF (fh > 0)
   err = FileObjWrite(fh, signal)
   err = FileClose(fh)
END
fh = FileOpenXLS("z:\dat\result.xls", 2)
IF (fh > 0)
   err = FileXLSCellWrite(fh, 1, 1, "Mean Value:")
   err = FileXLSCellWrite(fh, 2, 1, Mean(signal))
   err = FileClose(fh)
END
```

Siehe auch:

FileOpenASCII, FileOpenXLS2, FileXLSColumnRead, FileXLSCellRead, FileXLSCellWrite

FileOpenXLS2

Eine XLS-Datei wird zum spaltenweisen Schreiben von Datensätzen geöffnet. Die Formatspezifikation erfolgt mittels einer vordefinierten XLS-Exportvorlage.

Deklaration:

```
FileOpenXLS2 ( TxDatei, TxExportVorlage, EwModus, EwSpalte, EwZeile ) -> EwDateiID
```

Parameter:

TxDatei	Name der XLS-Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet.
TxExportVorlage	Name der zu verwendenden XLS-Exportvorlage.
EwModus	Modus
	1 : Datei wird zum Schreiben geöffnet. Eine eventuell bestehende Datei gleichen Namens wird komplett überschrieben.
	2 : Datei wird zum Schreiben geöffnet. Eine eventuell bestehende Datei gleichen Namens wird geladen und die zuzufügenden Datensätze in die bestehende Tabelle einkopiert, ansonsten wird die Originaldatei nicht geändert.
EwSpalte	Spaltennummer (1..256). Gibt zusammen mit dem nächsten Parameter die erste zu beschreibende Zelle an, links oben ist [1,1].
EwZeile	
EwDateiID	ID der geöffneten Datei bzw. Fehlercode
	0 : Fehler beim Öffnen der Datei. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.
	>0 : Der Identifikator für die geöffnete Datei. Dieser wird allen nachfolgenden Funktionen zur Bearbeitung dieser Datei als Parameter übergeben.

Beschreibung:

Diese Funktion ermöglicht das Schreiben von Dateien im EXCEL-XLS-Format, in denen Datensätze spaltenweise abgelegt werden. Die Formatspezifikation erfolgt mittels einer vordefinierten XLS-Exportvorlage.

Die Funktion kann nur verwendet werden, wenn auf dem Rechner eine EXCEL-Version installiert ist (Versionen Excel95.. Excel2016).

Mit dieser Funktion wird eine versteckte EXCEL-Instanz gestartet und diese per Fernsteuerung (der so genannten OLE-Automation) aufgefordert, eine neue Tabelle anzulegen bzw. die angegebene Datei zu laden.

Nachdem eine Datei mit dieser Funktion geöffnet wurde, können die zu speichernden Datensätze mit der Funktion FileObjWrite(..) zugefügt werden.

Es wird standardmäßig in das erste Tabellenblatt geschrieben. Mit der Funktion FileXLSSelectSheet() kann ein anderes Blatt zum Schreiben selektiert werden.

Jede mit FileOpenXLS2() geöffnete Datei muß durch einen abschließenden Aufruf von FileClose() wieder geschlossen werden. Hierbei wird dann auch die Datei gespeichert und die versteckte Excel-Programminstanz wieder beendet.

FileObjWrite(), FileXLSSelectSheet() und FileClose() sind die einzigen Funktionen, die auf mit FileOpenXLS2() geöffnete Dateien anwendbar sind.

Die konkrete Definition des Dateiformats (z.B. Spaltenköpfe, Skalierungsspalten, Zahlenformat..) erfolgt durch die angegebene XLS-Exportvorlage. Das Erstellen und die Verwaltung solcher Vorlagen erfolgt über den Menüpunkt "Extra / Optionen / Datei-Speichern / EXCEL"

Tipp: Verwenden Sie den Funktions-Assistent für die Parametrierung der Funktion. Sie können hier bequem in einer Liste aller XLS-Exportvorlagen auswählen.

Zum Import von XLS-Dateien können Sie die Funktion FileOpenXLS() verwenden.

Falls kein voller Dateiname angegeben ist, wird im aktuellen Ladeverzeichnis gesucht. Das aktuelle Ladeverzeichnis wird beim Starten von imc FAMOS auf die Voreinstellung (Dialog "Optionen" /"Verzeichnisse") gesetzt. Es kann mit dem Befehl LDIR oder der Funktion SetOption() umgesetzt werden. Ansonsten wird das Verzeichnis benutzt, das beim letzten Laden einer Datei angegeben wurde.

Diese Funktion prüft nicht die Gültigkeit des übergebenen Datei- und Vorlagennamen. Dies geschieht erst im korrespondierenden FileClose(), wenn die Datei tatsächlich geschrieben wird.

Wenn sich die Vorlagendatei (Erweiterung "*.aet") sich nicht im aktuell eingestellten Definitionsverzeichnis befindet, muß der komplette Pfadname inklusive Dateinamenserweiterung angegeben werden.

imc FAMOS kann maximal 10 geöffnete Dateien gleichzeitig verwalten. Durch fehlende Aufrufe von FileClose() kann diese Zahl schnell erreicht werden. Sie erhalten dann eine entsprechende Fehlermeldung. Die Funktion FileResetAll() schließt alle momentan geöffneten Dateien. Dies kann nützlich sein, wenn z. B. beim Austesten von imc FAMOS-Sequenzen im Einzelschrittmodus Dateien offen geblieben sind, deren Identifikatoren Sie nicht mehr kennen.

Das Verhalten der Funktion im Fehlerfall ist abhängig von der Einstellung "Keine Fehlerboxen bei Dateifunktionen" im "Optionen"/"Funktionen" -Dialog (bzw. der korrespondierenden Funktion SetOption("Func.ErrorBoxes", ..). Wenn diese Option ausgeschaltet ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und die Sequenzausführung abgebrochen (Standardverhalten der FAMOS-Funktionen). Wenn dagegen die

Option eingeschaltet ist, wird eine 0 zurückgegeben und keine Fehlernachricht angezeigt.

Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() und FileErrCode?() angefragt werden.

Texte/Textfelder: Ab Version 2023 können bei FileObjWrite() auch Text- oder Textfeld-Variablen angegeben werden. Texte werden beim Speichern auf maximal 32767 Zeichen abgeschnitten.

Multithreading: Die von der Funktion zurückgegebene Datei-ID ist nur für den aktuellen Ausführungs-Thread gültig. Dateien werden am Thread-Ende (also z.B. am Ende einer mittels BEGIN_PARALLEL ausgeführten Sequenzfunktion) automatisch geschlossen.

Beispiele:

Ein Datensatz wird gefiltert und anschließend zusammen mit dem Ergebnis der Filterung in einer XLS-Datei gespeichert. Das genaue Format wird durch die Exportvorlage "Vorlage #2" definiert, die vorher über "Extra / Optionen / Datei-Speichern / EXCEL" angelegt worden ist.

```
signal = ...
filtered =  FiltLP(signal, 0, 0, 6, 100)
fh = FileOpenXLS2("z:\dat\result.xls", "Template #2", 1, 1, 1)
IF (fh > 0)
   err = FileObjWrite(fh, signal)
   err = FileObjWrite(fh, filtered)
   err = FileClose(fh)
END
```

Die folgende Sequenz speichert die selektierten Variablen spaltenweise in einer XLS-Datei ab. Die Ausgabe beginnt in der 3. Spalte und 2. Zeile.

```
TxFileName = "c:\dat\protocol.xls"

fh = FileOpenXLS2(TxFileName, "Vorlage #2", 1, 3, 2)
IF (fh > 0)
   count = VarGetInit(1)
   n = 1
   WHILE (n <= count)
      TxVarName = VarGetName?(n)
      err = FileObjWrite(fh, <TxVarName>)
      n = n + 1
   END
   err = FileClose(fh)
END
```

Siehe auch:

FileOpenASCII, FileOpenASCII2, FileOpenXLS, FileXLSSelectSheet, FileObjWrite, FileClose

FileResetAll

Schließen aller geöffneten Dateien

Deklaration:

```
FileResetAll ( )
```

Parameter:

Beschreibung:

Alle mit einer FileOpen*-Funktion (z.B. FileOpenDSF() oder FileOpenFAS()) geöffneten Dateilisten werden geschlossen. Es erfolgt kein Schreiben von eventuell veränderten Dateilisten.

Die Funktion sollte nur verwendet werden, um beim Testen von Sequenzen oder Programmen wieder einen definierten Grundzustand herzustellen. Verwenden Sie sonst immer die Funktion FileClose().

FAMOS kann maximal 10 geöffnete Dateien gleichzeitig verwalten. Durch fehlende Aufrufe von FileClose() kann diese Zahl schnell erreicht werden. Sie erhalten dann eine entsprechende Fehlermeldung.

Die Funktion FileResetAll kann z.B. verwendet werden, wenn beim Austesten von Sequenzen im Einzelschrittmodus Dateien offen geblieben sind, deren Identifikatoren Sie nicht mehr kennen.

Multithreading: Die Funktion schließt nur Dateien, die im aktuellen Ausführungs-Thread geöffnet wurden.

Siehe auch:

FileClose, FileOpenDSF, FileOpenFAS, FileOpenASCII, FileOpenXLS

FileSave

Speichert die übergebenen Daten in einer Datei mit wählbarem Format.

Deklaration:

```
FileSave ( TxDatei, TxFormat, EwOptionen, Daten [, Daten2] [, Daten3] [, Daten4] [, Daten5] [, Daten6] ) -> EwErfolg
```

Parameter:

TxDatei	Name der Datei. Falls kein voller Pfadname angegeben wird, wird das voreingestellte Ladeverzeichnis verwendet.
TxFormat	Spezifikation des Dateiformats.
EwOptionen	Optionen
	0 : Standardverhalten
	1 : Übergebene Gruppen werden expandiert, d.h. gespeichert wird die Liste der enthaltenen Kanäle ohne Gruppenbezug. Diese Option ist beispielsweise hilfreich, wenn Sie in einer komplexen Auswertung viele Ergebnisvariablen erzeugen, die Sie zum Abschluss in eine Datei speichern möchten. Fügen Sie alle diese Variablen gleich bei der Erzeugung einer temporären Gruppe hinzu und übergeben diese dann zum Abschluss der Auswertung hier als Speicher-Parameter.
	10000 : Nur berücksichtigt, wenn das "imc/FAMOS"-Dateiformat verwendet wird. Ab FAMOS 7.4 wird standardmäßig ein neues, verbessertes Dateiformat (Version 3) verwendet, welches von älteren imc Programmen (imc FAMOS 7.3, imc STUDIO 5.x) möglicherweise nicht gelesen werden kann. Addieren Sie 10000, um das 'alte' Dateiformat (Version 2) zu erzwingen.
Daten	Zu speichernde Daten
Daten2	Weitere Daten (optional) (optional)
Daten3	Weitere Daten (optional) (optional)
Daten4	Weitere Daten (optional) (optional)
Daten5	Weitere Daten (optional) (optional)
Daten6	Weitere Daten (optional) (optional)
EwErfolg	Erfolg der Funktion (Optional)
	0 : Fehler beim Speichern der Datei. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.
	1 : Die Datei konnte erfolgreich gespeichert werden.

Beschreibung:

Tipp: Verwenden Sie den Funktions-Assistent für die Parametrierung der Funktion. Sie können hier bequem in einer Liste aller unterstützten Exportformate auswählen.

Für die Spezifikation des Formats sind mehrere Varianten möglich:

Speichern im Standard-imc/FAMOS-Format
Für <TxFormat> ist dann entweder ein leerer Text oder die Kennung "imc/FAMOS" anzugeben.

Falls der übergebene Dateiname keine Erweiterung besitzt, wird automatisch ".dat" angehängt.

Beim Speichern eines einzelnen Elementes aus einer Gruppe geht die Gruppeninformation verloren.

Alternative Funktion: FileOpenDSF().

Speichern einer Textvariablen in eine Textdatei
Um den Inhalt einer Textvariablen bzw. eines Textfeldes in eine Textdatei zu speichern, geben Sie die Formatkennung "imc/Text" an. Es kann nur eine Variable pro Datei gespeichert werden, die Parameter Daten2 bis Daten6 sind also wegzulassen.

Bei einem Textfeld wird jedes Element als eine Zeile der Datei gespeichert. Der Zeilenumbruch wird jeweils durch eine 'CarriageReturn' (ASCII 13)/'LineFeed' (ASCII 10) - Zeichenkombination gewährleistet. Nach der letzten Zeile wird kein Zeilenumbruch angehängt.

Falls der übergebene Dateiname keine Erweiterung besitzt, wird automatisch ".txt" angehängt.

Export mit einer entsprechenden Erweiterungsbibliothek (*.DLL).
Die Syntax von <TxFormat> lautet dann wie folgt:

```
"#DLLName|FormatName|Parameter"
```

- DLLName: gibt den Dateinamen der Erweiterungsbibliothek an, die Erweiterung ".DLL" kann weggelassen werden.
- FormatName: gibt den Namen des gewünschten Formats an (eine Erweiterungsbibliothek kann durchaus mehrere Formate unterstützen). Kann weggelassen werden, wenn die DLL nur ein einziges Format zur Verfügung stellt und keine weiteren Parameter benötigt.
- Parameter: dient zur optionalen Übergabe weiterer Parameter an die Bibliothek. Ob Parameter benötigt werden sowie deren Syntax hängt von der jeweiligen Erweiterungsbibliothek ab. Oft nicht benötigt, kann dann weggelassen werden.Suchen Sie ggf. nach dem Namen des

verwendenden Formats in der Hilfe, wo die formatspezifische Verwendung des Parameters beschrieben wird.

Im Lieferumfang von FAMOS sind bereits eine Reihe von Erweiterungsbibliotheken vorhanden, die verbreitete Dateiformate erzeugen können.

Beispiel: Export von 3 Datensätzen in eine MatLab-Datei:

```
FileSave("1.mat","#MatLabImportExport|Matlab 5 Format", 0, Kanal1, Kanal2, Kanal3)
```

Alternative Funktion: FileOpenFAS().

Export mit einer ASCII-Exportvorlage
Die übergebenen Datensätze werden spaltenweise in eine ASCII-Datei geschrieben. Die Formatspezifikation erfolgt mittels einer vordefinierten ASCII-Exportvorlage, welche die konkrete Definition des Dateiformats (z.B. Spaltenköpfe, Skalierungsspalten, Zahlenformat..) enthält. Das Erstellen und die Verwaltung solcher Vorlagen erfolgt über den FAMOS-Menüpunkt "Extra / Optionen / Datei Speichern/Export / ASCII".

Die Syntax von <TxFormat> lautet dann wie folgt:

```
"[ASC] ExportVorlagenName"
```

Alternative Funktion: FileOpenASCII2().

Export mit einer EXCEL-Exportvorlage
Die übergebenen Datensätze werden spaltenweise in eine EXCEL-Datei geschrieben. Die Formatspezifikation erfolgt mittels einer vordefinierten EXCEL-Exportvorlage, welche die konkrete Definition des Dateiformats (z.B. Spaltenköpfe, Skalierungsspalten) enthält. Das Erstellen und die Verwaltung solcher Vorlagen erfolgt über den FAMOS-Menüpunkt "Extra / Optionen / Datei Speichern/Export / EXCEL".

Die Syntax von <TxFormat> lautet dann wie folgt:

```
"[XLS] ExportVorlagenName"
```

Mit dieser Funktion wird eine versteckte EXCEL-Instanz gestartet und diese per Fernsteuerung (der OLE-Automation) aufgefordert, eine neue Tabelle zu füllen und zu speichern. Die Funktion kann also nur ausgeführt werden, wenn auch eine unterstützte Version von EXCEL installiert ist. Die Daten werden immer in das erste Tabellenblatt, beginnend in der 1.Zeile und 1. Spalte, geschrieben.

Alternative Funktion: FileOpenXLS2().

Sonderformat "ByteBlob": Uninterpretierte Binär-Daten.
Der Inhalt der Variablen (nur die eigentlichen Daten) wird binär und uninterpretiert in eine Datei geschrieben. Geben Sie dafür die Formatkennung "imc/ByteBlob" an.

Das Format kann nur für einkomponentige, unstrukturierte Datensätze mit einfachen Datenformaten (reell, ganzzahlig, aber nicht digital) oder TimeStampASCII verwendet werden. Wird im Allgemeinen nur für Spezialanwendungen benötigt, z.B. im Zusammenspiel mit dem FAMOS-Datenbankkit, um beliebige Dateien (z.B. PDF, Bilder, Videos) zusammen mit den eigentlichen Messdaten in Datenbanken ablegen und wieder auslesen zu können.

- Die jeweils aufgeführte alternative Funktion zum Speichern im jeweiligen Format ist in der Anwendung komplexer, bietet aber zusätzliche Möglichkeiten (z.B. beliebig viele Datensätze, Anhängen an existierende Dateien).
- Wenn sich die Vorlagendatei (Erweiterung "*.aet") beim ASCII- bzw. EXCEL-Export nicht im aktuell eingestellten Standard-Definitionsverzeichnis befindet, muss der komplette Pfadname angegeben werden.
- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im "Optionen"/ "Funktionen"-Dialog oder mit der Funktion SetOption). Wenn diese Option ausgeschaltet ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine 0 zurückgegeben und keine Fehlermitteilung angezeigt. Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.

Beispiele:

Die Variablen 'MyChannel' bis 'MyChannel3' werden im FAMOS-Format in der Datei 'result.dat' gespeichert.

```
FileSave("z:\tmp\result", "", 0, MyChannel1, MyChannel2, MyChannel3)
```

Der Kommentar des Datensatzes 'MyChannel' wird in einer Textdatei gespeichert.

```
FileSave("z:\tmp\comment.txt", "imc/Text", 0, Comm?(MyChannel1))
```

Die Gruppe 'result' enthält die Kanäle 'c1' bis 'c3' und wird im EXCEL-Format gespeichert. Die Gruppe wird expandiert, die Datei enthält die Kanäle c1,c2,c3 ohne Gruppenbezug.

```
FileSave("z:\tmp\c.xls", "[XLS] XLSX, individual scaling", 1, result)
```

Speichern einer Variable im MatLab4-Format.

```
FileSave("z:\tmp\export.mat", "#MatlabImportExport|Matlab 4 Format", 0, MyChannel)
```

Siehe auch:

FileOpenDSF, FileOpenASCII2, FileOpenXLS2, FileLoad

FileSetComm

Die Funktion setzt den Dateikommentar in einer zum Schreiben geöffneten Datei.

Deklaration:

```
FileSetComm ( EwFileID, TxComment ) -> EwStatus
```

Parameter:

EwFileID	ID der geöffneten Datei, die beim Öffnen der Datei mit den Funktionen FileOpenDSF(), FileOpenASCII2() oder FileOpenXLS2() zurückgegeben wurde.
TxComment	Neuer Kommentar für die Datei.
EwStatus	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt.
	-1 : Fehler beim Ausführen der Funktion.

Beschreibung:

Die Funktion setzt den Dateikommentar einer geöffneten Datei. Die Datei muss zum Schreiben geöffnet sein, die Funktion FileOpenDSF() muss also mit der Option 1 oder 2 verwendet worden sein. Der neue Kommentar wird intern gemerkt, tatsächlich geschrieben wird die geänderte Datei erst beim Aufruf von FileClose().

Beim Schreiben per Exportvorlage (FileOpenASCII2() / FileOpenXLS2()) wird der Platzhalter %FILECOMMENT% in der Exportvorlagen-Definition entsprechend belegt.

- Das Verhalten der Funktion im Fehlerfall ist abhängig von der globalen Voreinstellung 'Keine Fehlerboxen bei Dateifunktionen' (einstellbar im 'Optionen'/ 'Funktionen'-Dialog oder mit der Funktion SetOption()). Wenn diese Option nicht gewählt ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und der Rückgabewert nicht erzeugt. Dies entspricht dem Standardverhalten der FAMOS-Funktionen. Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlermitteilung angezeigt.
- Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() oder FileErrCode?() abgefragt werden.
- Die Abfrage und Auswertung des Rückgabewertes ist war nicht zwingend notwendig, wird aber empfohlen.

Beispiele:

Eine FAMOS-Datei "xxx.dat" wird zum Lesen und Schreiben im imc FAMOS-Format geöffnet. Der Kommentar wird abgefragt, falls er nicht vorhanden ist, wird ein Kommentar eingetragen und die Datei wieder geschlossen:

```
fileID = FileOpenDSF("c:\imc\dat\xxx.dat", 2)
TxComm = FileComm?(fileID)
IF TLeng(TxComm) = 0
   status = FileSetComm(fileID, "checked")
END
status = FileClose(fileID)
```

Siehe auch:

FileOpenDSF, FileComm?, FileOpenASCII2, FileOpenXLS2

FileXLSCellRead

Eine Zelle einer EXCEL-Tabelle (XLS-Format) wird ausgelesen (als Text).

Deklaration:

```
FileXLSCellRead ( EwFileID, EwSpalte, EwZeile, Null ) -> TxInhalt
```

Parameter:

EwFileID	ID der geöffneten XLS-Datei, die beim Öffnen der Datei mit dem Befehl FileOpenXLS() zurückgegeben wurde.
EwSpalte	Spaltennummer der zu lesenden Spalte (1..256 bzw. 1..16384 ab Excel 2007).
EwZeile	Zeilennummer der zu lesenden Zelle. Die oberste Zeile hat die Nummer 1. Die letzte (maximal mögliche) Zeile hat die Nummer 1048576 (ab Excel 2007) bzw. 65536 in älteren Versionen.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
TxInhalt	Inhalt der Zelle (als Text).

Beschreibung:

Diese Funktion liest den Inhalt einer Zelle in einer EXCEL-Tabelle im XLS-Format und gibt diesen als Text zurück.

Die Datei muß vorher mit der Funktion FileOpenXLS() geöffnet worden sein.

Es wird standardmäßig auf das erste Tabellenblatt zugegriffen. Wenn ein anderes Blatt gelesen werden soll, ist vorher die Funktion FileXLSSelectSheet() entsprechend aufzurufen.

Zum Lesen von numerischen Werten sollte immer die Funktion FileXLSColumnRead() verwendet werden.

Das Verhalten der Funktion im Fehlerfall ist abhängig von der Einstellung "Keine Fehlerboxen bei Dateifunktionen" im "Optionen"/"Funktionen" -Dialog (bzw. der korrespondierenden Funktion SetOption("Func.ErrorBoxes", ..). Wenn diese Option ausgeschaltet ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und die Sequenzausführung abgebrochen (Standardverhalten der FAMOS-Funktionen). Wenn dagegen die Option eingeschaltet ist, wird ein leerer Text zurückgegeben und keine Fehlernachricht angezeigt.

Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() und FileErrCode?() angefragt werden.

Beispiele:

Ein XLS-Datei wird geöffnet. Diese enthält eine Tabelle mit 2 Spalten, die jeweils Wertepaare (Zeit, Messwert) enthalten. Die Werte beginnen in der 2. Zeile. In der 1. Zeile stehen der Name des Datensatzes und daneben die y-Einheit.

```
fh = FileOpenXLS("c:\dat\tabelle.xls", 0)
IF (fh > 0)
   x = FileXLSColumnRead(fh, 1, 2, 1048575, 0)
   y = FileXLSColumnRead(fh, 2, 2, 1048575, 0)
   TxName = FileXLSCellRead(fh, 1, 1, 0)
   TxUnit = FileXLSCellRead(fh, 2, 1, 0)
   err = FileClose(fh)
   data = XYof(x,y)
   ; 1.Wert der Zeitspur = Triggerzeit
   first_time = data[1].x
   SetTime(data, first_time)
   ; Zeitspur bzgl. Triggerzeit normieren
   data.x = data.x-first_time
   ; Einheit und Name setzen
   SetUnit(data, TxUnit, 1)
   RENAME data <TxName>
END
```

Siehe auch:

FileOpenXLS, FileXLSColumnRead, FileClose

FileXLSCellWrite

Eine Zelle einer EXCEL-Tabelle (XLS-Format) wird beschrieben.

Deklaration:

```
FileXLSCellWrite ( EwFileID, EwSpalte, EwZeile, Inhalt ) -> EwStatus
```

Parameter:

EwFileID	ID der geöffneten XLS-Datei, die beim Öffnen der Datei mit dem Befehl FileOpenXLS() zurückgegeben wurde.
EwSpalte	Spaltennummer der zu schreibenden Zelle (1..256).
EwZeile	Zeilennummer der zu schreibenden Zelle. Die oberste Zeile hat die Nummer 1.
Inhalt	Zu schreibender Inhalt. Es ist entweder ein numerischer Einzelwert (bzw. Datensatz der Länge 1) oder Text erlaubt.
EwStatus	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt.
	-1 : Funktion fehlgeschlagen. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.

Beschreibung:

Diese Funktion setzt den Inhalt einer Zelle in einer EXCEL-Tabelle im XLS-Format. Es kann entweder ein Text oder ein einzelner numerischer Wert übertragen werden.

Die Datei muß vorher mit der Funktion FileOpenXLS() im Schreib-Modus geöffnet worden sein.

Zum Schreiben von längeren Datenreihen in eine Tabellenspalte sind die Funktionen FileOpenXLS2()/FileObjWrite() einem wiederholten Aufruf dieser Funktion vorzuziehen.

Es wird standardmäßig auf das erste Tabellenblatt zugegriffen. Wenn in ein anderes Blatt geschrieben werden soll, ist vorher die Funktion FileXLSSelectSheet() entsprechend aufzurufen.

Das Verhalten der Funktion im Fehlerfall ist abhängig von der Einstellung "Keine Fehlerboxen bei Dateifunktionen" im "Optionen"/"Funktionen" -Dialog (bzw. der korrespondierenden Funktion SetOption("Func.ErrorBoxes", ..). Wenn diese Option ausgeschaltet ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und die Sequenzausführung abgebrochen (Standardverhalten der FAMOS-Funktionen). Wenn dagegen die Option eingeschaltet ist, wird eine -1 zurückgegeben und keine Fehlernachricht angezeigt.

Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() und FileErrCode?() angefragt werden.

Beispiele:

Ein Datensatz wird in einer XLS-Datei ab der 2. Zeile gespeichert. Anschließend wird in die erste Zeile der Mittelwert der gespeicherten Daten eingetragen.

```
data = ...
fh = FileOpenXLS2("z:\dat\result.xls", "Template #2", 1, 1, 2)
IF (fh > 0)
   err = FileObjWrite(fh, data)
   err = FileClose(fh)
END
fh = FileOpenXLS("z:\dat\result.xls", 2)
IF (fh > 0)
   err = FileXLSCellWrite(fh, 1, 1, "Mean Value:")
   err = FileXLSCellWrite(fh, 2, 1, Mean(data))
   err = FileClose(fh)
END
```

Siehe auch:

FileOpenXLS, FileXLSCellRead, FileClose

FileXLSColumnRead

Eine Spalte einer EXCEL-Tabelle (XLS-Format) wird ausgelesen.

Deklaration:

```
FileXLSColumnRead ( EwFileID, EwSpalte, EwZeile, EwMaxAnzahl, Null ) -> Daten
```

Parameter:

EwFileID	ID der geöffneten XLS-Datei, die beim Öffnen der Datei mit dem Befehl FileOpenXLS() zurückgegeben wurde.
EwSpalte	Spaltennummer der zu lesenden Spalte (1..256 bzw. 1..16384 ab Excel 2007).
EwZeile	Zeilennummer, in der das Lesen beginnen soll. Die oberste Zeile hat die Nummer 1. Die letzte (maximal mögliche) Zeile hat die Nummer 1048576 (ab Excel 2007) bzw. 65536 in älteren Versionen.
EwMaxAnzahl	Maximale Anzahl der zu lesenden Werte.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
Daten	Inhalt der Spalte (als Datensatz).

Beschreibung:

Diese Funktion realisiert das spaltenweise Lesen von EXCEL-Tabellen im XLS-Format.

Die Datei muß vorher mit der Funktion FileOpenXLS() geöffnet worden sein.

Der erste Wert des erzeugten Datensatzes wird der durch die Parameter [Spalte] und [Zeile] festgelegten Zelle entnommen. Danach wird das Lesen nach unten fortgesetzt, bis entweder die erste Zelle gefunden wird, die keinen numerischen Wert enthält, oder aber die maximale Zahl der zu lesenden Werte (entsprechend dem Parameter [MaxAnzahl]) erreicht ist.

Es wird standardmäßig auf das erste Tabellenblatt zugegriffen. Wenn ein anderes Blatt gelesen werden soll, ist vorher die Funktion FileXLSSelectSheet() entsprechend aufzurufen.

Wenn die zu lesende Spalte als Datum/Uhrzeit formatiert ist, werden die Werte vom EXCEL-Zeitformat in das imc-Zeitformat konvertiert. Die Werte können dann z.B. mit der Funktion TimeToText() in einen lesbaren Text konvertiert werden oder auch direkt als X-Komponente eines XY-Datensatzes zugewiesen werden.

Das Datenformat des Ergebnisses ist immer Reell, 8Byte (Double).

Zum Lesen von Text kann die Funktion FileXLSCellRead() verwendet werden.

Das Verhalten der Funktion im Fehlerfall ist abhängig von der Einstellung "Keine Fehlerboxen bei Dateifunktionen" im "Optionen"/"Funktionen" -Dialog (bzw. der korrespondierenden Funktion SetOption("Func.ErrorBoxes", ..). Wenn diese Option ausgeschaltet ist, wird eine Ausgabebox mit der Fehlermeldung angezeigt und die Sequenzausführung abgebrochen (Standardverhalten der FAMOS-Funktionen). Wenn dagegen die Option eingeschaltet ist, wird ein leerer Datensatz zurückgegeben und keine Fehlernachricht angezeigt.

Ggf. kann die Fehlerursache mit den Funktionen FileErrText?() und FileErrCode?() angefragt werden.

Beispiele:

Ein XLS-Datei wird geöffnet. Diese enthält eine Tabelle mit 2 Spalten, die jeweils Wertepaare (Zeit, Messwert) enthalten. Die Werte beginnen in der 2. Zeile. In der 1. Zeile stehen der Name des Datensatzes und daneben die y-Einheit.

```
fh = FileOpenXLS("c:\dat\tabelle.xls", 0)
IF (fh > 0)
   x = FileXLSColumnRead(fh, 1, 2, 1048575, 0)
   y = FileXLSColumnRead(fh, 2, 2, 1048575, 0)
   TxName = FileXLSCellRead(fh, 1, 1, 0)
   TxUnit = FileXLSCellRead(fh, 2, 1, 0)
   err = FileClose(fh)
   data = XYof(x,y)
   ; 1.Wert der Zeitspur = Triggerzeit
   first_time = data[1].x
   SetTime(data, first_time)
   ; Zeitspur bzgl. Triggerzeit normieren
   data.x = data.x - first_time
   ; Einheit und Name setzen
   SetUnit(data, TxUnit, 1)
   RENAME data <TxName>
END
```

Siehe auch:

FileOpenXLS, FileXLSCellRead, FileClose

FileXLSSelectSheet

Ein Tabellenblatt einer EXCEL-Mappe (XLS-Format) wird für nachfolgende Lese- /Schreibzugriffe selektiert.

Deklaration:

```
FileXLSSelectSheet ( EwFileID, EwTabellenindex ) -> EwStatus
```

Parameter:

EwFileID	ID der geöffneten XLS-Datei, die beim Öffnen der Datei mit dem Befehl FileOpenXLS() oder FileOpenXLS2() zurückgegeben wurde.
EwTabellenindex	Index des zu selektierenden Tabellenblattes (1..).
EwStatus	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt.
	-1 : Funktion fehlgeschlagen. Die Fehlerursache kann mit den Funktionen FileErrCode?() oder FileErrText?() erfragt werden.

Beschreibung:

Lesen und zellenweises Schreiben:

Diese Funktion wählt das Tabellenblatt aus, auf das bei nachfolgenden Lesezugriffen mit FileXLSCellRead(), FileXLSColumnRead() oder FileXLSCellWrite() zugegriffen werden soll. Die Datei muß vorher mit der Funktion FileOpenXLS() mit dem Modus 'Nur Lesen' oder 'Schreiben/Anhängen' geöffnet worden sein.

Spaltenweises Schreiben:

Diese Funktion wählt das Tabellenblatt aus, in dem die mit FileObjWrite() geschriebenen Werte abgelegt werden sollen. Die Datei muß vorher mit der Funktion FileOpenXLS2() mit der Option 'Kombinieren' (Parameter 'Modus' = 2) geöffnet worden sein. Das hier spezifizierte Tabellenblatt muss in der geöffneten Datei bereits vorhanden sein, es wird also kein neues Blatt angelegt.

Pro FileOpenXLS2()/FileClose()-Block ist nur ein FileXLSSelectSheet()-Aufruf zulässig. Wenn in einer EXCEL-Datei mehrere Seiten beschrieben werden sollen, ist also für jede Seite ein separater Open/Close-Block notwendig.

Beispiele:

Ein XLS-Datei mit 3 Tabellenblättern wird geöffnet. Jedes Tabellenblatt enthält 2 Spalten, die jeweils Wertepaare (Zeit, Messwert) enthalten. Die Werte beginnen in der 2. Zeile. In der 1. Zeile steht der Name des Datensatzes.

```
fh = FileOpenXLS("c:\dat\tabelle.xls", 0)
IF (fh > 0)
   I = 1
   WHILE I <= 3
      FileXLSSelectSheet(fh, I)
      x = FileXLSColumnRead(fh, 1, 2, 65336, 0)
      y = FileXLSColumnRead(fh, 2, 2, 65336, 0)
      TxName = FileXLSCellRead(fh, 1, 1, 0)
      <TxName> = XYof(x,y)
      I = I + 1
   END
   err = FileClose(fh)
END
```

Eine vorbereitete EXCEL-Datei enthält 3 Tabellenblätter. In das 2. Blatt sollen die Werte des Datensatzes "Channel1" eingetragen werden, in das 3. Blatt das Ergebnis der FFT des Datensatzes. Die Vorlage wird zunächst in das Ausgabeverzeichnis umkopiert und dann entsprechend beschrieben.

```
TxOutputFileName = "c:\out\result1.xls"
Channel1= ...
FFT_Channel1 = FFT(Channel1)
res = FsCopyFile("c:\templates\result.xls", TxOutputFileName, 2, 0)
fh = FileOpenXLS2(TxOutputFileName, "XLS Template Data", 2, 1, 1)
IF (fh > 0)
   err = FileXLSSelectSheet(fh, 2)
   err = FileObjWrite(fh, Channel1)
   err = FileClose(fh)
END
fh = FileOpenXLS2(TxOutputFileName, "XLS Template FFT", 2, 1, 1)
IF (fh > 0)
   err = FileXLSSelectSheet(fh, 3)
   err = FileObjWrite(fh, FFT_Channel1)
   err = FileClose(fh)
END
```

Siehe auch:

FileOpenXLS, FileXLSColumnRead, FileClose

FiltBP

Bandpass-Filter

Deklaration:

```
FiltBP ( Daten, EwCharakteristik, EwParameter, EwOrdnung, EwFrequenzUnten, EwFrequenzOben ) -> Filtrat
```

Parameter:

Daten	Zu filternder Datensatz [ND].
EwCharakteristik	Filtercharakteristik
	0 : Butterworth
	1 : Bessel
	2 : Tschebyschew
	3 : Kritische Dämpfung
EwParameter	Bei Tschebyschew-Charakteristik die gewünschte Welligkeit (0..3) in dB. Sonst auf 0 zu setzen.
EwOrdnung	Ordnung des Filters. Für Bessel im Bereich 1..40, sonst 1..100.
EwFrequenzUnten	Gewünschte untere Grenzfrequenz in Hz.
EwFrequenzOben	Gewünschte obere Grenzfrequenz in Hz.
Filtrat	Gefilteter Datensatz.

Beschreibung:

Der übergebene Datensatz wird einer Bandpassfilterung unterzogen.

Aus den übergebenen Parametern werden die Filter-Koeffizienten mit bilinearer Transformation berechnet.

Für eine genaue Filterung sollten beide Grenzfrequenzen deutlich unterhalb der halben Abtastfrequenz des Ausgangssignals liegen. Je näher die Frequenzen an der Abtastfrequenz liegen, um so ungenauer wird der Amplitudengang des Filters.

Beispiele:

```
BandPassFiltered = FiltBP(signal, 2, 1, 10, 100, 10000)
```

Der Datensatz wird einer Bandpassfilterung mit Tschebyschew-Charakteristik unterzogen. Die Welligkeit soll 1dB betragen. Es wird ein Filter 10.Ordnung benutzt. Die Grenzfrequenzen liegen bei 100Hz und 10kHz.

Siehe auch:

FiltBS, FiltHP, FiltLP, DFilt, FiltBpZ

FiltBpZ

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Bandpass-Filter ohne Phasenverschiebung

Deklaration:

```
FiltBpZ ( Eingangskanal, Charakteristik, Parameter, Ordnung, Untere Grenzfrequenz, Obere Grenzfrequenz ) -> Ergebnis
```

Parameter:

Eingangskanal	Der zu filternde Zeitdatensatz, die Zeit in Sekunden skaliert.
Charakteristik	Filter-Charakteristik
	0 : Butterworth
	1 : Bessel
	2 : Tschebyschew
	3 : Kritische Dämpfung
Parameter	Bei Tschebyschew-Charakteristik die Welligkeit in dB (0..3). Sonst auf 0 zu setzen.
Ordnung	Ordnung des Filters: 4, 8, 12, 16, 20
Untere Grenzfrequenz	Untere Grenzfrequenz in Hz
Obere Grenzfrequenz	Obere Grenzfrequenz in Hz
Ergebnis	Gefilterter Datensatz

Beschreibung:

Der Datensatz wird einmal vorwärts, einmal rückwärts gefiltert. Dabei wird aber ein Filter benutzt, das zwar die vorgegebene Charakteristik, aber als Ordnung nur die Hälfte der angegebenen Ordnung hat.

Der Amplituden-Frequenzgang ist so normiert, dass bei der Grenzfrequenz eine Dämpfung von 3dB vorliegt.

Aus den übergebenen Parametern werden die Filter-Koeffizienten mit bilinearer Transformation berechnet.

Für eine sinnvolle Filterung sollte die Grenzfrequenz deutlich unterhalb der halben Abtastfrequenz des Ausgangssignals liegen. Je näher die Frequenz an der Abtastfrequenz liegt, um so ungenauer wird der Amplitudengang des Filters.

Das Filter ist nicht kausal. Damit gilt die zeitliche Reihenfolge von Ursache und Wirkung nicht mehr.

Einschwingvorgänge gibt es in beide Richtungen, insbesondere auch nicht nur zu Beginn, sondern auch am Ende des Datensatzes.

Die Phase hat idealerweise den Wert null. Praktisch trifft das aber nur auf den eingeschwungenen Zustand zu, also nicht auf die Randbereiche.

Beispiele:

```
f = FiltBpZ ( Vibration, 0, 0, 4, 45, 55 )
```

Ein Butterworth-Bandpass-Filter 4. Ordnung mit einem Frequenzbereich von 45Hz bis 55Hz wird berechnet.

Siehe auch:

FiltLpZ, FiltBp

FiltBS

Bandsperre-Filter

Deklaration:

```
FiltBS ( Daten, EwCharakteristik, EwParameter, EwOrdnung, EwFrequenzUnten, EwFrequenzOben ) -> Filtrat
```

Parameter:

Daten	Zu filternder Datensatz [ND].
EwCharakteristik	Filtercharakteristik
	0 : Butterworth
	1 : Bessel
	2 : Tschebyschew
	3 : Kritische Dämpfung
EwParameter	Bei Tschebyschew-Charakteristik die gewünschte Welligkeit (0..3) in dB. Sonst auf 0 zu setzen.
EwOrdnung	Ordnung des Filters. Für Bessel im Bereich 1..40, sonst 1..100.
EwFrequenzUnten	Gewünschte untere Grenzfrequenz in Hz.
EwFrequenzOben	Gewünschte obere Grenzfrequenz in Hz.
Filtrat	Gefilteter Datensatz.

Beschreibung:

Der übergebene Datensatz wird einer Bandsperrenfilterung unterzogen.

Aus den übergebenen Parametern werden die Filter-Koeffizienten mit bilinearer Transformation berechnet.

Für eine genaue Filterung sollten beide Grenzfrequenzen deutlich unterhalb der halben Abtastfrequenz des Ausgangssignals liegen. Je näher die Frequenzen an der Abtastfrequenz liegen, um so ungenauer wird der Amplitudengang des Filters.

Beispiele:

```
BandStopFiltered = FiltBS(signal, 2, 1, 10, 100, 10000)
```

Der Datensatz wird einer Bandsperrenfilterung mit Tschebyschew-Charakteristik unterzogen. Die Welligkeit soll 1dB betragen. Es wird ein Filter 10.Ordnung benutzt. Die Grenzfrequenzen liegen bei 100Hz und 10kHz.

Siehe auch:

FiltBP, FiltHP, FiltTP, DFilt, FiltBsZ

FiltBsZ

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Bandsperre-Filter ohne Phasenverschiebung

Deklaration:

```
FiltBsZ ( Eingangskanal, Charakteristik, Parameter, Ordnung, Untere Grenzfrequenz, Obere Grenzfrequenz ) -> Ergebnis
```

Parameter:

Eingangskanal	Der zu filternde Zeitdatensatz, die Zeit in Sekunden skaliert.
Charakteristik	Filter-Charakteristik
	0 : Butterworth
	1 : Bessel
	2 : Tschebyschew
	3 : Kritische Dämpfung
Parameter	Bei Tschebyschew-Charakteristik die Welligkeit in dB (0..3). Sonst auf 0 zu setzen.
Ordnung	Ordnung des Filters: 4, 8, 12, 16, 20
Untere Grenzfrequenz	Untere Grenzfrequenz in Hz
Obere Grenzfrequenz	Obere Grenzfrequenz in Hz
Ergebnis	Gefilterter Datensatz

Beschreibung:

Der Datensatz wird einmal vorwärts, einmal rückwärts gefiltert. Dabei wird aber ein Filter benutzt, das zwar die vorgegebene Charakteristik, aber als Ordnung nur die Hälfte der angegebenen Ordnung hat.

Der Amplituden-Frequenzgang ist so normiert, dass bei der Grenzfrequenz eine Dämpfung von 3dB vorliegt.

Aus den übergebenen Parametern werden die Filter-Koeffizienten mit bilinearer Transformation berechnet.

Für eine sinnvolle Filterung sollte die Grenzfrequenz deutlich unterhalb der halben Abtastfrequenz des Ausgangssignals liegen. Je näher die Frequenz an der Abtastfrequenz liegt, um so ungenauer wird der Amplitudengang des Filters.

Das Filter ist nicht kausal. Damit gilt die zeitliche Reihenfolge von Ursache und Wirkung nicht mehr.

Einschwingvorgänge gibt es in beide Richtungen, insbesondere auch nicht nur zu Beginn, sondern auch am Ende des Datensatzes.

Die Phase hat idealerweise den Wert null. Praktisch trifft das aber nur auf den eingeschwungenen Zustand zu, also nicht auf die Randbereiche.

Beispiele:

```
f = FiltBsZ ( Vibration, 0, 0, 4, 45, 55 )
```

Ein Butterworth-Bandsperre-Filter 4. Ordnung mit einem Frequenzbereich von 45Hz bis 55Hz wird berechnet.

Siehe auch:

FiltLpZ, FiltBp

FilterAnalog

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein Datensatz wird mit einem Filter gefiltert, dessen analoge Koeffizienten der Übertragungsfunktion H(s) gegeben sind.

Deklaration:

```
FilterAnalog ( Eingangskanal, Koeffizienten ) -> Ergebnis
```

Parameter:

Eingangskanal	Der zu filternde Zeitdatensatz, die Zeit in Sekunden skaliert.
Koeffizienten	Ein Datensatz, in dem die Koeffizienten der Reihe nach stehen. Beschreibung der Anordnung siehe unten.
Ergebnis	Der gefilterte Datensatz

Beschreibung:

Die Abtastzeit muss klein genug sein, sodass die entscheidenden Knicke oder Resonanzen der Frequenzgänge realisiert werden können.

Aus den analogen Koeffizienten wird ein digitales Filter mittels bilinearer Transformation entworfen. Das digitale Filter ist i. Allg. nur eine mehr oder weniger gute Annäherung an das wirkliche Verhalten des analogen Filters.

Vor allem bei den höchsten Frequenzen im Bereich der halben Abtastfrequenz sind die Fehler am größten. Differenzierer können so nicht realisiert werden.

Es wird stets ein rekursives digitales Filter benutzt, sodass die Phase i. Allg. auch nicht sehr gut eingehalten wird.

Die Funktion ist nur für stabile Filter geeignet.

Anordnung der Koeffizienten:

Die Übertragungsfunktion des Filters wird als Produkt von Termen 2. Ordnung (Biquads) dargestellt. Für jedes Biquad gibt es 6 Koeffizienten.

Sei:

- p = i * 2 * PI * f, f Frequenz, i = sqrt(-1).

Dann ist die Übertragungsfunktion (hier bei 2. Ordnung):

- A (p) = (d0 + d1 * p + d2 * p^2) / (c0 + c1 * p + c2 * p^2)

Beachten Sie die positiven Potenzen von p!

Alternative Darstellung:

- A (p) = (d0 * p^-2 + d1 * p^-1 + d2) / (c0 * p^-2 + c1 * p^-1 + c2)

Die Koeffizienten werden in folgender Reihenfolge eingetragen:

- d0, d1, d2, c0, c1, c2

Falls mehr als 1 Biquad enthalten ist, wiederholt sich das Schema, wobei die Koeffizientenliste entsprechend fortgesetzt wird:

- d0_1, d1_1, d2_1, c0_1, c1_1, c2_1, d0_2, d1_2, d2_2, c0_2, c1_2, c2_2, ...

Bis zu 1000 Biquads können bearbeitet werden.

Beispiele:

```
d0 = 1
d1 = 0.08
d2 = 0
c0 = 1
c1 = 0.125
c2 = 0.08 * 0.08
acoeff = leng( 0, 6 )
acoeff[1] = d0
acoeff[2] = d1
acoeff[3] = d2
acoeff[4] = c0
```

```
acoeff[5] = c1
acoeff[6] = c2
Gefiltert =  FilterAnalog ( Beschleunigung, acoeff)
```

Die Beschleunigung hat eine Abtastzeit von 1ms. Die Übertragungsfunktion hat folgendes Aussehen:

A(p) = (1 + 0.08 * p) / (1 + 0.125 * p + (0.08 * p)^2)

Siehe auch:

FiltTP, VibrationFilter, dFilt, ExpoRms

FiltHP

Hochpass-Filter

Deklaration:

```
FiltHP ( Daten, EwCharakteristik, EwParameter, EwOrdnung, EwGrenzfrequenz ) -> Filtrat
```

Parameter:

Daten	Zu filternder Datensatz [ND].
EwCharakteristik	Filtercharakteristik
	0 : Butterworth
	1 : Bessel
	2 : Tschebyschew
	3 : Kritische Dämpfung
EwParameter	Bei Tschebyschew-Charakteristik die gewünschte Welligkeit (0..3) in dB. Sonst auf 0 zu setzen.
EwOrdnung	Ordnung des Filters. Für Bessel im Bereich 1..20, Tschebyschew 1..50, sonst 1..100.
EwGrenzfrequenz	Gewünschte Grenzfrequenz in Hz.
Filtrat	Gefilteter Datensatz.

Beschreibung:

Der übergebene Datensatz wird einer Hochpassfilterung unterzogen.

Aus den übergebenen Parametern werden die Filter-Koeffizienten mit bilinearer Transformation berechnet.

Für eine sinnvolle Filterung sollte die Grenzfrequenz deutlich unterhalb der halben Abtastfrequenz des Ausgangssignals liegen. Je näher die Frequenz an der Abtastfrequenz liegt, um so ungenauer wird der Amplitudengang des Filters.

Beispiele:

```
HighPassFiltered = FiltHP(signal, 2, 1, 10, 100)
```

Der Datensatz wird einer Hochpassfilterung mit Tschebyschew-Charakteristik unterzogen. Die Welligkeit soll 1dB betragen. Es wird ein Filter 10.Ordnung benutzt. Die Grenzfrequenz liegt bei 100Hz.

Siehe auch:

FiltBS, FiltBP, FiltLP, DFilt, FiltHpZ

FiltHpZ

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Hochpass-Filter ohne Phasenverschiebung

Deklaration:

```
FiltHpZ ( Eingangskanal, Charakteristik, Parameter, Ordnung, Grenzfrequenz ) -> Ergebnis
```

Parameter:

Eingangskanal	Der zu filternde Zeitdatensatz, die Zeit in Sekunden skaliert.
Charakteristik	Filter-Charakteristik
	0 : Butterworth
	1 : Bessel
	2 : Tschebyschew
	3 : Kritische Dämpfung
Parameter	Bei Tschebyschew-Charakteristik die Welligkeit in dB (0..3). Sonst auf 0 zu setzen.
Ordnung	Ordnung des Filters: 2, 4, 6, ... 20
Grenzfrequenz	Untere Grenzfrequenz in Hz
Ergebnis	Gefilterter Datensatz

Beschreibung:

Der Datensatz wird einmal vorwärts, einmal rückwärts gefiltert. Dabei wird aber ein Filter benutzt, das zwar die vorgegebene Charakteristik, aber als Ordnung nur die Hälfte der angegebenen Ordnung hat.

Der Amplituden-Frequenzgang ist so normiert, dass bei der Grenzfrequenz eine Dämpfung von 3dB vorliegt.

Aus den übergebenen Parametern werden die Filter-Koeffizienten mit bilinearer Transformation berechnet.

Für eine sinnvolle Filterung sollte die Grenzfrequenz deutlich unterhalb der halben Abtastfrequenz des Ausgangssignals liegen. Je näher die Frequenz an der Abtastfrequenz liegt, um so ungenauer wird der Amplitudengang des Filters.

Das Filter ist nicht kausal. Damit gilt die zeitliche Reihenfolge von Ursache und Wirkung nicht mehr.

Einschwingvorgänge gibt es in beide Richtungen, insbesondere auch nicht nur zu Beginn, sondern auch am Ende des Datensatzes.

Die Phase hat idealerweise den Wert null. Praktisch trifft das aber nur auf den eingeschwungenen Zustand zu, also nicht auf die Randbereiche.

Beispiele:

```
f = FiltHpZ ( Vibration, 0, 0, 2, 5 )
```

Ein Butterworth-Hochpass-Filter 2. Ordnung mit einer unteren Grenzfrequenz von 5Hz wird berechnet.

Siehe auch:

FiltLpZ, FiltHp

FiltLP

Tiefpass-Filter
Alternativer Name: **FiltTP**

Deklaration:

```
FiltLP ( Daten, EwCharakteristik, EwParameter, EwOrdnung, EwGrenzfrequenz ) -> Filtrat
```

Parameter:

Daten	Zu filternder Datensatz [ND].
EwCharakteristik	Filtercharakteristik
	0 : Butterworth
	1 : Bessel
	2 : Tschebyschew
	3 : Kritische Dämpfung
EwParameter	Bei Tschebyschew-Charakteristik die gewünschte Welligkeit (0..3) in dB. Sonst auf 0 zu setzen.
EwOrdnung	Ordnung des Filters. Für Bessel im Bereich 1..20, Tschebyschew 1..50, sonst 1..100.
EwGrenzfrequenz	Gewünschte Grenzfrequenz in Hz.
Filtrat	Gefilteter Datensatz.

Beschreibung:

Der übergebene Datensatz wird einer Tiefpassfilterung unterzogen.

Aus den übergebenen Parametern werden die Filter-Koeffizienten mit bilinearer Transformation berechnet.

Für eine sinnvolle Filterung sollte die Grenzfrequenz deutlich unterhalb der halben Abtastfrequenz des Ausgangssignals liegen. Je näher die Frequenz an der Abtastfrequenz liegt, um so ungenauer wird der Amplitudengang des Filters.

Beispiele:

```
LowPassFiltered = FiltLP(signal, 2, 1, 10, 100)
```

Der Datensatz wird einer Tiefpassfilterung mit Tschebyschew-Charakteristik unterzogen. Die Welligkeit soll 1dB betragen. Es wird ein Filter 10.Ordnung benutzt. Die Grenzfrequenz liegt bei 100Hz.

Siehe auch:

FiltBS, FiltBP, FiltHP, DFilt, FiltLpZ, SavitzkyGolay

FiltLpZ

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Tiefpass-Filter ohne Phasenverschiebung

Deklaration:

```
FiltLpZ ( Eingangskanal, Charakteristik, Parameter, Ordnung, Grenzfrequenz ) -> Ergebnis
```

Parameter:

Eingangskanal	Der zu filternde Zeitdatensatz, die Zeit in Sekunden skaliert.
Charakteristik	Filter-Charakteristik
	0 : Butterworth
	1 : Bessel
	2 : Tschebyschew
	3 : Kritische Dämpfung
Parameter	Bei Tschebyschew-Charakteristik die Welligkeit in dB (0..3). Sonst auf 0 zu setzen.
Ordnung	Ordnung des Filters: 2, 4, 6, ... 20
Grenzfrequenz	Obere Grenzfrequenz in Hz
Ergebnis	Gefilterter Datensatz

Beschreibung:

Der Datensatz wird einmal vorwärts, einmal rückwärts gefiltert. Dabei wird aber ein Filter benutzt, das zwar die vorgegebene Charakteristik, aber als Ordnung nur die Hälfte der angegebenen Ordnung hat.

Der Amplituden-Frequenzgang ist so normiert, dass bei der Grenzfrequenz eine Dämpfung von 3dB vorliegt.

Aus den übergebenen Parametern werden die Filter-Koeffizienten mit bilinearer Transformation berechnet.

Für eine sinnvolle Filterung sollte die Grenzfrequenz deutlich unterhalb der halben Abtastfrequenz des Ausgangssignals liegen. Je näher die Frequenz an der Abtastfrequenz liegt, um so ungenauer wird der Amplitudengang des Filters.

Das Filter ist nicht kausal. Damit gilt die zeitliche Reihenfolge von Ursache und Wirkung nicht mehr.

Einschwingvorgänge gibt es in beide Richtungen, insbesondere auch nicht nur zu Beginn, sondern auch am Ende des Datensatzes.

Die Phase hat idealerweise den Wert null. Praktisch trifft das aber nur auf den eingeschwungenen Zustand zu, also nicht auf die Randbereiche.

Beispiele:

```
f = FiltLpZ ( Vibration, 0, 0, 2, 150 )
```

Ein Butterworth-Tiefpass-Filter 2. Ordnung mit einer oberen Grenzfrequenz von 150Hz wird berechnet.

Siehe auch:

FiltHpZ, FiltTp, SavitzkyGolay

Flag?

Die Funktion fragt den Zustand (an/aus) spezieller Datensatz-Attribute ab.

Deklaration:

```
Flag? ( Daten, EwFlag ) -> EwAnOderAus
```

Parameter:

Daten	Datensatz, dessen Attribut abgefragt werden soll.
EwFlag	Auswahl des Attributes
	0 : Das aktuelle Datenformat des Datensatzes ist fixiert, d.h. bei nachfolgenden Berechnungen wird nach Möglichkeit das Datenformat (und bei Integer-Formaten die Skalierungsinformation) beibehalten.
	1 : Die Werte des Datensatzes können als Farbinformation für 1 Pixel eines Bildes interpretiert werden. Nur erlaubt für die Datenformate '4 Byte ohne Vorzeichen' (Farbinformation ist im RGB-Format codiert) oder '1 Byte ohne Vorzeichen' (Farbinformation ist als Graustufenwert in Bereich 0..255 codiert). Das Attribut wird standardmäßig nur von Importfiltern für Bilddateien oder Spezialfunktionen wie VpGetImages() gesetzt. Es wird vom Kurvenfenster verwendet, um die Anzeige von Bilddaten zu optimieren.
EwAnOderAus	AnOderAus
	0 : Aus
	1 : An

Beschreibung:

Die Funktion fragt gewisse Datensatz-Attribute ab, die lediglich die boolschen Werte [An] (bzw. "Wahr") oder [Aus] (bzw. "Falsch") annehmen können.

Beispiele:

Ein Datensatz wird geladen und geglättet. Falls der Datensatz in einem ganzzahligen Integer-Format vorliegt und dieses auch fixiert ist, wird der Anwender vor dem Glätten gefragt, ob die Fixierung aufgehoben werden soll. Ansonsten würde das Ergebnis der Glättung wieder im Integer-Format abgelegt werden, dessen Auflösung für solche Zwecke oft nicht ausreichend ist.

```
FileLoad("test.dat", "", 0)
IF DataFormat?(test) > 1
   IF Flag?(test,0)
      IF BoxMessage("Frage", "Fixierung Datenformat aufheben?", "?4")
         SetFlag(test, 0, 0)
      END
   END
END
test = Smo5(test)
```

Siehe auch:

SetFlag, SetDataFormat, RGB, VpGetImages

Flipflop

Verfügbar ab: Professional Edition

FlipFlop

Deklaration:

```
Flipflop ( RJ, SK, Typ ) -> Ergebnis
```

Parameter:

RJ	Eingangsdaten R bzw. J
SK	Eingangsdaten S bzw. K
Typ	Typ des Flipflops
	"RS" : RS Flipflop
	"JK" : JK Flipflop
Ergebnis	Ergebnis

Beschreibung:

Sind die Eingangsdaten null, wird das als logisch null (false, low) gedeutet. Sonst als logisch 1 (true, high).

Beide Eingangskanäle müssen dieselbe Zeitbasis, Länge und Struktur (Segmente und Events) aufweisen.

RS Flipflop

Liefert eine 1 für den Zustand H und eine 0 für den Zustand L. Beginnend mit dem Zustand L wird der Zustand H eingenommen, wenn S ungleich null und R gleich null ist. Ist S gleich null und R ungleich null, so wird der Zustand L eingenommen. Bei den beiden anderen Kombinationen von Eingangswerten bleibt der Zustand erhalten.

JK Flipflop

Liefert eine 1 für den Zustand H und eine 0 für den Zustand L. Beginnend mit dem Zustand L wird der Zustand H eingenommen, wenn J ungleich null und K gleich null ist. Ist J gleich null und K ungleich null, so wird der Zustand L eingenommen. Ist sowohl J als auch K gleich null, so bleibt der Zustand erhalten. Ist sowohl J als auch K ungleich null, so wird in den anderen Zustand gewechselt.

Beispiele:

RS Flipflop

```
RS = Flipflop ( R, S, "RS" )
; R = 0 1 1 0 1 0 1 0 1 0 1 0
; S = 0 0 1 1 1 0 0 0 1 1 1 0
;RS = 0 0 0 1 1 1 0 0 0 1 1 1
```

Die LED wird eingeschaltet, wenn das Signal den Bereich von 0 bis 8 verlässt und wird erst wieder ausgeschaltet, wenn sich das Signal im Bereich von 1 bis 6 befindet

```
LED = Flipflop( (K1 < 6) AND (K1 > 1), (K1 > 8) OR  (K1 < 0), "RS" )
```

JK Flipflop

```
JK = Flipflop ( J, K, "JK" )
; J = 0 0 1 1 1 0 1 1 0
; K = 0 1 1 1 0 0 1 1 1
;JK = 0 0 1 0 1 1 0 1 0
```

Die LED blinkt solange das Signal größer als 9 ist.

```
GT = (Signal > 9)
LED = Flipflop( GT, GT, "JK" )
```

Siehe auch:

Monoflop

Floor

Nächstkleinere oder gleiche ganze Zahl.
Alternativer Name: **Ganz**

Deklaration:

```
Floor ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Parameter. Erlaubte Typen: [ND],[XY].
Ergebnis	Ganzzahliger Anteil des Parameters

Beschreibung:

Der ganzzahlige Anteil einer Zahl ist die nächstkleinere oder gleich große ganze Zahl. So wird z. B. bei 1.0, 1.1, 1.9 jeweils der ganzzahlige Anteil 1.0, bei -3.1, -3.4, -4.0 der ganzzahlige Anteil -4.0.

Anmerkungen

- Die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Der Parameter darf strukturiert sein (Events/ Segmente).
- Die Einheit wird nicht verändert.
- Beachten Sie, dass Rundungsfehler beim Rechnen mit reellen Zahlen auftreten können. So liefert Floor(sqr(sqrt(2))) nicht wie erwartet den Wert 2 sondern den Wert 1 zurück. Wollen Sie die Funktion Floor() sicherer (aber dafür nicht so exakt!) gestalten, addieren Sie einen kleinen Betrag zum Parameter von Floor, z. B. floor(1e-6 + sqr(sqrt(2))).
- Die Funktion Floor() ist besonders nützlich für diverse Rundungen. Diese können oft aber auch bequemer mit der Funktion Round() erledigt werden.

Beispiele:

Eine Zahl auf die nächstliegende ganze Zahl gerundet:

```
rounded = Floor(number + 0.5)
Hier werden die Werte eines Datensatzes auf 2 Stellen hinter dem Komma gerundet:
```

```
NDrounded = 0.01 * Floor(0.5 + 100 * NDdata)
```

Der Diskretisierungseffekt eines 8-Bit-AD-Wandlers wird simuliert. 8-Bit entsprechen 256 Stufen. Der Wandler hat einen analogen Aussteuerungsbereich von 10V:

```
NDdiscrete = Floor(NDdata* 256 / 10) * 10 / 256
```

Siehe auch:

Round

FOR

Zählergesteuerte Schleife

Deklaration:

```
FOR EwZähler = EwStart TO EwEnde STEP EwSchritt
```

Parameter:

EwZähler	Zählvariable
EwStart	Beim ersten Eintritt in die Schleife wird die Zählvariable auf diesen Wert gesetzt.
EwEnde	Die Schleife wird abgebrochen, wenn die Zählvariable diesen Wert über- (Vorwärtszählen) bzw. unterschreitet (Rückwärtszählen).
EwSchritt	Nach jedem Schleifendurchlauf wird die Zählvariable um diesen Wert verändert. Diese Angabe ist optional, Standard ist +1, d.h. die Zählvariable wird mit jedem Durchlauf um 1 erhöht.

Beschreibung:

Das Ende der Schleife wird mit dem Befehl END markiert.

Die Schrittweite kann auch negativ sein (rückwärts zählende Schleife). Der Startwert muss dann größer oder gleich dem Endwert sein.

In der Schleife können die Befehle BREAK und CONTINUE verwendet werden, um die Abarbeitung der inneren Anweisungen vorzeitig abzubrechen.

Die Zählvariable darf innerhalb der Schleife geändert werden, aber weder gelöscht noch im Typ verändert werden.

Das Konstrukt

```
FOR i = start TO end STEP add
   ;...Anweisungen
END
```

ist äquivalent zu:

```
i = start
WHILE i <= end
   ;...Anweisungen
   i = i + add
END
```

Beispiele:

Berechnung der Fakultät von 5:

```
F = 1
FOR i = 2 TO 5
   F = F * i
END
```

In einer Datengruppe werden alle Kanäle entfernt, deren Maximum kleiner als 0 ist.

```
n = GrChanNum?(group)
FOR i = n TO 1 STEP -1
   IF max(group:[i]) < 0
      GrChanDel(group, i)
   END
END
```

Es werden alle Dateien mit der Erweiterung "*.dat" in einem vorgegebenen Verzeichnis ermittelt und in einer Schleife aufgezählt. Wenn die Dateizeit nach einem festen Stichtag liegt, wird die Datei geladen und verarbeitet. Nach maximal 10 geladenen Dateien wird die Verarbeitung abgebrochen.

```
list  = FsFileListNew("c:\imc\dat", "*.dat", 0, 0, 0)
count = FsFileListGetCount(list)
loaded = 0
deadline = TimeJoin(1, 1, 2012, 0, 0, 0)
FOR i = 1 TO count
   time = FsFileListGetTime(list, i)
   IF time < deadline
      CONTINUE
   END
   ; load and process file
   TxName = FsFileListGetName(list, i)
   fh = FileOpenDSF(TxName, 0)
   ; ...
```

```
   loaded = loaded +1
   IF loaded = 10
      BREAK
   END
END
FsFileListClose(list)
```

Siehe auch:

FOREACH, WHILE, BREAK, CONTINUE

FOREACH

Dieses Kommando leitet eine Schleife ein, in der die Elemente eines Datenobjektes aufgezählt werden. Für jeden Durchlauf wird das jeweils aktuelle Element einer Laufvariablen zugewiesen.

Deklaration:

```
FOREACH ElementTyp Laufvariable IN Aufzählungsvariable
```

Parameter:

ElementTyp	Gibt den Typ der Elemente an, die aufgezählt werden sollen.
	SAMPLE : Es werden alle Werte eines Datensatzes aufgezählt.
	VALUE : Es werden alle Zahlenwerte eines Datensatzes aufgezählt (Einheit und andere Kennwerte werden ignoriert). Schneller als "SAMPLE". Die Laufvariable darf in der Schleife nicht geändert werden.
	SEGMENT : Es werden alle Segmente eines Datensatzes aufgezählt.
	EVENT : Es werden alle Events eines Datensatzes aufgezählt.
	CHANNEL : Die in einer Datengruppe enthaltenen Datensätze und Textvariablen werden aufgezählt.
	ELEMENT : Die in einem Textfeld enthaltenen Text-Elemente werden aufgezählt.
Laufvariable	Der Laufvariablen wird in jedem Durchlauf das aktuelle Element der Aufzählungsvariablen zugewiesen. Abhängig vom Aufzählungstyp handelt es sich dabei um einen einzelnen Wert, Segment oder Event der Aufzählungsvariablen oder um einen kompletten Kanal einer Datengruppe.
Aufzählungsvariable	Variable, deren Elemente aufgezählt werden sollen.

Beschreibung:

Das Ende einer FOREACH-Schleife wird durch den Befehl END markiert.

Die Abarbeitung einer FOREACH-Schleife kann vorzeitig durch die Befehle BREAK oder CONTINUE abgebrochen werden.

Die Aufzählungsvariable darf innerhalb des Schleifenkörpers nicht verändert werden.

Die Laufvariable darf innerhalb des Schleifenkörpers geändert werden (außer bei "VALUE"), der Typ muss allerdings erhalten bleiben.

Falls sich der Inhalt der Laufvariable geändert hat, wird am Ende eines Schleifendurchlaufs (END/BREAK/CONTINUE) der neue Inhalt in die Aufzählungsvariable zurückgeschrieben.

Für eine 'nur lesende' Aufzählung der numerischen Werte eines Datensatzes ist "VALUE" effizienter als "SAMPLE".

Das Konstrukt

```
FOREACH SAMPLE s IN data
   ; Anweisungen
END
```

ist äquivalent zu

```
i = 1
WHILE i<= Leng?(data)
   s = data[i]
   ; Anweisungen
   data[i] = s  ; Nur wenn s sich geändert hat
   i = i+1
END
```

Das Konstrukt

```
FOREACH CHANNEL c IN group
   ; Anweisungen
END
```

ist äquivalent zu

```
i = 1
WHILE i<= GrChanNum?(group)
   c = group:[i]
   ; Anweisungen
   group:[i] = c  ; Nur wenn c sich geändert hat
   i = i+1
END
```

Der Typ der Aufzählungsvariablen muss zum Aufzählungstyp passen. Falls für den Typ beispielsweise EVENT angegeben ist, muss die Aufzählungsvariable ein eventierter Datensatz sein. Der Aufzählungstyp CHANNEL erfordert dagegen eine Datengruppe.

Das Aufzählen kann nicht über Segment- oder Event-Grenzen hinweg erfolgen. Falls beispielsweise SAMPLE verwendet wird und die Aufzählungsvariable ein segmentierter Datensatz ist, muss zusätzlich das gewünschte Segment ausgewählt werden:

```
FOREACH SAMPLE s IN SegmentedData[12]
```

Um alle Werte eines Datensatzes mit Events und Segmenten aufzuzählen, können Sie auch mehrere Schleifen schachteln:

```
FOREACH EVENT ev IN data
   FOREACH SEGMENT seg IN ev
      FOREACH SAMPLE s IN seg
         ; Anweisungen
      END
   END
END
```

Setzen Sie Schleifen mit Bedacht ein. Viele Aufgabenstellungen, die beispielsweise den Einsatz der FOREACH SAMPLE-Aufzählung nahelegen würden (z.B. Suche im Datensatz nach bestimmten Kriterien und ggf. Bearbeitung der Fundstellen) sind durch geeigneten Einsatz der mathematischen und analytischen Funktionen in FAMOS wesentlich effizienter zu lösen.

Beispiele:

In einem Datensatz [data] sollen Störungen eliminiert werden. Diese sind durch einen Wert > 1000 identifizierbar. Diese Spitzen werden durch den jeweils gültigen Vorgängerwert ersetzt.

```
lastValid = 0
FOREACH SAMPLE s in data
   IF s > 1000
      s = lastValid
   ELSE
      lastValid = s
   END
END
```

In einem eventierten Datensatz werden alle Events identifiziert, deren Maximum > 0 ist und diese in einen neuen Datensatz kopiert.

```
newdata = EMPTY
FOREACH EVENT ev IN data
   IF max(ev) > 0
      EventAppend(newData, ev, 0)
   END
END
```

In einer Datengruppe werden alle Kanäle, deren Standardabweichung größer als 2 ist, einer Glättung unterzogen:

```
FOREACH CHANNEL c in group
   IF StDev(c) > 2
      c = Smo5(c)
   END
END
```

Ein Textfeld [AllPathNames] enthält die kompletten Pfadnamen von zu ladenden Dateien:

```
FOREACH ELEMENT path in AllPathNames
   FileLoad(path,"", 0)
END
```

Siehe auch:

FOR, WHILE

FrequencyResponse

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Übertragungsfunktion (FRF, Frequency response function). Berechnung mittels FFT.

Deklaration:

```
FrequencyResponse ( Eingangskanal, Ausgangskanal, Fensterbreite, Fenstertyp, Überlappung, Typ [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangskanal	Der Bezugskanal, Eingangskanal. Die Anregung eines Systems. In Sekunden skaliert.
Ausgangskanal	Der (verzögerte) Ausgangskanal. Die Antwort eines Systems. Ein- und Ausgangskanal haben dieselbe Zeitbasis und sind in Sekunden skaliert.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Typ	Nach welchem Verfahren wird die Übertragungsfunktion bestimmt? In den folgenden Formeln bedeuten: x: Eingangskanal, y: Ausgangskanal, G: Leistungsspektrum, F: Spektrum.
	0 : H1: Gxy / Gxx: Messung mit verrauschten Ausgangssignal, Berechnung über Auto- und Kreuzspektrum
	1 : H2: Gyy / Gyx: Messung mit verrauschten Eingangssignal, Berechnung über Auto- und Kreuzspektrum
	2 : HV: (Gxy / |Gxy|) * sqrt (Gyy / Gxx): Geometrisches Mittel von H1 und H2. Ein- und Ausgang sind verrauscht.
	3 : H: Fy / Fx: Nicht empfohlen!, Ausgangsspektrum / Eingangsspektrum.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Die Übertragungsfunktion. Das Ergebnis ist komplex, d.h. es enthält Betrag und Phase. Die Phase wird im Bereich -180 ... + 180 Grad bestimmt.

Beschreibung:

Die Übertragungsfunktion wird ermittelt durch lineare Mittelung von Leistungsspektren.

Die Berechnung der Mittelung erfolgt über Real- und Imaginärteil getrennt.

Die Berechnung wird erst sinnvoll, wenn eine merkliche Mittelung stattfindet.

Beispiele:

```
FRF = FrequencyResponse ( Kraft, Bewegung, 1000, 0, 50, 0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Leistungsspektren, die sich um je 50% überlappen. Eine Kraft wirkt auf ein mechanisches Teil. Die Bewegung des Bauteils am anderen Ende wird gemessen. Aus den gemittelten Leistungsspektren wird die Übertragungsfunktion vom Typ H1 bestimmt.

Siehe auch:

CrossPowerDS, Coherence, PhaseContinuous

FsCopyFile

Dateien oder Verzeichnisse kopieren

Deklaration:

```
FsCopyFile ( TxQuelldatei, TxZieldatei, Option, Tiefe ) -> Status
```

Parameter:

TxQuelldatei	Quelldateiname
TxZieldatei	Zieldateiname
Option	Option
	0 : Existierende Dateien nicht überschreiben
	1 : Nur neuere Dateien kopieren
	2 : Existierende Dateien immer überschreiben
Tiefe	Nur Verzeichnis oder auch Unterverzeichnisse kopieren
	0 : Nur angegebenes Verzeichnis kopieren
	1 : Unterverzeichnisse einschliessen
Status	Erfolg der Funktion
	0 : Funktion war erfolgreich
	-1 : Fehler ist aufgetreten. Fehlerursache kann mittels FsGetLastError() ermittelt werden.

Beschreibung:

Diese Funktion kopiert Dateien oder Verzeichnisse. Es kann mit dem Parameter Tiefe festgelegt werden, ob auch alle Unterverzeichnisse kopiert werden sollen.

Angabe des Quelldateinamens

- Der Name darf keine relative Pfadangabe sein.
- Der Name darf Globalzeichen enthalten.
- Wird ein eindeutiger Verzeichnis- bzw. Dateiname angegeben, so muß dieser existieren
- Entspricht ein eindeutige Name einer Datei, so ist der Parameter Tiefe ohne Bedeutung.

Angabe des Zieldateinamens

- Der Name darf keine relative Pfadangabe sein.
- Der Name darf keine Globalzeichen enthalten.
- Werden im Quellnamen Globalzeichen verwendet, so muß der Zielname ein existierendes Verzeichnis sein
- Ist die Quelle ein eindeutiger Dateiname und das Ziel ein existierendes Verzeichnis, so wird der Dateiname in das Ziel übernommen.

Beispiele:

Alle Dateien mit der Erweiterung *.raw aus dem Verzeichnis 'H:\neu' sollen in das Verzeichnis 'd:\archiv\0003' kopiert werden.Es werden in diesem Fall nur neuere Dateien kopiert.

```
erg=FsCopyFile("H:\neu\*.raw","d:\a\0003",1,0)
IF erg=-1
   error$=FsGetLastError()
END
```

Das Verzeichnis "c:\temdat\00001" und sein kompletter Inhalt wird nach "g:\archiv\" kopiert.

```
erg=FsCopyFile("c:\temdat\00001","g:\archiv\",1,1)
IF erg=-1
   error$=FsGetLastError()
END
```

Siehe auch:

FsMoveFile, FsGetLastError

FsCreateDirectory

Verzeichnis erzeugen

Deklaration:

```
FsCreateDirectory ( TxDirName ) -> Status
```

Parameter:

TxDirName	Verzeichnisname
Status	Erfolg der Funktion
	0 : Funktion war erfolgreich
	-1 : Fehler ist aufgetreten. Fehlerursache kann mittels FsGetLastError() ermittelt werden.

Beschreibung:

Diese Funktion erzeugt Verzeichnisse, auch mit mehreren Unterverzeichnissen. Es muß ein vollständiger Pfad angegeben werden.

Die Angabe in TxDirName darf keine Globalzeichen '*' oder '?' enthalten

Die Funktion kann folgende Werte liefern:

- 0 Das Erzeugen der Verzeichnisses war erfolgreich
- -1 Es ist ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

Beispiele:

In dem bestehenden Verzeichnis 'L:\Temp' sollen die Unterverzeichnisse 'Dat' und '00001' erzeugt werden.

```
dir$="l:/temp/dat/00001"
result=FsCreateDirectory(dir$)
IF result<>0
   error$=FsGetLastError()
END
```

Siehe auch:

FsRemoveDirectory

FsDeleteFile

Datei(en) löschen

Deklaration:

```
FsDeleteFile ( TxDateiname ) -> Anzahl
```

Parameter:

TxDateiname	Dateiname
Anzahl	Anzahl der gelöschten Dateien
	>=0 : Anzahl der gelöschten Dateien
	-1 : Fehler. Er kann mit der Funktion FsGetLastError() ermittelt werden.

Beschreibung:

Die Funktion löscht eine oder mehrere Dateien.Im Dateinamen können die Globalzeichen '*' oder '?' enthalten sein.

Schreibgeschützte Dateien können mit dieser Funktion nicht gelöscht werden

Die Funktion liefert als Ergebnis die Anzahl der gelöschten Dateien.

Die Pfadangabe muß vollständig sein.

Mit der Funktion FsGetLastError() kann ein eventueller Fehler ermittelt werden.

Beispiele:

Aus dem Verzeichnis c:\test werden alle Dateien mit der Erweiterung raw gelöscht.

```
file$="c:\test\*.raw"
n=FsDeleteFile(file$)
error$=FsGetLastError()
```

Siehe auch:

FsMoveFile, FsGetLastError

FsDlgSelectDirectory

Verzeichnisauswahldialog

Deklaration:

```
FsDlgSelectDirectory ( TxTitel, TxStartDir, Option ) -> TxDirname
```

Parameter:

TxTitel	Titelzeile des Dialogs
TxStartDir	Anfangsverzeichnis
Option	Option
	0 : Standard
	1 : Mit Eingabe von Verzeichnisnamen
TxDirname	Vollständiger Verzeichnisname

Beschreibung:

Mit dieser Funktion kann per Dialog ein Verzeichnis ausgewählt werden.

- Ist TxTitel leer, wird als Titel "Ordner wählen" ausgegeben.
- Existiert das übergebene Anfangsverzeichnis nicht, wird als Anfangsverzeichnis das Datenverzeichnis von FAMOS genommen.
- Die Funktion liefert die Pfadangabe des selektierten Verzeichnisses.
- Position und Größe des Dialoges kann mit der Funktion SetBoxPos() eingestellt werden.
- Wird der Dialog abgebrochen, so ist der Rückgabewert leer.

Beispiele:

```
dir$ = FsDlgSelectDirectory("Verzeichnis", "c:\copy", 0)
IF TLeng(dir$) > 0
   FileListID = FsFileListNew(dir$, "*.*", 2, 1, 0)
   n=FsFileListGetCount(FileListID)
   i=1
   WHILE i <= n
      datei$=FsFileListGetName(FileListID,i)
      ;
      i=i+1
   END
   FsFileListClose(FileListID)
END
```

Siehe auch:

FsFileListNew, FsFileListGetCount, FsFileListGetName, FsFileListClose, FsGetLastError

FsDlgSelectFiles

Dateienauswahldialog

Deklaration:

```
FsDlgSelectFiles ( TxTitel, TxStartDir, TxPattern, Reserviert ) -> FileListID
```

Parameter:

TxTitel	Titelzeile des Dialogs
TxStartDir	Anfangsverzeichnis
TxPattern	Dateifilter
Reserviert	Reservierter Parameter, immer 0
FileListID	ID-Wert der Dateiliste
	>=1 : ID-Wert der Dateiliste
	-1 : Fehler. Er kann mit der Funktion FsGetLastError() ermittelt werden.
	-2 : Der Dialog wurde abgebrochen.

Beschreibung:

Mit dieser Funktion können per Dialog mehrere Datei ausgewählt werden

- Wird in TxTitel keine Zeichenkette angegeben, so hat der Dialog seine Standardüberschrift.
- Ist das Anfangsverzeichnis in TxStartDir ungültig oder leer, so wird das eingestellte Datenverzeichnis von FAMOS verwendet.
- Ist der Parameter TxPattern leer, dann ist Dateiname *.* und Dateityp Alle Dateien (*.*).
- Enthält TxPattern "*.raw", dann ist Dateiname *.raw und im Dateityp Alle Dateien (*.*).
- In TxPattern können auch mehrere, durch Semikolon getrennte, Filter angegeben werden: "*.raw;*.dat".
- Ist TxPattern="Messdatei | *.dat", so ist Dateiname *.dat und im Dateityp Messdatei (*.dat) sowie Alle Dateien (*.*).

Beispiele:

```
FileListID=FsDlgSelectFiles("Auswahl","c:\copy","*.*",0)
n=FsFileListGetCount(FileListID)
i=1
WHILE i <= n
      datei$=FsFileListGetName(FileListID,i)
      ;
      i=i+1
END
```

Siehe auch:

FsFileListNew, FsFileListGetCount, FsFileListGetName, FsFileListClose, FsGetLastError

FsFileExists

Prüft, ob die Datei oder das Verzeichnis existiert

Deklaration:

```
FsFileExists ( TxPathName ) -> Result
```

Parameter:

TxPathName	Pfadname
Result	Ergebnis
	-1 : Fehler
	0 : Datei- oder Verzeichnis existieren nicht
	1 : Pfadname enspricht einer Datei und diese existiert
	2 : Pfadname enspricht einem Verzeichnis und dieses existiert

Beschreibung:

Mit der Funktion kann getestet werden, ob eine Datei oder ein Verzeichnis existieren.

Die Angabe in TxPathName darf keine Globalzeichen '*' oder '?' enthalten

Im Fehlerfall liefert die Funktion den Wert -1. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

FsFileListClose

Dateiliste löschen

Deklaration:

```
FsFileListClose ( FileListID )
```

Parameter:

FileListID	ID-Wert der Dateiliste

Beschreibung:

Die Funktion löscht alle oder eine bestimmte Dateiliste. Nicht mehr benötigte Dateilisten sollten mit dieser Funktion weggeräumt werden.

Das Datei-Kit kann maximal 20 Dateilisten verwalten.

Wird für den Parameter FileListID -1 angegeben, werden alle bestehenden Dateilisten gelöscht.

Beispiele:

Das Beispiel zeigt eine typische Anwendung der Funktion FsFileListClose().

Mit der Funktion wird FsDlgSelectFiles() wird eine Dateiliste erzeugt.

In der Programmschleife erfolgt eine Auswertung des Inhalts.

Am Ende wird die Dateiliste gelöscht.

```
FileListID = FsDlgSelectFiles("Datei selektieren", "c:\copy", "*.*", 0)
IF FileListID>0
   n=FsFileListGetCount(FileListID)
   i=1
   WHILE i <= n
      ro = FsFileListGetAttribute(FileListID, i, 1)
      :
      i=i+1
   END
   FsFileListClose(FileListID)
END
```

Siehe auch:

FsFileListNew, FsFileListGetCount, FsFileListGetName, FsFileListGetSize, FsFileListGetTime, FsFileListGetAttribute, FsDlgSelectFiles, FsGetLastError

FsFileListGetAttribute

Ermittelt ein Dateiattribut einer Datei aus einer Dateiliste

Deklaration:

```
FsFileListGetAttribute ( FileListID, Index, Attribut ) -> AnAus
```

Parameter:

FileListID	ID-Wert der Dateiliste
Index	Index des Eintrags in der Dateiliste. Der Index beginnt bei 1 und endet bei der Anzahl, die mit der Funktion FsFileListGetCount() ermittelt wurde.
Attribut	Gewünschtes Dateiattribut
	1 : Schreibgeschützt
	2 : Versteckt
	3 : Verzeichnis
AnAus	Dateiattribut
	-1 : Fehler ist aufgetreten. Fehlerursache kann mittels FsGetLastError() ermittelt werden.
	0 : Dateiattribut ist nicht gesetzt
	1 : Dateiattribut ist gesetzt

Beschreibung:

Mit dieser Funktion können einzelene Attribute einer Datei oder eines Verzeichnisses aus einer Dateiliste kann ermittelt werden.

Ist der ID-Wert der Liste ungültig oder liegt der Index des Eintrags ausserhalb des vorhandenen Bereiches, so wird -1 zurückgegeben. Mit der Funktion FsGetLastError() kann die Fehlerursache bestimmt werden.

Beispiele:

In diesem Beispiel wird das RO-Attribut aller Dateien und Verzeichnisse aus einer Dateiliste abgefragt.

In einem Auswahldialog können mehrere Datei ausgewählt werden. Diese Dateien werden intern in eine Dateiliste eingelesen und anschliessend ausgelesen.

Mit der Funktion FsFileListGetAttribute() wird in diesem Fall das RO-Attribut ermittelt.

```
FileListID = FsDlgSelectFiles("Datei selektieren", "c:\copy", "*.*", 0)
IF FileListID>0
   n=FsFileListGetCount(FileListID)
   i=1
   WHILE i <= n
      ro = FsFileListGetAttribute(FileListID, i, 1)
      :
      i=i+1
   END
   FsFileListClose(FileListID)
END
```

Siehe auch:

FsFileListNew, FsFileListGetCount, FsFileListGetName, FsFileListGetSize, FsFileListGetTime, FsFileListClose, FsDlgSelectFiles, FsGetLastError

FsFileListGetCount

Anzahl der Einträge einer Dateiliste ermitteln

Deklaration:

```
FsFileListGetCount ( FileListID ) -> Anzahl
```

Parameter:

FileListID	ID-Wert der Dateiliste
Anzahl	Anzahl der Einträge in der Dateiliste
	>=0 : Anzahl der Einträge in der Dateiliste
	-1 : Fehler. Er kann mit der Funktion FsGetLastError() ermittelt werden.

Beschreibung:

Die Funktion ermittelt die Anzahl der Einträge in einer Dateiliste. Diese Dateiliste wurde zuvor mit der Funktion FsFileListNew() erzeugt. Der von der Funktion FsFileListNew() erhaltene ID-Wert ist hier als Parameter anzugeben. Ist der Funktionswert -1, so liegt ein Fehler vor.Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

Beispiele:

Alle Dateien mit der Erweiterung DAT aus dem Verzeichnis 'c:\daten' einlesen und die Anzahl bestimmen

```
FileListID= FsFileListNew("c:\daten\", "*.dat", 2, 1, 1)
n=FsFileListGetCount(FileListID)
```

Siehe auch:

FsFileListNew, FsFileListGetName, FsFileListGetSize, FsFileListGetTime, FsFileListGetAttribute, FsFileListClose, FsGetLastError

FsFileListGetName

Dateinamen aus einer Dateiliste ermitteln

Deklaration:

```
FsFileListGetName ( FileListID, Index ) -> TxPathname
```

Parameter:

FileListID	ID-Wert der Dateiliste
Index	Index des Eintrags in der Dateiliste. Der Index beginnt bei 1 und endet bei der Anzahl, die mit der Funktion FsFileListGetCount() ermittelt wurde.
TxPathname	Vollständiger Dateiname. Die Zeichenkette ist leer, wenn ein Fehler aufgetreten ist.

Beschreibung:

Mit dieser Funktion werden Dateinamen aus einer Dateiliste gelesen. Die Dateiliste muss vorher mit den Funktionen FsFileListNew() oder FsDlgSelectFiles() angelegt worden sein.

Ist der ID-Wert der Liste ungültig oder liegt der Index des Eintrags ausserhalb des vorhandenen Bereiches, so wird eine leere Zeichenkette zurückgegeben. Mit der Funktion FsGetLastError() kann die Fehlerursache bestimmt werden.

Beispiele:

In einem Auswahldialog können mehrere Datei ausgewählt werden.

Diese Dateien werden intern in eine Dateiliste eingelesen und anschliessend ausgelesen.

Mit der Funktion FsFileListGetName() werden die Dateinamen ermittelt.

```
FileListID = FsDlgSelectFiles("Datei selektieren", "c:\copy", "*.*", 0)
IF FileListID>0
   n=FsFileListGetCount(FileListID)
   i=1
   WHILE i <= n
      datei$=FsFileListGetName(FileListID,i)
      ;
      i=i+1
   END
   FsFileListClose(FileListID)
END
```

Siehe auch:

FsFileListNew, FsFileListGetCount, FsFileListGetSize, FsFileListGetTime, FsFileListGetAttribute, FsFileListClose, FsDlgSelectFiles, FsGetLastError

FsFileListGetSize

Liefert die Größe einer Datei aus einer Dateiliste

Deklaration:

```
FsFileListGetSize ( FileListID, Index ) -> Groesse
```

Parameter:

FileListID	ID-Wert der Dateiliste
Index	Index des Eintrags in der Dateiliste. Der Index beginnt bei 1 und endet bei der Anzahl, die mit der Funktion FsFileListGetCount() ermittelt wurde.
Groesse	Dateigröße in Bytes
	>=0 : Größe der Datei in Bytes
	-1 : Fehler. Er kann mit der Funktion FsGetLastError() ermittelt werden.
	-2 : Es handelt sich bei dem Eintrag um ein Verzeichnis.

Beschreibung:

Mit dieser Funktion wird die Größe einer Datei aus einer Dateiliste ermittelt. Die Dateiliste muss vorher mit den Funktionen FsFileListNew() oder FsDlgSelectFiles() angelegt worden sein.

Ist der ID-Wert der Liste ungültig oder liegt der Index des Eintrags ausserhalb des vorhandenen Bereiches, so wird -1 zurückgegeben. Mit der Funktion FsGetLastError() kann die Fehlerursache bestimmt werden.

Handelt es sich um einen Verzeichniseintrag, so ist der Rückgabewert -2.

Beispiele:

In einem Auswahldialog können mehrere Datei ausgewählt werden.

Diese Dateien werden intern in eine Dateiliste eingelesen und anschliessend ausgelesen.

Mit der Funktion FsFileListGetSize() wird die Dateigröße ermittelt.

```
FileListID = FsDlgSelectFiles("Datei selektieren", "c:\copy", "*.*", 0)
IF FileListID>0
   n=FsFileListGetCount(FileListID)
   i=1
   WHILE i <= n
      size=FsFileListGetSize(FileListID,i)
      ;
      i=i+1
   END
   FsFileListClose(FileListID)
END
```

Siehe auch:

FsFileListNew, FsFileListGetCount, FsFileListGetName, FsFileListGetTime, FsFileListGetAttribute, FsFileListClose, FsDlgSelectFiles, FsGetLastError

FsFileListGetTime

Liefert die Zeit der letzten Modifikation einer Datei aus einer Dateiliste

Deklaration:

```
FsFileListGetTime ( FileListID, Index ) -> Zeit
```

Parameter:

FileListID	ID-Wert der Dateiliste
Index	Index des Eintrags in der Dateiliste. Der Index beginnt bei 1 und endet bei der Anzahl, die mit der Funktion FsFileListGetCount() ermittelt wurde.
Zeit	Zeit der letzten Modifikation im FAMOS-Format
	>=0 : Dateizeit
	-1 : Fehler. Er kann mit der Funktion FsGetLastError() ermittelt werden.

Beschreibung:

Die Funktion liefert die Zeit der letzten Änderung einer Datei aus einer Dateiliste.Die Dateiliste muss vorher mit den Funktionen FsFileListNew() oder FsDlgSelectFiles() angelegt worden sein.

Ist der ID-Wert der Liste ungültig oder liegt der Index des Eintrags ausserhalb des vorhandenen Bereiches, so wird -1 zurückgegeben. Mit der Funktion FsGetLastError() kann die Fehlerursache bestimmt werden.

Handelt es sich um einen Verzeichniseintrag, so wird die Erzeugungszeit des Verzeichnisses zurückgegeben.

Beispiele:

In einem Auswahldialog können mehrere Datei ausgewählt werden. Diese Dateien werden intern in eine Dateiliste eingelesen und anschliessend ausgelesen.

Mit der Funktion FsFileListGetTime() wird die Dateizeit bestimmt. Diese steht im FAMOS-Format zur Verfügung.

```
FileListID = FsDlgSelectFiles("Datei selektieren", "c:\copy", "*.*", 0)
IF FileListID>0
   n=FsFileListGetCount(FileListID)
   i=1
   WHILE i <= n
      time=FsFileListGetTime(FileListID,i)
      time$=TimeToText(time,3)
      ;
      i=i+1
   END
   FsFileListClose(FileListID)
END
```

Siehe auch:

FsFileListNew, FsFileListGetCount, FsFileListGetName, FsFileListGetSize, FsFileListGetAttribute, FsFileListClose, FsDlgSelectFiles, FsGetLastError

FsFileListNew

Einlesen von Dateien und Verzeichnissen

Deklaration:

```
FsFileListNew ( TxPathname, TxFilePattern, Auswahl, Tiefe, Sort ) -> FileListID
```

Parameter:

TxPathname	Verzeichnis
TxFilePattern	Dateimaske zum Einlesen der Dateien und Verzeichnisse
Auswahl	Auswahl, was soll eingelesen werden.
	0 : Nur Dateien
	1 : Nur Verzeichnisse
	2 : Dateien und Verzeichnisse
Tiefe	Nur Verzeichnis einlesen oder Unterverzeichnisse auch
	0 : Nur angegebenes Verzeichnis
	1 : Unterverzeichnisse einschliessen
Sort	Sortierkriterium
	0 : Unsortiert
	1 : Name (alphabetisch)
	2 : Zeit
	3 : Größe
	4 : Name (natürlich)
FileListID	ID-Wert der Dateiliste
	>=1 : Gültiger ID-Wert der Dateiliste
	-1 : Fehler. Er kann mit der Funktion FsGetLastError() ermittelt werden.

Beschreibung:

Mit dieser Funktion werden Dateien und/oder Unterverzeichnisse von einem vorgegebenen Verzeichnis ermittelt. Die Datei- und Verzeichnisnamen werden in einer Dateiliste gesammelt. In dieser Liste werden die Einträge nach verschiedenen Kriterien sortiert. Mit den FsFileList* - Funktionen kann der Inhalt einer Liste ausgelesen werden. Die Funktion gibt bei Erfolg einen ID-Wert für die Dateiliste zurück. Dieser Wert ist als Parameter bei den FsFileList* - Funktionen anzugeben.

Das Datei-Kit kann gleichzeitig maximal 20 Dateilisten verwalten. Nicht mehr benötigte Dateilisten sollten mit der Funktion FsFileListClose() geschlossen werden.

- Mit der Dateimaske können spezielle Dateien eingelesen werden. Wird bspw. "*.raw" angegeben, so werden nur Dateien mit diesen Erweiterungen erfaßt. Sollen auch Verzeichnisse erfaßt werden, so ist hier stets "*.*" anzugeben.
- Es können auch mehrere Filter angegeben werden, die durch ein Semikolon zu trennen sind. "*.raw;*.dat" findet z.B. alle Dateien, die die Erweiterung "raw" oder "dat" besitzen (ab FAMOS-Version 2022).
- Wird der Dateimaske ein "|" vorangestellt, so werden alle Dateien und Verzeichnisse zurückgegeben, die nicht der Maske entsprechen.
- Mit Tiefe kann definiert werden, ob nur die Dateien und Unterverzeichnisse des angegebenen Verzeichnisses ermittelt werden sollen, oder ob auch alle darunterliegenden Verzeichnisse durchlaufen werden sollen.
- Das Sortieren kann nach Namen,Größe oder Zeit erfolgen. Unabhängig vom definierten Kritierium, werden immer zu erst Verzeichnisnamen aufgeführt. Beim Sortieren nach der Größe werden die Dateien von der größten zur kleinsten angeordnet. Beim Sortieren nach der Zeit werden die Datei von der aktuellsten zur ältesten angeordnet.
- Bei der 'natürlichen' Sortierreihenfolge (Sort = 4) werden Ziffern im Namen über ihren Wert sortiert. Dadurch wird z.B. 'Channel2' vor 'Channel10' angeordnet. Diese Sortierung wird auch im Windows-Explorer (ab Windows 7) verwendet.
- Multithreading: Die von der Funktion zurückgegebene ID-Wert für die Dateiliste ist global (für alle Ausführungs-Threads) gültig.

Beispiele:

Mit der Funktion FsDlgSelectDirectory() wird ein Verzeichnis festgelegt, in dem alle Dateien und Verzeichnisse erfaßt werden sollen. Dabei sollen auch die Unterverzeichnisse eingeschlossen werden. Die ermittelten Einträge sollen in der Liste nach den Namen sortiert werden.

```
dir$ = FsDlgSelectDirectory("Verzeichnis", "c:\copy",0)
IF TLeng(dir$) > 0
   FileListID= FsFileListNew(dir$, "*.*", 2, 1, 1)
```

```
   n=FsFileListGetCount(FileListID)
   i=1
   WHILE i <= n
      datei$=FsFileListGetName(FileListID,i)
      ;
      i=i+1
   END
   FsFileListClose(FileListID)
END
```

Siehe auch:

FsFileListGetCount, FsFileListGetName, FsFileListGetSize, FsFileListGetTime, FsFileListGetAttribute, FsFileListClose, FsGetLastError

FsGetDiskFreeSpace

Freien Speicherplatz eines Laufwerks bestimmen

Deklaration:

```
FsGetDiskFreeSpace ( TxLaufwerk ) -> EwFreierPlatz
```

Parameter:

TxLaufwerk	Laufwerksangabe
EwFreierPlatz	Freien Speicherplatz auf dem Laufwerk

Beschreibung:

Die Funktion ermittelt den freien Speicherplatz auf einem Laufwerk.

Für den Parameter TxLaufwerk sind solche Angaben wie 'c', 'c:', 'c:\', 'c:\test', 'c:\test\datei.dat' zulässig.

Die Funktion liefert folgende Ergebnisse:

- -1 Es ist ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.
- >=0 Freier Speicherplatz in Bytes

Beispiele:

Der freie Speicherplatz auf dem Laufwerk 'D:' wird ermittelt.

```
Freespace = FsGetDiskFreeSpace("d:")
IF Freespace=-1
   error$=FsGetLastError()
END
```

Siehe auch:

FsGetLogicalDrives, FsGetDriveType

FsGetDriveType

Typ eines logischen Laufwerks ermitteln

Deklaration:

```
FsGetDriveType ( TxLaufwerk ) -> EwTyp
```

Parameter:

TxLaufwerk	Laufwerksangabe
EwTyp	Laufwerkstyp
	-1 : Fehler
	2 : Diskettenlaufwerk oder ZIP-Laufwerk
	3 : Festplatte
	4 : Netzlaufwerk
	5 : CD-ROM-Laufwerk
	6 : RAM-Laufwerk

Beschreibung:

Die Funktion ermittelt den Typ des Laufwerks.

Für den Parameter TxLaufwerk sind solche Angaben wie 'c', 'c:', 'c:\', 'c:\test', 'c:\test\datei.dat' zulässig.

Die Funktion liefert folgende Ergebnisse:

- -1 Es ist ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.
- 2 Diskettenlaufwerk oder ZIP-Laufwerk
- 3 Festplatte
- 4 Netzlaufwerk
- 5 CD-ROM-Laufwerk
- 6 RAM-Laufwerk

Beispiele:

Es wird der Typ des logischen Laufwerks 'F' ermittelt.

```
drivetyp = FsGetDriveType("f:\")
IF drivetyp=-1
   error$=FsGetLastError()
END
```

Siehe auch:

FsGetLogicalDrives, FsGetDiskFreeSpace

FsGetFileAttributes

Dateiattribute ermitteln

Deklaration:

```
FsGetFileAttributes ( TxDateiName, Attribut ) -> AnAus
```

Parameter:

TxDateiName	Dateiname
Attribut	Dateiattribut
	1 : Schreibgeschützt
	2 : Versteckt
	3 : Verzeichnis
AnAus	Dateiattribut
	-1 : Fehler
	0 : Dateiattribut ist nicht gesetzt
	1 : Dateiattribut ist gesetzt

Beschreibung:

Mit der Funktion kann getestet werden, ob bestimmte Attribute einer Datei gesetzt sind.

Die Angabe in TxDateiName darf keine Globalzeichen '*' oder '?' enthalten

Im Fehlerfall liefert die Funktion den Wert -1. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

Beispiele:

Das Schreibschutzattribut einer Datei wird ermittelt und in sein Gegenteil umgewandelt.

```
file$="\\server\copy\dirwal.h"
ro=FsGetFileAttributes(file$,1)
IF ro=1
   ro=0
ELSE
   ro=1
END
ok=FsSetFileAttributes(file$,1,ro)
```

FsGetFileNames

Einlesen von Dateien und Verzeichnissen

Deklaration:

```
FsGetFileNames ( TxPathname, TxFilePattern, Auswahl, Tiefe, Sort ) -> TxArrayFileNames
```

Parameter:

TxPathname	Verzeichnis
TxFilePattern	Dateimaske zum Einlesen der Dateien und Verzeichnisse
Auswahl	Auswahl, was soll eingelesen werden.
	0 : Nur Dateien
	1 : Nur Verzeichnisse
	2 : Dateien und Verzeichnisse
Tiefe	Nur Verzeichnis einlesen oder Unterverzeichnisse auch
	0 : Nur angegebenes Verzeichnis
	1 : Unterverzeichnisse einschliessen
Sort	Sortierkriterium
	0 : Unsortiert
	1 : Name (alphabetisch)
	2 : Zeit
	3 : Größe
	4 : Name (natürlich)
TxArrayFileNames	Text-Array mit den ermittelten Datei- und Verzeichnisnamen

Beschreibung:

Mit dieser Funktion werden Dateien und/oder Unterverzeichnisse von einem vorgegebenen Verzeichnis ermittelt. Die Datei- und Verzeichnisnamen werden in einer Dateiliste gesammelt. In dieser Liste werden die Einträge nach verschiedenen Kriterien sortiert. Die Rückgabe erfolgt in einem Text-Array.

Bei einem Fehler wird ein leeres Text-Array zurückgegeben

- Mit der Dateimaske können spezielle Dateien eingelesen werden. Wird bspw. "*.raw" angegeben, so werden nur Dateien mit diesen Erweiterungen erfaßt. Sollen auch Verzeichnisse erfaßt werden, so ist hier stets "*.*" anzugeben.
- Es können auch mehrere Filter angegeben werden, die durch ein Semikolon zu trennen sind. "*.raw;*.dat" findet z.B. alle Dateien, die die Erweiterung "raw" oder "dat" besitzen (ab FAMOS-Version 2022).
- Wird der Dateimaske ein "|" vorangestellt, so werden alle Dateien und Verzeichnisse zurückgegeben, die nicht der Maske entsprechen.
- Mit Tiefe kann definiert werden, ob nur die Dateien und Unterverzeichnisse des angegebenen Verzeichnisses ermittelt werden sollen, oder ob auch alle darunterliegenden Verzeichnisse durchlaufen werden sollen.
- Das Sortieren kann nach Namen,Größe oder Zeit erfolgen. Unabhängig vom definierten Kritierium, werden immer zu erst Verzeichnisnamen aufgeführt. Beim Sortieren nach der Größe werden die Dateien von der größten zur kleinsten angeordnet. Beim Sortieren nach der Zeit werden die Datei von der aktuellsten zur ältesten angeordnet.
- Bei der 'natürlichen' Sortierreihenfolge (Sort = 4) werden Ziffern im Namen über ihren Wert sortiert. Dadurch wird z.B. 'Channel2' vor 'Channel10' angeordnet. Diese Sortierung wird auch im Windows-Explorer (ab Windows 7) verwendet.

Beispiele:

Aus einem angegebenen Ordner werden alle Dateien geladen, die dem Muster "a*.dat" entsprechen:

```
fileNames = FsGetFileNames("c:\imc\dat","a*.dat", 0, 0, 0)
FOREACH ELEMENT name in fileNames
   FileLoad( name, "", 0)
END
```

Siehe auch:

FsGetFileSize, FsGetFileTime, FsGetFileAttributes, FsGetLastError

FsGetFileSize

Größe einer Datei bestimmen

Deklaration:

```
FsGetFileSize ( TxDateiName ) -> Groesse
```

Parameter:

TxDateiName	Dateiname
Groesse	Größe der Datei in Bytes

Beschreibung:

Die Funktion liefert die Größe einer Datei in Bytes.

Die Angabe in TxDateiName darf keine Globalzeichen '*' oder '?' enthalten

Die Funktion liefert im Fehlerfall folgende Ergebnisse:

- -1 Es ist ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.
- -2 Es handelt sich um ein Verzeichnis.

Beispiele:

Die Größe der Datei rund.raw wird bestimmt.

```
file$="1:\temp\00284\rund.raw"
size=FsGetFileSize(file$)
IF size=-1
   error$=FsGetLastError()
ELSE
   IF size=-2
      error$="Es ist ein Verzeichnis"
   END
END
```

Siehe auch:

FsFileListGetSize

FsGetFileTime

Zeit der letzten Änderung der Datei oder Erzeugungszeit eines Verzeichnisses bestimmen

Deklaration:

```
FsGetFileTime ( TxDateiName ) -> Zeit
```

Parameter:

TxDateiName	Dateiname
Zeit	Zeit im FAMOS-Format

Beschreibung:

Handelt es sich um eine Datei, so wird die Zeit der letzten Änderung der Datei ermitttelt.

Ist TxDateiName ein Verzeichnis, dann wird die Erzeugungszeit des Verzeichnisses bestimmt.

Die Rückgabe der Zeit erfolgt im FAMOS-Format.

Die Funktion liefert -1, wenn ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

Die Angabe in TxDateiName darf keine Globalzeichen '*' oder '?' enthalten

Beispiele:

Das Erzeugungsdatum des Verzeichnisses '00284' wird ermittelt und in eine Textform konvertiert.

```
file$="l:\temp\00284\"
time=FsGetFileTime(file$)
time$=TimeToText(time,3)
```

Das Änderungsdatum der Datei 'sparn01.raw' wird ermittelt und in eine Textform konvertiert.

```
file$="l:\temp\00284\sparn01.raw"
time=FsGetFileTime(file$)
IF time=-1
   error$=FsGetLastError()
ELSE
   time$=TimeToText(time,3)
END
```

FsGetLastError

Letzten Fehler einer Fs*-Funktion ermitteln

Deklaration:

```
FsGetLastError ( ) -> TxFehler
```

Parameter:

TxFehler	Fehlertext

Beschreibung:

Die Funktioen liefert einen Fehlertext zum letzten aufgetretenen Fehler zurück.

Alle Fs*-Funktionen die den Funktionswert -1 liefern, weisen daraufhin, dass ein Fehler in der Ausführung aufgetreten ist. Die genaue Fehlerursache kann durch diese Funktion abgefragt werden.

Der letzte aufgetretene Fehler wird mit dem Aufruf einer Fs*-Funktion gelöscht. Die Ausnahme bilden die Funktionen FsGetLastError() und FsGetLastErrorNumber().

Für eine programmtechnische Auswertung eines Fehlers ist die Funktion FsGetLastErrorNumber() besser geeignet. Sie liefert eine Fehlernummer zurück.

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Beispiele:

Siehe auch:

FsGetLastErrorNumber

FsGetLastErrorNumber

Letzte Fehlernummer einer Fs*-Funktion ermitteln

Deklaration:

```
FsGetLastErrorNumber ( ) -> EwFehlerNummer
```

Parameter:

EwFehlerNummer	Fehlernummer
	10001 : Kein Dateiname angegeben
	10002 : Ungültiges Dateiattribut
	10003 : Ungültige Zeitangabe
	10004 : Ungültige Option
	10005 : Ungültige Auswahl
	10006 : Ungültiges Sortierkriterium
	10007 : Ungültiger ID-Wert für eine Dateiliste
	10008 : Unültiger Index in einer Dateiliste
	10010 : Zu viele Dateilisten
	10011 : Neue Dateiliste kann nicht erzeugt werden
	10012 : Verzeichnis ist nicht leer
	10013 : Verzeichnis existiert nicht
	10014 : Das aktuelle Verzeichnis kann nicht gelöscht werden
	10015 : Root-Verzeichnisse können aus Sicherheitsgründen mit dieser Funktion nicht gelöscht werden.
	10016 : Eine relative Pfadangabe ist nicht zulässig
	10020 : Datei- oder Verzeichnisname enthält '*' oder '?'
	10021 : Datei- oder Verzeichnis existiert nicht
	10022 : Ungültige Datei- oder Verzeichnisangabe
	10023 : Zugriff auf Datei verweigert (LW ist schreibgescützt)
	10024 : Laufwerk ist nicht bereit
	10025 : Datenträger ist voll
	10026 : Ungültige Pfadangabe
	10027 : Datei existiert schon
	10028 : Laufwerk existiert nicht
	10029 : Zielname muß ein existierendes Verzeichnis sein.
	10030 : Der Prozess kann nicht auf die Datei zugreifen, da sie von einem anderen Prozess verwendet wird.

Beschreibung:

Die Funktion liefert die Nummer des letzten aufgetretenen Fehlers zurück.

Alle Fs*-Funktionen die den Funktionswert -1 liefern, weisen daraufhin, dass ein Fehler in der Ausführung aufgetreten ist. Die genaue Fehlerursache kann durch diese Funktion abgefragt werden.

Der letzte aufgetretene Fehler wird mit dem Aufruf einer Fs*-Funktion gelöscht. Die Ausnahme bilden die Funktionen FsGetLastError() und FsGetLastErrorNumber().

Wird ein fehlerbeschreibender Text benötigt, so ist die Funktion FsGetLastError() zu verwenden

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Beispiele:

Siehe auch:

FsGetLastError

FsGetLogicalDrives

Logische Laufwerke ermitteln

Deklaration:

```
FsGetLogicalDrives ( ) -> TxLaufwerke
```

Parameter:

TxLaufwerke	Aufzählung der Laufwerksbuchstaben

Beschreibung:

Die Funktion ermittelt alle vorhandenen logischen Laufwerke des Rechners .

Die entsprechenden Laufwerksbuchstaben werden als Zeichenkette zurückgegeben.

Beispiele:

Nach Ausführung der Anweisung enthählt drives beispielsweise folgenden Inhalt 'ACDEFGHIJKLMN'.

```
drives = FsGetLogicalDrives()
```

Siehe auch:

FsGetDriveType, FsGetDiskFreeSpace

FsGetLongPathName

Konvertierung eines kurzen Dateinamen in einen langen

Deklaration:

```
FsGetLongPathName ( TxShortName ) -> TxLongName
```

Parameter:

TxShortName	Kurzer Dateiname
TxLongName	Langer Dateiname

Beschreibung:

Die Funktion konvertiert einen kurzen Datei- oder Verzeichnisnamen in den langen Namen.

Voraussetzung für den Erfolg der Funktion ist das Existieren der Datei bzw. des Verzeichnisses.

Die Funktion liefert eine leere Zeichenkette, wenn ein Fehler aufgetreten ist. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

In der Angabe für den kurzen Dateinamen dürfen keine Globalzeichen '*' oder '?' enthalten sein.

Es muß ein vollständiger Dateiname angegeben werden.

Beispiele:

Ein kurzer Dateiname wird in seinen langen Dateinamen umgewandelt.

Ergebnis: "e:\kit32\Filekit\filemanager.h".

```
file$="e:\kit32\filekit/filema~1.h"
longname$=FsGetLongPathName(file$)
```

Siehe auch:

FsGetShortPathName

FsGetParentDirectoryName

Elternverzeichnis zu einem Verzeichnis ermitteln

Deklaration:

```
FsGetParentDirectoryName ( TxDirName ) -> TxParentDirName
```

Parameter:

TxDirName	Verzeichnisname
TxParentDirName	Name des Elternverzeichnisses

Beschreibung:

Die Funktion ermittelt zu dem übergebenen Verzeichnisnamen das Elternverzeichnis.

Das Verzeichnis muss existieren. Im Fehlerfall wird eine leere Zeichenkette zurückgegeben.

Beispiele:

Zum Verzeichnis "d:\imc\dat\" soll das Elternverzeichnis ermittelt werden.

```
parentdir$=FsGetParentDirectoryName("d:\imc\dat\")
```

Die Variable parentdir$ enthält die Zeichenkette "d:\imc". Der nächste Aufruf ergibt "d:\"

```
parentdir$=FsGetParentDirectoryName(parentdir$)
```

Alle weiteren Aufrufe würden jetzt "d:\" ergeben.

Siehe auch:

FsGetShortPathName

FsGetShortPathName

Konvertierung eines langen Dateinamen in einen kurzen

Deklaration:

```
FsGetShortPathName ( TxLongDateiname ) -> TxShortDateiname
```

Parameter:

TxLongDateiname	Langer Dateiname
TxShortDateiname	Kurzer Dateiname

Beschreibung:

Die Funktion konvertiert einen langen Datei- oder Verzeichnisnamen in den zugehörigen kurzen Dateinamen.

Der kurze Dateiname ist im MSDOS 8.3 - Format.

Die Funktion liefert eine leere Zeichenkette, wenn ein Fehler aufgetreten ist. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

In der Angabe für den langen Dateinamen dürfen keine Globalzeichen '*' oder '?' enthalten sein.

Es muß ein vollständiger Dateiname angegeben werden

Beispiele:

Der lange Verzeichnisname "c:\multimedia files\music\" ergibt nach der Konvertierung in den kurzen Namen "c:\multim~1\music\".

```
dir$="c:\multimedia files\music\"
short$=FsGetShortPathName(dir$)
```

FsMakePath

Vollständigen Dateinamen bilden

Deklaration:

```
FsMakePath ( TxLaufwerk, TxDirectory, TxName, TxExtension ) -> TxDateiname
```

Parameter:

TxLaufwerk	Laufwerksangabe
TxDirectory	Verzeichnisangabe
TxName	Dateiname
TxExtension	Dateierweiterung
TxDateiname	Vollständiger Dateiname

Beschreibung:

Diese Funktion bildet aus den Komponenten eines Dateinamens einen vollständigen Dateinamen.

Folgende Parameter in TxLaufwerk ergeben das gleiche Ergebnis:

- 'c:\'
- 'c:'
- 'c'

In TxDirectory ist es egal, ob die Zeichenkette mit einem Backslash beginnt bzw. endet.

Die Dateierweiterung kann mit oder ohne führenden Punkt angegeben werden.

Beispiele:

Der Dateiname 'c:\imc\bin\famos.exe' soll zusammengesetzt werden:

```
drive$="c:"
dir$="\imc\bin"
name$="famos"
ext$="exe"
file$=FsMakePath(drive$,dir$,name$,ext$)
```

Siehe auch:

FsSplitPath

FsMoveFile

Dateien und Verzeichnisse verschieben

Deklaration:

```
FsMoveFile ( TxQuelldatei, TxZieldatei, Option, Tiefe ) -> Status
```

Parameter:

TxQuelldatei	Quelldateiname
TxZieldatei	Zieldateiname
Option	Option
	0 : Existierende Dateien nicht überschreiben
	1 : Nur neuere Dateien verschieben
	2 : Existierende Dateien immer überschreiben
Tiefe	Nur Verzeichnis oder auch Unterverzeichnisse verschieben
	0 : Nur Inhalt des Verzeichnisses verschieben
	1 : Unterverzeichnisse einschliessen
Status	Erfolg der Funktion
	0 : Funktion war erfolgreich
	-1 : Fehler ist aufgetreten. Fehlerursache kann mittels FsGetLastError() ermittelt werden.

Beschreibung:

Diese Funktion verschiebt Dateien oder Verzeichnisse. Es kann mit dem Parameter Tiefe festgelegt werden, ob auch alle Unterverzeichnisse verschoben werden sollen.

Angabe des Quelldateinamens

- Der Name darf keine relative Pfadangabe sein.
- Der Name darf Globalzeichen enthalten.
- Wird ein eindeutiger Verzeichnis- bzw. Dateiname angegeben, so muß dieser existieren
- Entspricht ein eindeutige Name einer Datei, so ist der Parameter Tiefe ohne Bedeutung.

Angabe des Zieldateinamens

- Der Name darf keine relative Pfadangabe sein.
- Der Name darf keine Globalzeichen enthalten.
- Werden im Quellnamen Globalzeichen verwendet, so muß der Zielname ein existierendes Verzeichnis sein
- Ist die Quelle ein eindeutiger Dateiname und das Ziel ein existierendes Verzeichnis, so wird der Dateiname in das Ziel übernommen.

Beispiele:

Eine Datei 'H:\Temp\2001.dat' soll nach 'c:\archiv\2001.raw' verschoben und umbenannt werden. Wenn die Datei 'c:\archiv\2001.raw' bereits existiert, erfolgt kein Verschieben.

```
erg=FsMoveFile("H:\Temp\2001.dat","c:\archiv\2001.raw",0,0)
IF erg=-1
   error$=FsGetLastError()
END
```

Alle Dateien mit der Erweiterung *.raw aus dem Verzeichnis 'H:\neu' sollen in das Verzeichnis d:\a\0003 verschoben werden.Es werden in diesem Fall nur neuere Dateien verschoben.

```
erg=FsMoveFile("H:\neu\*.raw","d:\a\0003",1,0)
IF erg=-1
   error$=FsGetLastError()
END
```

Das Verzeichnis 'H:\Temp\' inklusive aller Unterverzeichnisse und Dateien wird in das Verzeichnis e:\dat\0003 verschoben.

```
erg=FsMoveFile("H:\Temp\","e:\dat\0003",0,1)
IF erg=-1
   error$=FsGetLastError()
```

```
END
```

Siehe auch:

FsCopyFile, FsRenameFile, FsGetLastError

FsPathCombine

Kombiniert zwei Zeichenfolgen zu einem Pfad

Deklaration:

```
FsPathCombine ( TxPathName1, TxPathName2 ) -> TxPathName
```

Parameter:

TxPathName1	Erster zu kombinierender Pfadname
TxPathName2	Zweiter zu kombinierender Pfadname
TxPathName	Der zusammengesetzte Pfad

Beschreibung:

Die Funktion kombiniert zwei Pfandangaben zu einem Pfad

Wenn einer der beiden angegebenen Pfadnamen eine leere Zeichenfolge ist, gibt die Funktion den anderen Pfadnamen zurück.

Wenn TxPathName2 einen absoluten Pfad enthält, gibt die Funktion TxPathName2 zurück.

TxPathName2 kann als relative Pfadangabe auch mit ".." oder "." beginnen:

```
FsPathCombine("c:\folder1\folder2", ".\folder3")  => "c:\folder1\folder2\folder3"
FsPathCombine("c:\folder1\folder2", "..\folder3") => "c:\folder1\folder3"
```

Wenn TxPathName2 mit einem "\" beginnt, wird dies als Wurzelverzeichnis des aktuellen Laufwerks (durch TxPathName1 vorgegeben) interpretiert:

```
FsPathCombine("c:\folder1\folder2", "folder3")  => "c:\folder1\folder2\folder3"
FsPathCombine("c:\folder1\folder2", "\folder3") => "c:\folder3"
```

Im Fehlerfall ist die zurückgegebene Zeichenkette leer.

Beispiele:

FsPathCombine("","") => \

FsPathCombine("abc", "") => "abc"

FsPathCombine("", "abc") => "abc"

FsPathCombine("c:\abc\", "data\file.xml") => "c:\abc\data\file.xml"

FsPathCombine("c:\abc", ".xml") => "c:\abc\.xml"

FsPathCombine("\\ww\copy\", "\lib") => "\\ww\copy\lib"

FsPathCombine("\\ww\copy\something\else", "\lib") => "\\ww\copy\lib"

FsPathCombine("\\ww\copy\", "\\lib") => "\\lib"

FsPathCombine("lib","c:\ab\datei.exe")=> "c:\ab\datei.exe

Siehe auch:

FsMakePath

FsRemoveDirectory

Verzeichnis löschen

Deklaration:

```
FsRemoveDirectory ( TxDirName, Option ) -> Status
```

Parameter:

TxDirName	Verzeichnisname
Option	Option
	0 : Nur leeres Verzeichniss
	1 : Verzeichnis komplett mit Dateien und Unterverzeichnissen
Status	Erfolg der Funktion
	0 : Funktion war erfolgreich
	-1 : Fehler ist aufgetreten. Fehlerursache kann mittels FsGetLastError() ermittelt werden.

Beschreibung:

Das angegebene Verzeichnis wird gelöscht.

Die Angabe in TxDirName darf keine Globalzeichen '*' oder '?' enthalten

Das aktuelle Verzeichnis kann nicht gelöscht werden.

Root-Verzeichnisse ("c:\") können aus Sicherheitsgründen nicht gelöscht werden.

Mit der Option 0 wird das Verzeichnis nur gelöscht, wenn es leer ist.

Die Option 1 löscht den Inhalt des Verzeichnisses einschließlich seiner Unterverzeichnisse und Dateien. Schreibgeschützte Dateien in diesen Verzeichnissen werden auch gelöscht. Anschließend wird das Verzeichnis selbst gelöscht.

Die Funktion liefert folgende Ergebnisse:

- 0 Das Löschen des Verzeichnisses war erfolgreich
- -1 Es ist ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

Beispiele:

Das Verzeichnis '00110' soll gelöscht werden. Mit der Option 0 geschieht das nur, wenn dieses leer ist.

```
dir$="\\server\imcan\temp\00110"
result=FsRemoveDirectory(dir$,0)
IF result<>0
   error$=FsGetLastError()
END
```

Das Verzeichnis '00110' und sein gesamter Inhalt soll gelöscht werden. Durch die Option 1 geschieht das unabhängig, ob im Verzeichnis schreibgeschützte Dateien sind.

```
dir$="\\server\imcan\temp\00110"
result=FsRemoveDirectory(dir$,1)
IF result<>0
   error$=FsGetLastError()
END
```

Siehe auch:

FsCreateDirectory, FsGetLastError

FsRenameFile

Datei oder Verzeichnis umbenennen

Deklaration:

```
FsRenameFile ( TxDateiname, TxNeuerName ) -> Status
```

Parameter:

TxDateiname	Bestehender Dateiname
TxNeuerName	Neuer Dateiname
Status	Erfolg der Funktion

Beschreibung:

Eine bestehende Datei oder Verzeichnis wird umbenannt.

Für Verzeichnisse gilt; sie können nur umbenannt werden.

Dateien können gleichzeitig verschoben werden.

In den Angaben für beide Namen dürfen keine Globalzeichen '*' oder '?' enthalten sein.

Für die Angabe TxDateiname gilt:

- Die Datei oder das Verzeichnis müssen existieren
- Die Pfadangabe darf nicht relativ sein.

Für die Angabe TxNeuerName gilt:

- Eine Datei oder Verzeichnis mit diesem Namen darf es noch nicht geben
- Wird ein Dateiname ohne Pfadangabe angegeben, dann wird die neue Datei im Verzeichnis der alten Datei erzeugt.

Die Funktion liefert folgende Ergebnisse:

- 0 Das Umbenennen war erfolgreich
- -1 Es ist ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

Beispiele:

Umbennen einer Datei im Datenverzeichnis von FAMOS.

```
existfile$="d:\imc\dat\b.dat"
newfile$="a.dat"
result=FsRenameFile(existfile$,newfile$)
IF result=-1
   error$=FsGetLastError()
END
```

Umbennen und Verschieben einer Datei.

```
existfile$="c:\test\b.dat"
newfile$="e:\histroy\file28.dat"
result=FsRenameFile(existfile$,newfile$)
IF result=-1
   error$=FsGetLastError()
END
```

Umbennen eines Verzeichnisses.

```
dir$="c:\test"
newdir$="c:\histroy"
result=FsRenameFile(dir$,newdir$)
IF result=-1
   error$=FsGetLastError()
END
```

Siehe auch:

FsMoveFile, FsGetLastError

FsSetFileAttributes

Dateiattribute verändern

Deklaration:

```
FsSetFileAttributes ( TxDateiName, Attribut, AnAus ) -> Status
```

Parameter:

TxDateiName	Dateiname
Attribut	Dateiattribut
	1 : Schreibgeschützt
	2 : Versteckt
AnAus	Attribut setzen oder löschen
	0 : Löschen
	1 : Setzen
Status	Erfolg der Funktion
	0 : Funktion war erfolgreich
	-1 : Fehler ist aufgetreten. Fehlerursache kann mittels FsGetLastError() ermittelt werden.

Beschreibung:

Mit dieser Funktion können die Dateiattribute Schreibschutz und Versteckt verändert werden.

Die Angabe in TxDateiName darf keine Globalzeichen '*' oder '?' enthalten

Die Funktion liefert folgende Ergebnisse:

- 0 Das Ändern des Attributes war erfolgreich
- -1 Es ist ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.

Beispiele:

Das Schreibschutzattribut einer Datei wird ermittelt und in sein Gegenteil umgewandelt.

```
file$="\\server\copy\dirwal.h"
ro=FsGetFileAttributes(file$,1)
IF ro=1
   ro=0
ELSE
   ro=1
END
ok=FsSetFileAttributes(file$,1,ro)
```

FsSetFileTime

Dateizeit ändern

Deklaration:

```
FsSetFileTime ( TxDateiName, Zeit ) -> Status
```

Parameter:

TxDateiName	Dateiname
Zeit	Zeit im FAMOS-Format
Status	Erfolg der Funktion
	0 : Funktion war erfolgreich
	-1 : Fehler ist aufgetreten. Fehlerursache kann mittels FsGetLastError() ermittelt werden.
	-2 : Die Angabe ist ein Verzeichnis.

Beschreibung:

Mit dieser Funktionen kann die Zeit der letzten Änderung einer Datei modifiziert werden.

Die Dateizeit wird auch geändert, wenn die Datei schreibgeschützt ist.

Auf Verzeichnisse kann die Funktion nicht angewendet werden.

Die Angabe in TxDateiName darf keine Globalzeichen '*' oder '?' enthalten

Die Funktion liefert folgende Ergebnisse:

- 0 Das Ändern der Dateizeit war erfolgreich
- -1 Es ist ein Fehler aufgetreten. Die Fehlerursache kann mit der Funktion FsGetLastError() ermittelt werden.
- -2 Es handelt sich um ein Verzeichnis.

Beispiele:

Die Zeit der letzten Änderung der Datei 'bisch01.raw' wird um eine Stunde verändert.

```
file$="1:\temp\00284\bisch01.raw"
time=FsGetFileTime(file$)
time=time+3600
ok=FsSetFileTime(file$,time)
IF ok<>0
   error$=FsGetLastError()
END
```

FsSplitPath

Vollständigen Dateinamen zerlegen

Deklaration:

```
FsSplitPath ( TxPfadname, Option ) -> TxTeilName
```

Parameter:

TxPfadname	Vollständiger Dateiname
Option	Teil des Namens, der soll ermittelt werden.
	0 : Laufwerk
	1 : Verzeichnis
	2 : Dateiname ohne Extension
	3 : Extension
	4 : Dateiname + Extension
	5 : Verzeichnis + Dateiname + Extension
	6 : Letzter Teil der Verzeichnisangabe
	7 : UNC-Share-Name
	8 : Laufwerk + Verzeichnis
TxTeilName	Teilname

Beschreibung:

Mit dieser Funktion können Sie aus einer Zeichenkette, die einen vollständigen Pfadnamen (also Laufwerksbezeichnung, Verzeichnisangabe und Dateiname) enthält, verschiedene Teile des Namens abgerufen werden.

Die Optionen liefern beispielsweise für 'c:\imc\bin\famos.exe':

- 0 c:
- 1 \imc\bin\
- 2 famos
- 3 .exe
- 4 famos.exe
- 5 \imc\bin\famos.exe
- 6 bin
- 8 c:\imc\bin\

Mit der Option 7 kann der UNC-Pfad eines lokalen Laufwerkes gebildet werden. Sogenannte "UNC-Pfade" kommen immer dann ins Spiel, wenn Ressourcen im Netz (auf einem "File-Server") benutzt werden sollen. Einem öfter benötigtem Server-Verzeichnis wird der Einfachheit halber oft ein Laufwerk zugeordnet (z.B. als "Arbeitsgruppen-Laufwerk", im Beispiel N:). Dann kann z.B. statt mit der unübersichtlichen UNC-Notation \\SERVER\TEMP\ einfach über ein "virtuelles" Laufwerk zugegriffen werden: N:\ .

```
unc_name = FsSplitPath("n:\temp", 7)
unc_name enthält \\SERVER\TEMP\TEMP\
```

Beispiele:

Die folgenden Anweisungen zerlegen den Dateinamen file$.Die Ergebnisse sind wie oben angezeigt.

```
file$="c:\imc\bin\famos.exe"
drive$=FsSplitPath(file$,0)
dir$=FsSplitPath(file$,1)
name$=FsSplitPath(file$,2)
ext$=FsSplitPath(file$,3)
nameext$=FsSplitPath(file$,4)
path$=FsSplitPath(file$,5)
subdir$=FsSplitPath(file$,6)
```

Siehe auch:

FsMakePath

FsTempFileName

Temporären Dateinamen bilden

Deklaration:

```
FsTempFileName ( Option ) -> TxFileName
```

Parameter:

Option	Option
	0 : Kompletter Pfadname
	1 : Dateiname + Erweiterung
	2 : Nur Verzeichnis
TxFileName	Dateiname oder Verzeichnisname

Beschreibung:

Diese Funktion bildet temporäre Datei- bzw. Verzeichnisnamen. Die Beispiele zeigen die Ergebnisse der einzelnen Optionen.

Mit der Option 0 wird im temporären Verzeichnis von Windows ein eindeutiger temporärer Dateiname gebildet.

Die Option 1 ist mit der Option 0 identisch. Es wird aber nur der Dateiname ohne Pfadangabe zurückgegeben.

Bei Option 0 und 1 wird eine leere Datei mit dem angegebenen Namen erzeugt! Diese wird bei Programmende nicht automatisch gelöscht, sollte also ggf. nach Verwendung mit FsDeleteFile() gelöscht werden.

Die Option 2 liefert den Pfad des temporären Verzeichnis von Windows.

Ist die zurückgegebene Zeichenkette leer, so ist ein Fehler aufgetreten. Die Fehlerursache kann mittels FsGetLastError() ermittelt werden.

Beispiele:

Die Optionen liefern folgende Ergebnisse:

```
tempfile$=FsTempFileName(0)
```

Ergebnis: C:\WINDOWS\TEMP\FAM2062.TMP

```
tempname$=FsTempFileName(1)
```

Ergebnis: FAM2063.TMP

```
tempdir$=FsTempFileName(2)
```

Ergebnis: C:\WINDOWS\TEMP\

Siehe auch:

GetLastError

Letzten Fehler abfragen.

Deklaration:

```
GetLastError ( ) -> TxFehler
```

Parameter:

TxFehler	Letzter Fehler.

Beschreibung:

Durch den Befehl OnError() kann die Fehlerbehandlung in Sequenzen so konfiguriert werden, dass die Sequenzausführung trotz Fehler zunächst fortgesetzt wird. Die Fehlerursache wird dann intern vermerkt und kann mit dieser Funktion abgefragt werden.

Einige Funktionen, insbesondere solche aus Erweiterungsbibliotheken (Kits) melden nicht-kritische Fehler durch einen speziellen Rückgabewert. Der zugehörige Fehlertext kann dann ebenfalls mit dieser Funktion ermittelt werden.

Nach dem Aufruf von GetLastError() ist der interne Fehlerspeicher gelöscht.

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Beispiele:

Die Sequenz 'DoSomething' ist durch einen OnError("Return")-Befehl so konfiguriert, dass beim Auftreten eines Fehlers an das Ende der Sequenz gesprungen wird. Die aufrufende Sequenz prüft, ob ein Fehler aufgetreten ist, und zeigt diesen gegebenenfalls in einer Meldungsbox an.

```
OnError("ResumeNext")
SEQUENCE DoSomething
err = GetLastError()
IF err <> ""
   BoxMessage("Fehler in DoSomething", err, "!1")
END
```

Eine Panel-Datei wird per Kit-Funktion geladen. Wenn die Datei nicht geladen werden kann, wird eine Meldung mit der Fehlerursache angezeigt.

```
ok = PnLoad("d:\templates\result.panel")
IF ok <> 0
   BoxMessage("Fehler", GetLastError(), "!1")
END
```

Siehe auch:

OnError, ThrowError

GetOption

Abfragen diverser aktueller Voreinstellungen, wie z.B. Standardverzeichnisse und Vorgaben für mathematische Funktionen.

Deklaration:

```
GetOption ( TxOptionsName ) -> TxOptionsWert
```

Parameter:

TxOptionsName	Name der gewünschten Option, siehe Liste.
	"Dir.DataFiles" : Standardverzeichnis zum Laden und Speichern von Meßwert-Dateien. Verwendet von den Befehlen LOAD, SAVE u.ä. sowie den Funktionen FileOpenDSF() und FileOpenFAS().
	"Dir.Sequences" : Aktuelles Standardverzeichnis zum Laden von Sequenzen. Verwendet vom Befehl SEQUENCE. Dieses steht beim Aufstart von imc FAMOS zunächst auf dem unter "Optionen"/ "Verzeichnisse" vereinbarten Verzeichnis. Es kann mit dem Befehl MDIR oder der Funktion SetOption() umgesetzt werden. Ansonsten das Verzeichnis, aus dem zuletzt eine Sequenz geladen wurde.
	"Dir.CurrentProject" : Wenn ein Projekt aktiv ist, wird das zugehörige Projektverzeichnis zurückgegeben, ansonsten ein leerer Text.
	"Dir.CurrentSequence" : Enthält während der Ausführung einer Sequenz oder eines Dialoges das Verzeichnis, aus dem die Sequenz bzw. der Dialog geladen wurde. Dies entspricht dem aktuellen FAMOS-Arbeitsverzeichnis. Ansonsten wird ein leerer Text zurückgeliefert.
	"Dir.DefinitionFiles" : Liefert das Standardverzeichnis für Definitionsdateien. Dazu zählen z.B. ASCII-/EXCEL-Exportvorlagen (*.aet), mit dem Datei-Assistenten erstellte Importfilter (*.fas) und Definitionsdateien für externe DLL-Funktionen (*.def).
	"Func.WarnLevel" : Legt fest, welche Warnungen, die bei der Ausführung von Funktionen auftreten, angezeigt werden sollen
	"Func.NoInfoMessages" : Einige Funktionen (Stat, LFit) geben standardmäßig ihre Ergebnisse im Ausgabefenster aus. Mit dieser Option kann diese Ausgabe unterbunden werden.
	"Func.FFT.Window" : Legt die FFT Fensterfunktion fest. Verwendet von den Funktionen FFT(), Spec() u.ä.
	"Func.FFT.Mode" : Der implementierte FFT-Algorithmus verlangt, dass die Länge des Ausgangsdatensatzes eine 2er-Potenz ist. Diese Option legt fest, wie bei anderen Datensatzlängen verfahren wird.
	"Func.ResultFormat" : Legt das Datenformat fest, in dem Funktionen standardmäßig Ihre Ergebnisse zurückliefern.
	"Func.ErrorBoxes" : Einige Funktionen können im Fehlerfall wahlweise entweder eine Fehlerbox erzeugen oder den Fehler (still) durch Ihren Rückgabewert signalisieren. Betrifft bisher nur die Gruppe der Dateifunktionen FileOpenDSF() u.ä.
	"DLLImport.DefinitionFile" : Dateiname mit Definitionen für externe DLL-Funktionen. Die hier gelisteten Funktionen werden über die allgemeine DLL-Schnittstelle in FAMOS importiert und können in Sequenzen verwendet werden. Die Datei muss über den Dialog 'Optionen'/'DLL-Funktionen registrieren' erzeugt worden sein.
	"Display.DecimalSeparator" : Legt den Dezimaltrenner für die Anzeige reeller Zahlen fest.
	"DDE.Text.NumFormat" : Legt das Zahlenformat beim DDE-Senden im Text-Format fest.
	"DDE.Text.Delimeter" : Legt das (die) Trennzeichen fest, welche(s) beim DDE-Senden im Text-Format zwischen 2 Zahlen verwendet wird.
	"DDE.TimeOut" : Legt die maximale Zeit fest, die FAMOS bei der DDE-Kommunikation mit einer anderen Applikation auf eine Antwort wartet.
	"Units.Ctrl.Compatible" :
	"Units.Display.Greek" :
	"Units.Display.Ohm" :
	"Units.Create.Delim" :
	"Units.Create.Nm" :
	"Units.Create.Pow.1/2" :
	"Units.Create.Pow.2" :
	"Units.Create.Pow.3" :
	"Units.Create.Pow.Neg" :
	"Units.Create.u" :
	"Units.Create.Num.Space" :

	"Units.Create.1e3" :
	"Units.Create.1e-3" :
	"Units.Create./s" :
	"Units.Read.cal" :
	"Units.Read.Exp" :
	"Units.Read.g" :
	"Units.Read.Gs" :
	"Units.Read.hp" :
	"Units.Read.kt" :
	"Units.Read.L" :
	"Units.Read.lb" :
	"Units.Read.oz" :
	"Units.Read.pt" :
	"Units.Read.quot" :
	"Units.Read.s" :
	"Units.Read.ton" :
	"Units.Read.u" :
	"Units.Reduce.C" :
	"Units.Reduce.F" :
	"Units.Reduce.H" :
	"Units.Reduce.J" :
	"Units.Reduce.Ohm" :
	"Units.Reduce.Pa" :
	"Units.Reduce.S" :
	"Units.Reduce.T" :
	"Units.Reduce.Wb" :
TxOptionsWert	Wert der aktuellen Einstellung. Eine Liste der möglichen Text-Konstanten, bezogen auf die jeweilige Vorgabe für den Parameter, finden Sie in der Beschreibung der Funktion SetOption().

Beschreibung:

Diese Funktion gestattet das Abfragen des Zustandes diverser Voreinstellungen, die bei der Ausführung von Befehlen und mathematischen Funktionen berücksichtigt werden.

Wenn eine Einstellung nicht explizit durch den Befehl SetOption() gesetzt wurde, wird die globale Vorgabe verwendet (Dialoge "Optionen/Verzeichnisse", "Optionen/Funktionen",...).

Beispiele:

Das aktuell verwendete Standard-Datenverzeichnis wird ermittelt. Das Unterverzeichnis "Results" zu diesem Verzeichnis wird als neue Vorgabe eingestellt und Dateien in dieses Verzeichnis gesichert.

```
folder = GetOption("Dir.DataFiles")
SetOption("Dir.DataFiles", folder + "\results")
;...Abspeichern der Ergebnisse
SAVE result "result.dat"
```

Siehe auch:

SetOption, FFTOPTION, MDIR, LDIR, SDIR

GetScale

Verfügbar ab: Professional Edition

Abfrage der Skalierung

Deklaration:

```
GetScale ( Variable, Eigenschaft ) -> Wert
```

Parameter:

Variable	Variable
Eigenschaft	Welche Eigenschaft?
	"factor" : Skalierungsfaktor
	"offset" : Skalierungsoffset
	"max" : Maximal darstellbarer Wert
	"min" : Minimal darstellbarer Wert
	"minEx" : Erweiterter minimal darstellbarer Wert
Wert	Wert

Beschreibung:

Skalierungsfaktor und Skalierungsoffset gehorchen dieser Formel:

[Physikalischer Wert] = [Ganze Zahl] * Skalierungsfaktor + Skalierungsoffset

Skalierungsfaktor und Skalierungsoffset gibt es nur bei ganzzahligen Formaten.

Minimum und Maximum sind nicht die im Datensatz wirklich vorhandenen Werte, sondern die mittels des vorliegenden Datenformates minimal bzw. maximal darstellbar sind.

Der Skalierungsfaktor stellt den Wert von 1 LSB (least significant bit) dar. Das wird auch als Auflösung bezeichnet.

Für die Berechnung des Minimums wird bei ganzen Zahlen mit Vorzeichen der negative maximale Wert der ganzen Zahl zur Berechnung verwendet, also z.B. -127 bei 1 Byte mit Vorzeichen. Kompatibel zu SetDatFormat().

Für die Berechnung des erweiterten Minimums wird bei ganzen Zahlen mit Vorzeichen der um 1 verringerte negative maximale Wert der ganzen Zahl zur Berechnung verwendet, also z.B. -128 bei 1 Byte mit Vorzeichen.

Ist ein Datensatz, der aus 2 Komponenten besteht, angegeben, so wird bei XY entsprechend .Y, bei RI entsprechend .R und bei BP entsprechend .B. benutzt. Der Anwender kann im Aufruf selbst die Komponente auswählen, indem z.B. .Y angehängt wird.

Der Zahlenwert wird ohne Einheit zurückgegeben.

Bei Timestamp ASCII Daten wird die Information zu den Zeitstempeln erteilt.

Bei reellen Zahlen werden -1e35 und 1e35 als minimale und maximale Werte zurückgegeben.

Es gibt Datensätze mit mehreren Events, bei denen die Eigenschaften für jedes Event einen anderen Wert aufweisen können. So z.B. die explizite Zeitspur bei Daten aus imc DEVICES bzw. imc STUDIO. Dann kann die Eigenschaft nur für ein ausgewähltes Event abgefragt werden.

Beispiele:

LSB eines gemessenen Kanals, der im Format 2 Byte ganze Zahl mit Vorzeichen abgelegt ist

```
LSB = GetScale ( Channel_01, "factor")
```

Offset (verschobene Mittellage) der Y-Spur eines XY Kanals

```
Offset = GetScale ( data.y, "offset")
```

Siehe auch:

SetDatFormat, DatFormat?

GetSystemInfo

Abfragen von Informationen zu Programm und Betriebssystem.

Deklaration:

```
GetSystemInfo ( TxInfo, TxParameter ) -> AktuellerWert
```

Parameter:

TxInfo	Name der gewünschten Information, siehe Liste.
	"Famos.Version" : Liefert die aktuelle FAMOS-Programmversion als ganze Zahl, wobei die Hauptversion die Hunderter-Stelle und die Nebenversion die beiden letzten Ziffern (ggf. mit führender Null) bildet, z.B. also 603 für FAMOS 6.3. Die Revisionsnummer wird nicht berücksichtigt.
	"Famos.VersionString" : Liefert die aktuelle FAMOS-Programmversion als Text in der Form "HAUPTVERSION.NEBENVERSION Rev. REVISIONSNUMMER", also z.B. "6.3 Rev. 2".
	"Famos.Edition" : Liefert eine Zahl, die die aktuell ausgeführte Edition beschreibt. -1: Runtime, 0: Reader, 1: Standard, 2: Professional, 3: Enterprise.
	"Famos.IsX64" : Liefert eine 1, wenn es sich um die FAMOS x64-Version handelt. Eine 0 zeigt an, dass es sich um die x86-Version (32 Bit) handelt.
	"Famos.IsKitAvailable" : Liefert eine 1, wenn die im 2. Parameter spezifizierte FAMOS-Erweiterungsbibliothek vorhanden ist und verwendet werden kann. 0 sonst. Das gesuchte Kit kann entweder über seinen Dateinamen (ohne Verzeichnis) oder über die folgenden Kürzel angegeben werden: "CLS": Klassier-Kit. "SPC"; Spektralanalyse-Kit. "OTR": Ordertracking-Kit. "VPL": Videoplayer-Kit. "ODS": ODS-Browser-Kit. "RKT": R-Kit. "RWY": Railway-Kit.
	"Famos.IsDLLFunctionAvailable" : Liefert eine 1, wenn der im 2. Parameter spezifizierte Funktionsname eine zur Zeit in FAMOS registrierte externe DLL-Funktion ist, 0 sonst. Die über die allgemeine DLL-Schnittstelle in FAMOS eingebundenen Funktionen werden über den Dialog 'Optionen'/'DLL-Funktionen registrieren' oder per Funktion SetOption("DLLImport.DefinitionFile",...) registriert.
	"Famos.IsSeqFunctionAvailable" : Liefert eine 1, wenn der im 2. Parameter spezifizierte Funktionsname eine zur Zeit in FAMOS registrierte Sequenzfunktion ist, 0 sonst. Es muss der komplette Funktionsname (inklusive vorangestelltem '!') angegeben werden. Siehe Beispiel #2. FAMOS sucht in den unter 'Optionen'/'Bibliotheken' angegebenen Verzeichnissen nach Sequenzbibliotheken (Dateierweiterung .sqf) und registriert alle enthaltenen Sequenzfunktionen (diese werden in der Funktionsliste im Ordner 'Bibliotheken' angezeigt).
	"Famos.Path.SampleProjects" : Liefert das Verzeichnis für die bei der FAMOS-Installation mitgelieferten Beispiel-Projekte. Bei einer Standardinstallation ist dies das Verzeichnis "C:\Users\Public\Documents\imc\imc FAMOS_Demo Projects". [Unterstützt ab V2023]
	"Famos.Path.SampleData" : Liefert das Verzeichnis für die bei der FAMOS-Installation mitgelieferten Beispiel-Messwertdateien. Bei einer Standardinstallation ist dies das Verzeichnis "C:\Users\Public\Documents\imc\imc FAMOS_Demo Projects_Demo Files\dat". [Unterstützt ab V2023]
	"System.Path.AppData" : Liefert das Windows-Standardverzeichnis für applikations-spezifische Daten. Ein typischer Pfad ist "C:\Dokumente und Einstellungen\USERNAME\Anwendungsdaten" unter Windows XP oder "C:\Users\USERNAME\AppData\Roaming" unter Windows 7.
	"System.Path.CommonAppData" : Liefert das Windows-Standardverzeichnis für applikations-spezifische Daten, die für alle Anwender verfügbar sind. Ein typischer Pfad ist "C:\Dokumente und Einstellungen\All Users\Anwendungsdaten" unter Windows XP oder "C:\ProgramData" unter Windows 7.
	"System.Path.Documents" : Liefert das Windows-Standardverzeichnis für Anwender-Dokumente. Ein typischer Pfad ist "C:\Dokumente und Einstellungen\USERNAME\Eigene Dateien" unter Windows XP oder "C:\Users\USERNAME\Documents" unter Windows 7.
	"System.Path.CommonDocuments" : Liefert das Windows-Standardverzeichnis für Dokumente, die gemeinsam für alle Anwender sind. Ein typischer Pfad ist "C:\Dokumente und Einstellungen\All Users\Gemeinsame Dokumente" unter Windows XP oder "C:\Users\Public\Documents" unter Windows 7.
	"System.Net.HostName" : Liefert den Standard-Host-Namen des lokalen Computers
	"System.Net.IPAddr1" : Liefert die primäre IPv4-Adresse des Computers
	"System.Net.IPAddr*" : Liefert alle IPv4-Adressen des Computers, durch '|' getrennt. Sie können die Funktion TxSplit() verwenden, um die einzelnen Adressen zu isolieren.
	"System.Path.TempFiles" : Liefert das Windows-Standardverzeichnis für temporäre Dateien. Ein typischer Pfad ist "C:\Dokumente und Einstellungen\USERNAME\Lokale Einstellungen\Temp" unter Windows XP oder "C:\Users\USERNAME\AppData\Local\Temp" unter Windows 7.
	"System.EnvVariable" : Liefert den Wert einer Windows-Umgebungsvariablen als Text. Der Name der Variablen ist im 2. Parameter der Funktion anzugeben. Es werden maximal 256 Zeichen zurückgegeben, wenn der Inhalt der Variablen länger ist, wird abgeschnitten. Wenn die Variable nicht existiert, wird ein leerer Text zurückgegeben.

	"Device.LogicalProcessors" : Liefert die Anzahl der logischen Prozessoren (Kerne) im System.
	"Screen.VirtualArea" : Liefert die Größe des virtuellen Bildschirms. Der virtuelle Bildschirm ermittelt sich aus der vereinigten Größe aller vorhandenen Monitore. Für den zweiten Parameter können die Text-Konstanten "left" (links), "top" (oben), "width" (Breite) bzw. "height" (Höhe) angegeben werden, die Funktion ermittelt dann die entsprechende Koordinate/Abmessung.
	"Screen.MonitorCount" : Liefert die Anzahl der vorhandenen Monitore.
	"Screen.PrimaryWorkArea" : Liefert die Größe des Arbeitsbereich des primären Monitors (Hauptanzeige). Der Arbeitsbereich entspricht der Bildschirmgröße abzüglich des benötigten Platzes für die Windows-Taskleiste. Für den zweiten Parameter können die Text-Konstanten "left" (links), "top" (oben), "width" (Breite) bzw. "height" (Höhe) angegeben werden, die Funktion ermittelt dann die entsprechende Koordinate/Abmessung.
	"Screen.WorkArea#1" : Liefert die Koordinaten des Arbeitsbereiches des 1. Monitors. Der Arbeitsbereich entspricht der Bildschirmgröße abzüglich des benötigten Platzes für die Windows-Taskleiste (wenn angezeigt). Für den zweiten Parameter können die Text-Konstanten "left" (links), "top" (oben), "width" (Breite) bzw. "height" (Höhe) angegeben werden, die Funktion ermittelt dann die entsprechende Koordinate/Abmessung.
	"Screen.WorkArea#2" : Liefert die Koordinaten des Arbeitsbereiches des 2. Monitors. Der Arbeitsbereich entspricht der Bildschirmgröße abzüglich des benötigten Platzes für die Windows-Taskleiste (wenn angezeigt). Für den zweiten Parameter können die Text-Konstanten "left" (links), "top" (oben), "width" (Breite) bzw. "height" (Höhe) angegeben werden, die Funktion ermittelt dann die entsprechende Koordinate/Abmessung.
	"Screen.WorkArea#3" : Liefert die Koordinaten des Arbeitsbereiches des 3. Monitors.
TxParameter	Bei "System.EnvVariable" der Name der gewünschten Variable, bei "Screen.*" der Bezeichner der gewünschten Größe, sonst ein leerer Text.
AktuellerWert	Wert der abgefragten Information. Zahl oder Text.

Beschreibung:

Bei der Abfrage der Systempfade wird der tatsächliche Name im Dateisystem zurückgegeben. Bei der Anzeige im Windows-Explorer sind diese Pfade unter Umständen nicht sichtbar (weil sie bei Standardeinstellungen versteckt werden) oder sie werden unter einem anderen (lokalisierten) Namen angezeigt. Beispiel Windows 7 (deutsch): der Ordner "C:\Users" wird im Explorer unter dem lokalisierten Namen "C:\Anwender" angezeigt.

Beispiele:

Abfrage des Standard-Ordners für Dokumente:

```
MyDocFolder = GetSystemInfo("System.Path.Documents", "")
```

Abfrage der Umgebungsvariable USERNAME, enthält den Namen des aktuellen Benutzerkontos.

```
CurrentUser = GetSystemInfo("System.EnvVariable", "USERNAME")
```

Test, ob die Funktionen des Ordertracking-Kits verwendet werden können. Beide Aufrufe sind gleichwertig:

```
ok = GetSystemInfo("FAMOS.IsKitAvailable","OTR")
ok = GetSystemInfo("FAMOS.IsKitAvailable","im7otrk1.dll")
```

Eine Sequenz verwendet mehrere Sequenzfunktionen aus der Sequenz-Bibliothek "MyFilters". Am Beginn der Sequenz wird geprüft, ob die Funktionen zur Laufzeit tatsächlich vorhanden sind und ggf. eine aussagekräftige Fehlermeldung generiert. Ohne diesen Check würde der Formelinterpreter bei der Aufrufzeile eine generische Fehlermeldung der Form "Unbekanntes Objekt" liefern, die für den Anwender nicht aussagekräftig ist.

```
ok = GetSystemInfo("Famos.IsSeqFunctionAvailable", "!MyFilters.HP_200Hz")
ok = ok AND GetSystemInfo("Famos.IsSeqFunctionAvailable", "!MyFilters.LP_100Hz")
IF NOT(ok)
   BoxMessage("Fehler", "Die Sequenzbibliothek 'MyFilters.sqf' ist entweder nicht vorhanden oder veraltet!", "!1")
   EXITSEQUENCE
END
```

Siehe auch:

GetOption

GrChanAppend

Neuen Kanal einer Gruppe hinzufügen
Alternativer Name: **GrKanalAdd**

Deklaration:

```
GrChanAppend ( Gruppe, NeuerKanal )
```

Parameter:

Gruppe	Datengruppe, zu der ein Kanal hinzugefügt werden soll.
NeuerKanal	Datensatz oder Text, der der Gruppe zugefügt werden soll.

Beschreibung:

Die Prozedur fügt der angegebenen Datengruppe eine Kopie des spezifizierten Datensatzes oder Textes hinzu. Dazu wird dieser Datensatz oder Text dupliziert und in der Inhaltsliste der Gruppe eingetragen.

Statt dieser Funktion kann in vielen Anwendungsfällen auch eine direkte Zuweisung benutzt werden:

```
ExistierendeGruppe:NeuerKanal = ...
```

Beispiele:

Eine neue Datengruppe wird erzeugt und ein Datensatz und ein Text zugefügt:

```
newGroup = GrNew()
d1 = Ramp(0,1,100)
t1 = "Date: 1.1.1995")
GrChanAppend(newGroup, d1)
GrChanAppend(newGroup, t1)
; oder auch:
; newGroup:d1 = Ramp(0,1,100)
; newGroup:t1 = "Date: 1.1.1995"
```

Siehe auch:

GrChanDel, GrNew, GrChanNum?, GrChanName?, GrChanFind

GrChanDel

Kanal einer Datengruppe löschen
Alternativer Name: **GrKanalEntf**

Deklaration:

```
GrChanDel ( Gruppe, KanalIdent )
```

Parameter:

Gruppe	Datengruppe, aus der ein Kanal zu löschen ist.
KanalIdent	Index (1..) oder Name des zu löschenden Kanals.

Beschreibung:

Der angegebene Datensatz oder Text wird aus der Gruppe gelöscht. Falls er nicht existiert (ungültiger Index, Name nicht vorhanden), wird eine Warnung ausgegeben.

Statt dieser Funktion kann auch das Kommando ENTFERNE benutzt werden.

Beispiele:

Die Datengruppe Messung_01 enthält an zweiter Position den Kanal c2, der gelöscht werden soll. Die folgenden Aufrufe sind gleichwertig:

```
GrChanDel(Messung_01, 2)
GrChanDel(Messung_01, "c2")
DELETE Messung_01:c2
DELETE Messung_01:[2]
```

Aus einer Datengruppe werden alle Kanäle entfernt, deren Maximum kleiner als 100 ist:

```
i = GrChanNum?(MEASUREMENT_01)
WHILE i >= 1
   channelMax= max(MEASUREMENT_01:[i])
   IF channelMax < 100
      GrChanDel(MEASUREMENT_01, i)
   END
   i = i - 1
END
```

Siehe auch:

GrChanAppend, GrNew, GrChanNum?, GrChanName?, GrChanFind, DELETE

GrChanFind

Suche nach einem Kanal mit einem gegebenen Namen in einer Datengruppe
Alternativer Name: **GrKanalFinde**

Deklaration:

```
GrChanFind ( Gruppe, TxKanalname ) -> EwKanalIndex
```

Parameter:

Gruppe	Datengruppe, deren Kanäle zu durchsuchen sind
TxKanalname	Name des gesuchten Datensatzes oder Textes.
EwKanalIndex	Index des Kanals in der Gruppe (1..) oder 0, wenn nicht gefunden

Beschreibung:

In der angegebenen Gruppe wird ein Datensatz oder Text mit dem spezifizierten Namen gesucht. Bei Erfolg wird der Index des Kanals zurückgegeben, sonst 0.

Beim Verändern der Gruppenstruktur durch Einfügen oder Löschen von Kanälen wird der Index ungültig, die Zuordnung Kanalname <-> Index muss dann neu ermittelt werden.

Beispiele:

In einer Datengruppe wird nach einem Kanal mit der Bezeichnung "MessKanal_123" gesucht. Bei erfolgreicher Suche wird der Kanal angezeigt, ansonsten wird eine Fehlermeldung generiert.

```
Index = GrChanFind(Messung011_A, "MessKanal_123")
IF Index = 0
   BoxMessage("Achtung", "Kanal existiert nicht!", "!1")
ELSE
   SHOW Messung011_A:[Index]
   ;oder auch: SHOW Messung011_A:MessKanal_123
END
```

Siehe auch:

GrChanAppend, GrChanDel, GrNew, GrChanNum?, GrChanName?

GrChanName?

Abfrage des Namens eines Kanals einer Datengruppe
Alternativer Name: **GrKanalName?**

Deklaration:

```
GrChanName? ( Gruppe, EwKanalIndex ) -> TxKanalName
```

Parameter:

Gruppe	Datengruppe, aus der ein Kanalname zu bestimmen ist.
EwKanalIndex	Index des gewünschten Kanals (1..).
TxKanalName	Name des Kanals

Beschreibung:

Die Funktion liefert den Namen des angegebenen Datensatzes oder Textes in der Gruppe.

Der erste Kanal in der Gruppe wird mit dem Index 1 adressiert.

Alternativ kann auch die leistungsfähigere Funktion Name?() verwendet werden.

Beispiele:

Eine Datengruppe wird aus einer bestehenden Datengruppe gebildet, indem sowohl die Originalkanäle als auch die durch Glättung der Originalkanäle entstandenen Datensätze eingefügt werden. Der Name des geglätteten Datensatzes wird aus dem Originalnamen durch Anhängen einer festen Endung gebildet.

```
channelCount = GrChanNum?(MEASUREMENT_01)
RESULT_01 = GrNew()
FOR i = 1 TO channelCount
   TxName = GrChanName?(MEASUREMENT_01, i)
   TxName2 = TxName + "_SMO"
   RESULT_01:<TxName> = MEASUREMENT_01:[i]
   ; oder auch GrChanAppend(RESULT_01, MEASUREMENT_01:[i])
   RESULT_01:<TxName2> = Smo(MEASUREMENT_01:[i], 10)
END
```

Siehe auch:

Name?, GrChanAppend, GrChanDel, GrNew, GrChanNum?, GrChanFind

GrChanNum?

Abfrage der Kanalzahl einer Datengruppe
Alternativer Name: **GrKanalZahl?**

Deklaration:

```
GrChanNum? ( Gruppe ) -> EwKanalZahl
```

Parameter:

Gruppe	Datengruppe, deren Kanalzahl zu bestimmen ist.
EwKanalZahl	Anzahl der enthaltenen Kanäle

Beschreibung:

Die Funktion liefert die Anzahl der Kanäle in einer Datengruppe. Es werden alle enthaltenen Datensätze und Texte gezählt.

Falls der Parameter keine Gruppe darstellt, wird -1 zurückgegeben.

Eine typische Anwendung ist die Bestimmung der Kanalzahl für eine anschließende Aufzählung (FOR, WHILE) der Kanäle. Hier kann alternativ auch die FOREACH CHANNEL - Anweisung verwendet werden.

Beispiele:

In einer Schleife über alle Kanäle einer Datengruppe werden alle Datensätze angezeigt, deren Maximum größer als 100 ist.

```
channelCount = GrChanNum?(MESSUNG_01)
i = 1
WHILE i <= channelCount
   channelMax= max(MESSUNG_01:[i])
   IF channelMax > 100
      SHOW MESSUNG_01:[i]
   END
   i = i + 1
END
```

Siehe auch:

GrChanAppend, GrChanDel, GrNew, GrChanName?, GrChanFind, FOREACH

GrConcat

Verbindet die angegebenen Parameter-Variablen zu einer neuen Datengruppe.

Deklaration:

```
GrConcat ( P1 [, P2] [, P3] [, P4] [, P5] [, P6] [, P7] [, P8] [, P9] [, P10] [, P11] [, P12] [, P13] [, P14] [, P15] ) -> Gruppe
```

Parameter:

P1	1. Parameter
P2	2. Parameter (optional)
P3	3. Parameter (optional)
P4	4. Parameter (optional)
P5	5. Parameter (optional)
P6	6. Parameter (optional)
P7	7. Parameter (optional)
P8	8. Parameter (optional)
P9	9. Parameter (optional)
P10	10. Parameter (optional)
P11	11. Parameter (optional)
P12	12. Parameter (optional)
P13	13. Parameter (optional)
P14	14. Parameter (optional)
P15	15. Parameter (optional)
Gruppe	Neue Gruppe. Die enthaltenen Elemente sind Kopien der angegebenen Parameter.

Beschreibung:

Die Funktion verbindet die angegebenen Parameter zu einer neuen Datengruppe. Die neue Datengruppe enthält Kopien der angegebenen Parameter-Variablen, der jeweilige Variablen-Name wird als Element-Name übernommen.

Für die Parameter sind beliebige Datentypen erlaubt, außer bei Datengruppen sind nur direkt angegebene Variablen erlaubt (keine Indizierungen, keine Zwischenergebnisse). Bei Datengruppen werden die enthaltenen Elemente expandiert und deren Kopien in die Ergebnisgruppe übernommen.

Die Reihenfolge der Elemente in der erzeugten Gruppe entspricht der Reihenfolge der Parameter.

Wenn der Name einer anzuhängenden Variablen in der Ergebnisgruppe bereits vorhanden ist, so wird das gleichnamige Element in der Ergebnisgruppe überschrieben, wobei die Position erhalten bleibt. Bei Namensdopplungen "gewinnt" also der zuletzt angegebene Parameter. Dadurch kann die Funktion auch verwendet werden, um in einer existierenden Gruppe mehrere Elemente auszutauschen. Siehe Beispiel #3.

Beispiele:

An eine bestehende Gruppe sollen 3 weitere Kanäle angehängt werden:

```
a_fft = FFT(gr:a)
b_fft = FFT(gr:b)
c_fft = FFT(gr:c)
grNew = GrConcat(gr, a_fft, b_fft, c_fft)
```

Bei einer Gruppe mit vielen Kanälen soll die FFT eines jeden Kanals berechnet werden.

```
GrFFTs = FFT(GrInput, 0, 0)
```

Diese Berechnung soll durch Parallelisierung der Ausführung beschleunigt werden. Die Datengruppe wird dazu in 4 Teile aufgespalten, die parallel verrechnet werden. Die 4 Teilergebnisse werden dann wieder zur einer Ergebnis-Datengruppe zusammengefügt.

```
count = GrChanNum?(GrInput)
chunksize = floor(count/4)
BEGIN_PARALLEL
   GrFFT1 = !MyFFT(GrPart(GrInput, 1,              chunksize))
   GrFFT2 = !MyFFT(GrPart(GrInput, 1+ chunksize,   chunksize))
   GrFFT3 = !MyFFT(GrPart(GrInput, 1+ chunksize*2, chunksize))
   GrFFT4 = !MyFFT(GrPart(GrInput, 1+ chunksize*3, count-3*chunksize))
```

```
END_PARALLEL
GrFFTs = GrConcat(GrFFT1, GrFFT2, GrFFT3, GrFFT4)
```

Die Sequenzfunktion !MyFFT ist dabei wie folgt definiert:

```
; Deklaration:
; !MyFFT(Par [Datentyp: Normal]) => Result [Datentyp: Komplex]
Result = FFT(Par, 0, 0)
```

Achtung bei gleichnamigen Parametern, der "letzte" gewinnt:

```
a = 1
g = GrConcat(a, a)
; Die Ergebnisgruppe hat nur einen Kanal "a".
g:b = 1
b = 2
g2 = GrConcat(g, b)
; Die Ergebnisgruppe hat die Kanäle "a" und "b", wobei "b" den Wert 2 hat.
g3 = GrConcat(b, g)
; Die Ergebnisgruppe hat die Kanäle "b" und "a", wobei "b" den Wert 1 hat.

; Austausch von mehreren Kanälen in einer existierenden Gruppe:
g4:a = 2
g4:b = 1
g4:c = 1
g5:b = 2
g5:c = 2
g = GrConcat(g4, g5)
; Die Ergebnisgruppe enthält nun 3 Kanäle, die alle den Wert 2 haben.
```

Siehe auch:

GrNew, GrChanAppend, GrPart

Unterstützt ab:

Version 2022

GrExpand

Die in einer Datengruppe enthaltenen Kanäle werden expandiert.

Deklaration:

```
GrExpand ( Gruppe )
```

Parameter:

Gruppe	Zu expandierende Datengruppe

Beschreibung:

Die in einer Datengruppe enthaltenen Kanäle werden in einzelne Variablen expandiert und die Gruppenvariable anschließend gelöscht.

Wenn bereits Variablen mit dem gleichen Namen existieren, werden diese überschrieben.

Das Verhalten der Funktion ist identisch zum Menübefehl "Variable"/"Expandieren".

Beispiele:

Mehrere Variablen sollen gleichartigen Berechnungen unterzogen wegen. Um Schreibaufwand zu sparen, werden zunächst alle diese Variablen in einer Datengruppe zusammengefasst, diese dann wie gewünscht bearbeitet und anschließend die Gruppe wieder aufgelöst.

```
Temp:channel1 = channel1
Temp:channel2 = channel2
Temp:channel3 = channel3
Temp = CutIndex(Temp, 100, 300)
Temp = Smo(Temp, 0.2)
; weitere Berechnungen...
GrExpand(Temp)
```

Siehe auch:

GrChanAppend, GrChanDel, GrChanName?, GrChanFind

GrJoin

Aneinanderbinden der Kanäle einer Gruppe
Alternativer Name: **GrBinde**

Deklaration:

```
GrJoin ( Gruppe, Null ) -> Datensatz
```

Parameter:

Gruppe	Datengruppe, deren Kanäle zu verbinden sind.
Null	Reservierter Parameter, immer 0.
Datensatz	Datensatz, der aus der Aneinanderreihung der Kanäle der Gruppe besteht.

Beschreibung:

Die einzelnen Kanäle einer Gruppe werden zu einem neuen Datensatz verbunden. Texte werden dabei übersprungen. Kennwerte und Datenformat des Ergebnisses richten sich nach dem ersten Datensatz in der Gruppe. Alle Datensätze in der Gruppe müssen den gleichen Datentyp aufweisen.

Beispiele:

Bestimmung des absoluten Maximums über alle Kanäle einer Gruppe:

```
AbsMaximum = max(GrJoin(max(myGroup), 0))
```

Siehe auch:

GrNew, GrChanAppend, GrChanNum?, GrChanName?, GrChanFind

GrNew

Erzeugen einer leeren Datengruppe
Alternativer Name: **GrNeu**

Deklaration:

```
GrNew ( ) -> NeueGruppe
```

Parameter:

NeueGruppe	Leere Datengruppe

Beschreibung:

Die Funktion erzeugt eine leere Gruppe. Dieser können dann im weiteren Kanäle zugefügt werden.

Es ist auch möglich, eine neue Gruppe und den ersten Kanal zugleich mit einer Zuweisung zu erzeugen:

```
NeueGruppe:ErsterKanal = ...
```

Beispiele:

Eine neue Gruppe wird erzeugt und 2 Kanäle zugewiesen:

```
Messung1 = GrNew()
GrChanAppend(Messung1, Kanal1_1)
GrChanAppend(Messung1, Kanal1_2)
```

Die folgenden Formeln führen zum gleichen Ergebnis:

```
Messung1:Kanal1_1 = Kanal1_1
Messung1:Kanal1_2 = Kanal1_2
```

Siehe auch:

GrChanAppend, GrChanDel, GrChanNum?, GrChanName?, GrChanFind

GrPart

Ein Abschnitt einer Datengruppe wird in eine neue Datengruppe kopiert.

Deklaration:

```
GrPart ( GrQuelle, EwIndex, EwAnzahl ) -> GrTeilKopie
```

Parameter:

GrQuelle	Datengruppe, deren Elemente kopiert werden sollen.
EwIndex	Index des ersten zu kopierenden Elements. Das erste Element hat den Index 1.
EwAnzahl	Anzahl der zu kopierenden Elemente. -1, wenn bis zum letzten Element kopiert werden soll.
GrTeilKopie	Neu erzeugte Gruppe mit Kopien der angegebenen Elemente.

Beschreibung:

Die Funktion erzeugt eine neue Datengruppe, die eine Kopie eines Abschnitts der Parameter-Datengruppe ist.

Wenn für [EwIndex] eine -1 angegeben wird oder ([EwIndex]+[EwAnzahl]) größer ist als die Anzahl der Elemente in der Quell-Datengruppe, wird bis zum Ende kopiert.

Beispiele:

Aus einer Datengruppe mit mehr als 4 Elementen werden die beiden ersten und die beiden letzten Elemente in eine neue Datengruppe kopiert.

n = GrChanNum?(grSource)

grNew = GrConcat(GrPart(grSource, 1, 2), GrPart(grSource, n-1, 2))

Bei einer Gruppe mit vielen Kanälen soll die FFT eines jeden Kanals berechnet werden.

```
GrFFTs = FFT(GrInput, 0, 0)
```

Diese Berechnung soll durch Parallelisierung der Ausführung beschleunigt werden. Die Datengruppe wird dazu in 4 Teile aufgespalten, die parallel verrechnet werden. Die 4 Teilergebnisse werden dann wieder zur einer Ergebnis-Datengruppe zusammengefügt.

```
count = GrChanNum?(GrInput)
chunksize = floor(count/4)
BEGIN_PARALLEL
   GrFFT1 = !MyFFT(GrPart(GrInput, 1,              chunksize))
   GrFFT2 = !MyFFT(GrPart(GrInput, 1+ chunksize,   chunksize))
   GrFFT3 = !MyFFT(GrPart(GrInput, 1+ chunksize*2, chunksize))
   GrFFT4 = !MyFFT(GrPart(GrInput, 1+ chunksize*3, count-3*chunksize))
END_PARALLEL
GrFFTs = GrConcat(GrFFT1, GrFFT2, GrFFT3, GrFFT4)
```

Die Sequenzfunktion !MyFFT ist dabei wie folgt definiert:

```
; Deklaration:
; !MyFFT(Par [Datentyp: Normal]) => Result [Datentyp: Komplex]
Result = FFT(Par, 0, 0)
```

Siehe auch:

GrConcat

Unterstützt ab:

Version 2022

Histo

Histogramm mit vorgebbarer Unterteilungsanzahl und -breite

Deklaration:

```
Histo ( Daten, EwBreite, EwAnzahl ) -> Histogramm
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND],[XY]
EwBreite	Breite einer Histogramm-Schublade
EwAnzahl	Anzahl der Histogramm-Schubladen
Histogramm	Ermitteltes Histogramm

Beschreibung:

Die Histogrammfunktion ermittelt die Häufigkeit von Amplitudenwerten (Werten eines Datensatzes). Der mögliche Bereich der Werte wird in eine Anzahl von Schubladen einer bestimmten Breite eingeteilt. Jeder auftretende Wert des Datensatzes wird in eine dieser Schubladen einsortiert. Das Histogramm gibt an, wie viele Werte in jede Schublade einsortiert wurden.

Der zweite Parameter gibt die Breite einer Schublade an. Wenn der Wert gleich Null ist, wird die Breite der Schubladen automatisch bestimmt.

Der dritte Parameter gibt die Anzahl der Schubladen an. Wenn der Wert gleich Null ist, wird die Anzahl der Schubladen automatisch bestimmt.

Bei einer automatischen Bestimmung werden die zu Null übergebenen Parameter aus dem Wertebereich des Datensatzes bestimmt.

Die Ränder einer Schublade liegen stets auf einem ganzzahligen Vielfachen der Schubladenbreite.

- Der erzeugte Datensatz hat keine y-Einheit, seine x-Einheit entspricht der y-Einheit des übergebenen Datensatzes.
- Bei vollautomatischer Histogramm-Erstellung wird die Schubladenzahl auf etwa 100 gesetzt.
- Die Schubladenanzahl ist ohne Beschränkung der Allgemeinheit auf 10000 begrenzt. Üblicherweise werden Werte zwischen einigen 10 und einigen 100 gewählt.
- Wenn Sie die Schubladenbreite und -anzahl von Hand fest vorgeben, kann es passieren, dass es Werte im Datensatz gibt, die in keine Schublade einsortiert werden können. Die Summe aller Werte des Histogramms ist dann kleiner als die Länge des übergebenen Datensatzes. Wenn Sie möchten, dass solche weit außerhalb liegenden Werte des Datensatzes noch in die Schubladen am Rand einsortiert werden, benutzen Sie die Funktion Band() zum Einschränken des y-Wertebereichs vor Anwenden der Funktion Histo. Benutzen Sie die Funktionen Min() und Max(), um den Wertebereich einer Funktion zu ermitteln, die Funktion Floor(), um diese Werte zu runden. Daraus können Sie dann die Schubladenbreite ableiten.

Beispiele:

```
NDhisto = Histo(NDdata, 0, 0)
```

Vollautomatische Histogramm-Erstellung

```
NDhisto1 = Histo(NDstrom1, 0.1 'A', 0)
NDhisto2 = Histo(NDstrom2, 0.1 'A', 0)
```

Es liegen die beiden Datensätze NDstrom1 und NDstrom2 vor, deren Stromwerte in Spitzen bis etwa 10A reichen. Die beiden Histogramme sind zu vergleichen. Zweckmäßigerweise wird eine sinnvolle Schubladenbreite vorgegeben, so dass etwa 100 Schubladen erzeugt werden. Die genaue Schubladenanzahl soll automatisch bestimmt werden. Die beiden Histogramme sind direkt vergleichbar, wo sich die Wertebereiche der beiden Strom-Datensätze überschneiden.

```
NDhisto = Histo(NDdata, 1 'V', 200)
```

Der Wertebereich des Datensatzes ist bekannt, er erstreckt sich von etwa -50V bis etwa +130V. Es ist ein Histogramm zu erstellen mit 200 Schubladen der Breite 1V.

Siehe auch:

ClsTimeAtLevel, Clip, Min, Max

HttpGetFile

Lädt die angeforderte Resource herunter und speichert diese in einer lokalen Datei.

Deklaration:

```
HttpGetFile ( TxURL, TxFileName ) -> EwStatus
```

Parameter:

TxURL	Identifikator der gewünschten Web-Resource.
TxFileName	Kompletter Dateiname, unter dem die Resource zu speichern ist.
EwStatus	Status. 1 bei Erfolg, 0 bei Fehler.

Beschreibung:

Die Funktion ruft die angegebene Resource ab, die über ihre URL (Uniform Resource Locator) angegeben wird und speichert diese in einer lokalen Datei.

Die URL besteht im Allgemeinen aus den folgenden Teilen:

- Protokoll (unterstützt sind "http://" oder "https://")
- Adresse des Servers (z.B. "www.testserver.com")
- (Optionaler) Pfad zur Resource (z.B. "/weather")
- (Optionale) Abfrage-Parameter ('Query', üblicherweise angegeben als Liste von Wertepaaren in der Form "?Wert1=Inhalt@Wert2=Inhalt2@...")

An den angegebenen Server wird ein 'HTTP GET'-Befehl gesendet, der als Parameter den Pfad und die Abfrage-Parameter enthält.

Die als Antwort empfangenen Daten werden unverändert in der angegebenen Datei gespeichert.

Im Fehlerfall (Rückgabewert = 0) kann die Fehlerursache mit der Funktion GetLastError() erfragt werden.

Beispiele:

Lädt die aktuelle "Microsoft News"-Seite bei Twitter herunter und speichert diese als lokale HTML-Datei:

```
uri ="https://twitter.com/MSFTnews"
status = HttpGetFile(uri, "c:\temp\LatestMSNews.html")
```

Lädt ein Bild aus dem Bildarchiv des Hubble-Teleskops herunter:

```
uri = "https://cdn.spacetelescope.org/archives/images/screen/heic1501a.jpg"
status = HttpGetFile(uri, "c:\temp\_nebula.jpg")
```

Siehe auch:

HttpGetText, HttpOption

HttpGetText

Lädt die angeforderte Resource als Text herunter.

Deklaration:

```
HttpGetText ( TxURL ) -> TxReturn
```

Parameter:

TxURL	Identifikator der gewünschten Web-Resource.
TxReturn	Empfangener Text

Beschreibung:

Die Funktion ruft die angegebene Resource ab, die über ihre URL (Uniform Resource Locator) angegeben wird.

Die URL besteht im Allgemeinen aus den folgenden Teilen:

- Protokoll (unterstützt sind "http://" oder "https://")
- Adresse des Servers (z.B. "www.testserver.com")
- (Optionaler) Pfad zur Resource (z.B. "/weather")
- (Optionale) Abfrage-Parameter ('Query', üblicherweise angegeben als Liste von Wertepaaren in der Form "?Wert1=Inhalt@Wert2=Inhalt2@...")

An den angegebenen Server wird ein 'HTTP GET'-Befehl gesendet, der als Parameter den Pfad und die Abfrage-Parameter enthält.

Nach dem Herunterladen werden die empfangenen Daten in einen Text konvertiert.

Im Fehlerfall wird ein leerer Text geliefert, die Fehlerursache kann mit der Funktion GetLastError() erfragt werden.

Beispiele:

Mit dem nachstehenden Aufruf wird aus der Angabe von Längen- und Breitengrad die zugehörige Adresse geliefert (reverse Geocoding).

Dazu wird die Suchmaschine von OpenStreetMap verwendet. Der Parameter 'format' spezifiziert das gewünschte Format der Antwort (hier XML, eine Alternative wäre JSON), die Parameter 'lat' (latitude) und 'lon' (longitude) geben Längen- und Breitengrad an.

```
answer = HttpGetText( "http://nominatim.openstreetmap.org/reverse?format=xml&lat=52.541861&lon=13.3869693")
```

Der gelieferte Text enthält u.a. den folgenden Abschnitt:

```
<road>Voltastraße</road>
<suburb>Gesundbrunnen</suburb>
<city_district>Mitte</city_district>
<state>Berlin></state>
<postcode>13355></postcode>
```

Um aus solchen strukturierten Antworten einzelne Felder zu extrahieren, empfiehlt sich die Funktion TxRegExMatch():

```
NameOfTheRoad = TxRegExMatch( answer, "<road>(.*?)</road>", ",", 0, 1)
; NameOfTheRoad hat jetzt den Inhalt 'Voltastraße'.
```

Wenn die Anwort im JSON-Format geliefert wird:

```
answer = HttpGetText( "http://nominatim.openstreetmap.org/reverse?format=json&lat=52.541861&lon=13.3869693")
;answer enthält u.a.: ...{... "road":"Voltastraße","suburb":"Gesundbrunnen", ...}
NameOfTheRoad = TxRegExMatch( answer, "~034road~034:~034(.*?)~034", ",", 0, 1)
```

Siehe auch:

HttpGetFile, HttpOption

HttpOption

Die Funktion setzt Optionen für nachfolgende Aufrufe von HttpGetText() oder HttpGetFile().

Deklaration:

```
HttpOption ( TxName, TxValue ) -> EwStatus
```

Parameter:

TxName	Name der Option
	"" : Alle Optionen werden auf ihren Standardzustand zurückgesetzt. Der 2. Parameter muss ebenfalls leer sein.
	"Timeout" : Timeout für Verbindung. Der für den 2. Parameter angegebene Text enthält die Angabe als Zahl in Millisekunden. Mit "0" wird der Standardwert wieder hergestellt. Dieser beträgt 20s.
	"User" : Benutzername für Webserver mit HTTP-Basisauthentifizierung (nach RFC 2016).
	"Password" : Passwort für Webserver mit HTTP-Basisauthentifizierung (nach RFC 2016).
	"HEAD:???" : Setzt ein Feld im HTTP Request Header. '???' steht für den Namen eines Feldes. Die Felder und ihre Bedeutung sind durch RFC 2616 definiert. Ist der 2. Parameter ein leerer Text, so wird das entsprechende Feld entfernt.
TxValue	Wert der Option. Der erlaubte Inhalt ist abhängig vom ersten Parameter.
EwStatus	Status. 1 bei Erfolg, 0 bei Fehler.

Beschreibung:

Die hier festgelegten Einstellungen bleiben gültig bis:

- expliziter Änderung durch einen erneuten Aufruf der Funktion HttpOption()
- nächster Neustart einer Top-Level-Sequenz
- Menübefehl 'Neubeginn'.

Im Fehlerfall (Rückgabewert = 0) kann die Fehlerursache mit der Funktion GetLastError() erfragt werden.

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Beispiele:

Mit dem folgenden Befehl wird der Server bei nachfolgenden Aufrufen von HttpGetText() oder HttpGetFile() angewiesen, die Daten bzw. Dateien vor dem Senden gegebenenfalls zu aktualisieren und nicht serverseitig zwischengespeicherte und damit möglicherweise veraltete Daten zu verwenden.

```
status = HttpOption("HEAD:Cache-Control", "no-store, max-age=0")
```

Siehe auch:

HttpGetText, HttpGetFile

Hyst

Anwendung einer Hysterese auf einen Datensatz

Deklaration:

```
Hyst ( Daten, EwBreite ) -> Filtrat
```

Parameter:

Daten	Zu filternder Datensatz [ND].
EwBreite	Hysteresebreite (>= 0)
Filtrat	Gefilteter Datensatz.

Beschreibung:

Eine Hysterese mit der eingegebenen Hysteresebreite wird auf den Datensatz [Daten] angewendet. Die Hysterese filtert kleinere Zwischenschwingungen des Datensatzes heraus. Durch den Parameter EwBreite wird die Hysteresebreite eingegeben. Die Vorgehensweise bei der Anwendung der Hysterese wird folgend erläutert:

Der Startwert und der nächste Funktionswert vom Datensatz [Daten] werden im Datensatz [Filtrat] gespeichert.

Ist der zweite Funktionswert größer als der Startwert, steigt die Funktion an und befindet sich in einem Aufwärtstrend. Ist der zweite Funktionswert kleiner als der Startwert, fällt die Funktion und befindet sich in einem Abwärtstrend. Sind die beiden Funktionswerte gleich, wird der nächste Funktionswert vom Datensatz [Daten] im Datensatz [Filtrat] gespeichert und gegebenenfalls der Trend bestimmt usw.

Befindet sich die Funktion im Aufwärtstrend, wird je nach Größe des aktuellen Funktionswerts vom Datensatz [Daten] nach einer der drei folgenden Möglichkeiten vorgegangen:

- Der aktuelle Funktionswert vom Datensatz [Daten] ist größer als der vorige Funktionswert vom Datensatz [Filtrat]. Der aktuelle Funktionswert vom Datensatz [Daten] wird als aktueller Funktionswert im Datensatz [Filtrat] gespeichert.
- Der aktuelle Funktionswert vom Datensatz [Daten] ist kleiner oder gleich dem vorigen Funktionswert vom Datensatz [Filtrat], aber nicht kleiner als der vorige Funktionswert vom Datensatz [Filtrat] abzüglich der Hysteresebreite. Der vorige Funktionswert vom Datensatz [Filtrat] wird auch als aktueller Funktionswert im Datensatz [Filtrat] gespeichert.
- Der aktuelle Funktionswert vom Datensatz [Daten] ist kleiner als der vorige Funktionswert vom Datensatz [Filtrat] abzüglich der Hysteresebreite. Der aktuelle Funktionswert vom Datensatz [Daten] wird als aktueller Funktionswert im Datensatz [Filtrat] gespeichert. Der Trend wechselt zum Abwärtstrend.

Befindet sich die Funktion im Abwärtstrend, wird je nach Größe des aktuellen Funktionswerts vom Datensatz [Daten] nach einer der drei folgenden Möglichkeiten vorgegangen:

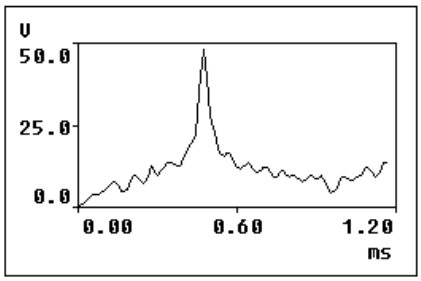
- Der aktuelle Funktionswert vom Datensatz [Daten] ist kleiner als der vorige Funktionswert vom Datensatz [Filtrat]. Der aktuelle Funktionswert vom Datensatz [Daten] wird als aktueller Funktionswert im Datensatz [Filtrat] gespeichert.
- Der aktuelle Funktionswert vom Datensatz [Daten] ist größer oder gleich dem vorigen Funktionswert vom Datensatz [Filtrat], aber nicht größer als der vorige Funktionswert vom Datensatz [Filtrat] zuzüglich der Hysteresebreite. Der vorige Funktionswert vom Datensatz [Filtrat] wird auch als aktueller Funktionswert im Datensatz [Filtrat] gespeichert.
- Der aktuelle Funktionswert vom Datensatz [Daten] ist größer als der vorige Funktionswert vom Datensatz [Filtrat] zuzüglich der Hysteresebreite. Der aktuelle Funktionswert vom Datensatz [Daten] wird als aktueller Funktionswert im Datensatz [Filtrat] gespeichert. Der Trend wechselt zum Aufwärtstrend.

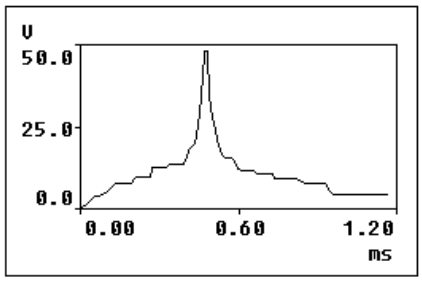
Eine Hysteresebreite von 0 liefert als Zieldatensatz [Filtrat] wieder den Quelldatensatz [Daten]. Im Gegensatz zu den Glättungsfunktionen Smo(), Smo3()... wird hier keine frequenz-abhängige, sondern eine rein amplituden-abhängige Störunterdrückung vorgenommen.

Beispiele:

```
NDhyst = Hyst(NDdata, 10)
```

Auf den zur linken Abbildung gehörigen Datensatz NDdata soll eine Hysterese mit der Hysteresebreite 10 angewendet werden. Der Datensatz NDdata besteht aus 118 Abtastwerten. Als Ergebnis der Anwendung der Hysterese wird in imc FAMOS der auf der rechten Abbildung dargestellte Datensatz NDhyst ausgegeben. Alle Zwischenschwingungen, die kleiner als die Hysteresebreite 10 sind, werden herausgefiltert:





Siehe auch:

Smo, Smo3, SlClip, STri, MInt

idB

Inverse Funktion zu dB

Deklaration:

```
idB ( EingangsDaten ) -> Transformiert
```

Parameter:

EingangsDaten	Daten, die aus dB auf das lineare Maß umgerechnet werten soll.
Transformiert	Resultierender Datensatz.

Beschreibung:

Diese Funktion ist die Umkehrfunktion zur Funktion dB. Eine dB-Berechnung wird also rückgängig gemacht. Bei komplexen Datensätzen, bei denen der Betrag in dB vorliegt, wird dieser wieder ins lineare Maß umgerechnet.

Dezibel (dB) bedeuten den zwanzigfachen Zehnerlogarithmus einer Zahl. Eine inverse dB-Berechnung ist also gleichbedeutend mit dem Term:

```
InverseDB = 10 ^ (number / 20)
```

- Bei der inversen dB-Berechnung wird stets die Einheit gelöscht.
- Komplexe Datensätze, die nicht vom Typ DP sind, können mit dieser Funktion nicht bearbeitet werden.
- Der Parameter darf strukturiert sein (Events/ Segmente).
- Bei normalen oder XY-Datensätzen sind die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Die Funktion idB kann keine negativen Werte erzeugen.

Beispiele:

```
linear = idB(dataInDB)
; Diese Formel ist äquivalent zu der Formel:
linear = 10 ^ (dataInDB / 20)
; nur die Einheit kann sich unterscheiden.
```

Siehe auch:

dB, log

IF

Leitet eine bedingte Verzweigung ein. Die nachfolgenden Anweisungen werden nur abgearbeitet, wenn der hier angegebene Ausdruck einen Wert > 0 liefert.
Alternativer Name: **WENN**

Deklaration:

```
IF Bedingung
```

Parameter:

Bedingung	Nur wenn die Bedingung erfüllt ist (Auswertung liefert einen Wert >0), werden die Anweisungen des folgenden Blocks durchlaufen.

Beschreibung:

Das Ende der zu diesem Block zugehörigen Anweisungen wird durch ein nachfolgendes ELSEIF, ELSE oder END markiert.

Als Bedingung kann z.B. eine Einzelwert-Variable oder ein komplexerer Ausdruck unter Verwendung von logischen Operatoren (AND,OR..) und/oder Vergleichsoperatoren (<, =, ...) angegeben werden.

Ein IF-Block kann beliebig viele ELSEIF-Verzweigungen enthalten. Eine eventuell ebenfalls angegebene ELSE-Verzweigung muss an letzter Stelle der Kette stehen.

Statt IF/ELSEIF/ELSE kann in vielen Anwendungsfällen auch bequemer die SWITCH/CASE/DEFAULT-Anweisung verwendet werden.

Beispiele:

Wenn das Maximum einer Variablen größer 0 ist, wird diese in einem Kurvenfenster angezeigt.

```
Maxi = Max(data)
IF Maxi
   SHOW data
END
```

Beim Laden einer Datei wird der Rückgabewert geprüft und ggf. eine Fehlermeldung ausgegeben.

```
fh = FileOpenDSF("Channel1.dat", 0)
IF fh = 0
   Pause ==> Fehler beim Laden der Datei <==
ELSE
 ; ...
   FileClose(fh)
END
```

Aus einem äquidistant abgetasteten Datensatz [data] soll ein Stück ausgeschnitten werden. Die beiden Grenzen [xmin] und [xmax]sind vorher vom Anwender eingegeben worden und werden nun auf Gültigkeit überprüft:

```
IF xmin < xoff?(data) OR xmin <= 0
   txError = "Illegal lower limit"
ELSEIF xmax > (xoff?(data)+(leng?(data)-1)*xdel?(data))
   txError = "Illegal upper limit"
ELSEIF xmin >= xmax
   txError = "Illegal limit relation"
ELSE
   result = Cut(data, xmin, xmax)
END
```

Siehe auch:

ELSEIF, ELSE, SWITCH

iFFT

inverse FFT, Transformation eines Spektrums in eine Zeitfunktion

Deklaration:

```
iFFT ( Spektrum ) -> ZeitDaten
```

Parameter:

Spektrum	Zu transformierendes Spektrum. Erlaubte Datentypen: [BP],[DP],[RI]
ZeitDaten	Ergebnis der inversen FFT.

Beschreibung:

Die Funktion iFFT() ist die inverse Fast Fourier-Transformation und kehrt die Funktion FFT um. Aus dem Spektrum wird die Zeitfunktion rekonstruiert.

Das Spektrum darf in jedem komplexen Datentyp vorliegen. Ansonsten wird das Spektrum in einer Form erwartet, wie es die Funktion FFT() mit dem Radix-2-Verfahren liefert. Es liegt nur eine Seite des Spektrums vor. Der erste komplexe Wert des Spektrums stellt den Gleichanteil dar, hat also den Imaginärteil Null. Es folgt eine Zweierpotenz von komplexen Spektrallinien, von denen die allerletzte reell ist. Intern wird diese eine Seite des Spektrums konjugiert komplex erweitert, so dass der inverse FFT-Algorithmus angewendet werden kann. Es wird ein Satz von Zeitdaten erzeugt, der stets eine Zweierpotenz lang ist.

- Hat der komplexe Datensatz 2^n + 1 Punkte, so hat der erzeugte Zeitdatensatz 2^(n+1) Punkte. Hat der komplexe Datensatz eine größere Länge, wird auf den nächstkleineren möglichen Wert abgeschnitten.
- Die maximal bearbeitbare Länge ist 67108865 (2^26+1) Punkte, das entspricht 134.217.728 (2^27) Zeitdaten. Verkürzen Sie ansonsten dann den Datensatz mit der Funktion Leng().
- Bei der iFFT wird keine Fensterfunktion berücksichtigt. Wenn Sie eine bei der FFT eingestellte Fensterfunktion nach der iFFT rückgängig machen möchten, erzeugen Sie einen Datensatz mit der Fensterfunktion (siehe Beispiel bei FFT) und dividieren den Zeitdatensatz durch die Fensterfunktion. Beachten Sie, dass sich meistens an den Rändern des Datensatzes größere Ungenauigkeiten ergeben.
- Als x-Einheit des Zeitdatensatzes wird stets "s" gewählt. Als y-Einheit wird die y-Einheit des Betrages oder Realteils benutzt.
- Die iFFT benötigt während ihrer Ausführung im Arbeitsspeicher temporären Speicherplatz. Ist (vor allem bei längeren Datensätzen) nicht ausreichend Speicherplatz vorhanden, wird eine Fehlermeldung erzeugt.
- Der x-Offset des übergebenen komplexen Datensatzes sollte Null sein. Ist er nicht Null, wird eine entsprechende Warnung erzeugt. Der x-Offset wird stets zu Null angenommenen. Sie können einen (positiven) x-Offset nur berücksichtigen, indem Sie das Spektrum mit der Funktion Cut() bis zur Frequenz Null nach unten erweitern.
- Die Funktion iFFT() kehrt die Funktion FFT() um, wenn ein Rechteckfenster benutzt wurde. Ein Spektrum, das mit der Funktion Spec() erzeugt wurde, kann ohne weiteres nicht umgekehrt werden.
- Eine FFT, die mit dem Mixed-Radix-Verfahren berechnet wurde, kann im Allgemeinen mit der iFFT() ebenfalls nicht rückgängig gemacht werden, da die Datensatzlänge des Ergebnisses beim hier angewendeten Verfahren immer eine 2er-Potenz ist.

Beispiele:

```
NDdata = iFFT(BPspektr)
```

BPspektr ist ein komplexer Datensatz mit 513 Punkten, es wird die Zeitfunktion mit 1024 Punkten erzeugt. Die iFFT einer Übertragungsfunktion liefert die Gewichtsfunktion eines Übertragungssystems.

```
NDdata = iFFT(Compl(BPspektr.M, 0))
```

Die Phase des Spektrums wird auf Null gesetzt, dann die zugehörige Zeitfunktion berechnet.

Siehe auch:

FFT, Spec

InDegr

Umrechnungsfaktor Bogenmaß in Grad = 57,297...
Alternativer Name: **InGrad**

Beschreibung:

Einheit: "Grad"/"Rad"

Beispiele:

Die Konstante PI als Maß für einen Winkel in Bogenmaß (Radiant) entspricht 180 Grad:

```
Degree180 = PI * INDEGR
```

Siehe auch:

InRad, PI, PI2

InRad

Umrechnungsfaktor Grad in Bogenmaß = 0.01745...

Beschreibung:

Einheit: "Rad"/"Grad"

Beispiele:

Ein Winkel von 180 Grad in Gradmaß entspricht in Bogenmaß (Radiant) der Kreiszahl PI (3.1415..):

```
_PI = 180'°' * INRAD
```

Siehe auch:

InDegr, PI, PI2

Int

Bildung des Integrals

Deklaration:

```
Int ( Daten ) -> Ergebnis
```

Parameter:

Daten	Zu integrierender Datensatz. Erlaubte Datentypen: [ND],[XY]
Ergebnis	Ergebnis der Integration.

Beschreibung:

Der übergebene Datensatz wird punktweise integriert. Es wird nach einem einfachen, effektiven Algorithmus integriert:

Bei normal reellen Datensätzen:

Die Summe aller Abtastwerte des Datensatzes wird bis zum aktuellen Punkt (aber nicht inklusive) gebildet und mit der Abtastzeit multipliziert. Damit wird die Integration umkehrbar zur Differentiation.

Bei XY-Datensätzen:

Für je zwei aufeinander folgende Punkte wird das Produkt aus dem Abstand ihrer x-Koordinaten und dem Mittelwert ihrer y-Werte gebildet und bis zum aktuellen Punkt aufsummiert.

Das Integral gibt die Fläche unter dem Datensatz vom Startpunkt bis zum aktuellen Punkt an. Beachten Sie, dass das Integrieren einer Kurve diese glättet.

- Entsprechend dem benutzten Algorithmus zur Integration ist der erste Wert des erzeugten Datensatzes stets Null. Damit wird der erzeugte Datensatz stets um einen Wert länger sein.
- Die Einheit des integrierten Datensatzes ist das Produkt aus der x- und y-Einheit des übergebenen Datensatzes.
- Im Gegensatz zum gleitenden Integral GInt wächst hier das Integrationsintervall von Null bis auf die Länge des Datensatzes.

Beispiele:

```
NDarea = Int(NDdata)
```

Einfache Berechnung der Fläche.

```
NDarea = Red(Int(IPol(NDdata, 3)), 3)
```

(Bei stärker gekrümmten Datensätzen oder Datensätzen mit größeren Sprüngen ist eine Spline- Interpolation vor dem Integrieren angebracht. Nach dem Integrieren kann die Datenmenge auf die angemessene Informationsmenge mit der Abtastfunktion Red reduziert werden

```
NDint = Int(NDdata)
definiteInt = Value(NDint, 10) - Value(NDint, 5)
```

Das bestimmte Integral eines Datensatzes NDdaten soll zwischen den Stützstellen x = 5 und x = 10 berechnet werden. Dazu wird zunächst die Integralfunktion NDint bestimmt. Die Differenz zwischen Stützstellen ergibt das bestimmte Integral. Beachten Sie, dass das bestimmte Integral nur dann als Fläche interpretierbar ist, wenn die Integralfunktion keine Nullstellen in diesem Intervall aufweist.

Siehe auch:

IntEx, Diff, MInt, Sum, MvSum

IntervalFrom2Levels

Verfügbar ab: Professional Edition

Intervalle werden aus einem Signal gebildet, wobei die Grenzen über Durchgänge des Signals durch angegebene Schwellen gegeben sind.

Deklaration:

```
IntervalFrom2Levels ( Eingangsdaten, LevelBegin, SlopeBegin, LevelEnd, SlopeEnd [, Runden] [, ResultFormat] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Eingangsdaten
LevelBegin	Das Signal muss durch diese Schwelle, die einen y-Wert bzw. Amplitude darstellt, hindurchgehen, um den Beginn eines Intervalls festzulegen.
SlopeBegin	Welche Flanke soll beim Beginn des Intervalls beachtet werden, die steigende (positive) oder fallende (negative)?
	1 : Positive Flanke (vom Punkt weg). Am Übergang gilt: y[k] <= Level < y[k+1]
	2 : Positive Flanke (zum Punkt hin). Am Übergang gilt: y[k] < Level <= y[k+1]
	3 : Negative Flanke (vom Punkt weg). Am Übergang gilt: y[k] >= Level > y[k+1]
	4 : Negative Flanke (zum Punkt hin). Am Übergang gilt: y[k] > Level >= y[k+1]
LevelEnd	Das Signal muss durch diese Schwelle, die einen y-Wert bzw. Amplitude darstellt, hindurchgehen, um das Ende eines Intervalls festzulegen.
SlopeEnd	Welche Flanke soll beim Ende des Intervalls beachtet werden, die steigende (positive) oder fallende (negative)?
	1 : Positive Flanke (vom Punkt weg). Am Übergang gilt: y[k] <= Level < y[k+1]
	2 : Positive Flanke (zum Punkt hin). Am Übergang gilt: y[k] < Level <= y[k+1]
	3 : Negative Flanke (vom Punkt weg). Am Übergang gilt: y[k] >= Level > y[k+1]
	4 : Negative Flanke (zum Punkt hin). Am Übergang gilt: y[k] > Level >= y[k+1]
Runden	Sollen die gefundenen Positionen gerundet werden? (optional , Standardwert: 0)
	0 : Automatisch: Präzisen Wert benutzen, aber bei digitalen Signalen aufgerundet.
	1 : Präzisen Wert benutzen, das Signal kann linear interpoliert zwischen den Stützstellen gedacht werden.
	2 : Die gefundene Position wird auf die Stützstelle des Signals gerundet, die am nächsten am interpolierten x-Wert liegt. Bei Signalen mit diskreten Werten.
	3 : Die gefundene Position wird auf die nächste Stützstelle des Signals abgerundet, die <= dem interpolierten x-Wert liegt.
	4 : Die gefundene Position wird auf die nächste Stützstelle des Signals aufgerundet, die >= dem interpolierten x-Wert ist. Bei digitalen Signalen.
ResultFormat	Format des Ergebnisses (optional , Standardwert: "")
	"" : automatisch: Erzeugt ein äquidistantes Ergebnis bei äquidistanten Eingangsdaten, ein XY-Ergebnis bei XY-Eingangsdaten mit monotoner Zeitspur, sonst ein Intervall-Code über Parameter.
	"equi" : äquidistant. Nur für äquidistante Eingangsdaten. Auch für Daten mit Segmenten
	"xy" : XY-Paare, Intervall-Code mit Zeitstempel. Nur für äquidistante Eingangsdaten oder XY-Eingangsdaten mit monotoner Zeitspur. Nicht für Daten mit Segmenten. Falls Daten in Treppenstufen vorliegen, ist zu beachten, dass bei späteren Zurückgewinnen eines Wertes aus dem Zeitstempel (falls am Stützpunkt liegend) durch Rundungsfehler zufällig das vorherige oder das Sample am Stützpunkt selbst benutzt wird.
	"para equi" : Äquidistanter Datensatz mit Intervall-Code aufgetragen über dem Parameter der zweikomponentigen Daten.
	"para xy" : XY-Datensatz mit Intervall-Code und Wert des Parameters der zweikomponentigen Daten. Nicht für Daten mit Segmenten
Ergebnis	Intervalldatensatz

Beschreibung:

Die Funktion kann auf Eingangsdaten mit Events oder Segmenten angewendet werden.

Bei XY-Daten, die mal monoton, mal nicht monoton ausfallen können, sollte automatisches Resultformat vermieden werden.

Sind die Eingangsdaten mit einem Rauschen beaufschlagt, kann die Funktion im Signal nicht beabsichtigte Durchgänge finden. Die Funktion Hyst() verhindert kleine Schwankungen und ist dann vorher aufzurufen. Ein Glätten löst das Problem der kleinen Schwankungen i. Allg. nicht umfassend. Dennoch ist ein Glätten sinnvoll. Aber anschließend werden die noch die verbleibenden störenden Schwankungen mit Hyst()

ausgeblendet.

Beide Flankenbedingungen können auch beim selben Messwert erfüllt sein. Das führt dann auf ein punktförmiges Intervall, dessen Position die erste Flanke ist.

Die Funktion sucht immer abwechselnd nach den Flankenbedingungen.

Intervallformat äquidistant

Eine Folge von Codes, die andeuten, an welchen Stellen Intervalle beginnen und enden.

0: außerhalb eines Intervalls

1: innerhalb eines Intervalls

2+Nachkomma: Ende eines Intervalls. Die Nachkommastellen bezeichnen die relative Position innerhalb des Abtastintervalls

3+Nachkomma: Beginn eines Intervalls. Die Nachkommastellen bezeichnen die relative Position innerhalb des Abtastintervalls

4+Nachkomma: Übergang zwischen 2 benachbarten Intervallen. Die Nachkommastellen bezeichnen die relative Position innerhalb des Abtastintervalls

5+Nachkomma: Punktförmiges Intervall. Die Nachkommastellen bezeichnen die relative Position innerhalb des Abtastintervalls

Intervallformat XY

Die x-Koordinate gibt die Position an, die y-Koordinate hat folgende Bedeutung:

2: Ende eines Intervalls

3: Beginn eines Intervalls

4: Übergang zwischen 2 benachbarten Intervallen

5: Punktförmiges Intervall

Beispiele:

Bestimmen aller Intervalle, in denen ein Signal auf über 2.0 ansteigt und am Ende auf unter 8.0 abfällt. Das Signal ist stark verrauscht.

```
input = 10*sin (Ramp(0,1e-3,400)*PI2*10+1.1) + Random ( 400, 2, 0, 0, 33 ) ; test data
input_smooth = Hyst( Smo(input,0.01), 0.1 )
ivl = IntervalFrom2Levels ( input_smooth, 2.0, 1, 8.0, 3, 0, "equi")
```

Bestimmen aller Intervalle, in denen ein Signal von 2.0 auf 8.0 ansteigt. Das Signal ist leicht verrauscht.

```
input = 10*sin (Ramp(0,1e-4,4000)*PI2*10+1.1) + 0.2 * Random ( 4000, 2, 0, 0, 0 ) ; test data
input_smooth = Smo(input, 0.001)
input_smooth = Hyst ( input_smooth, 0.05)
ivl = IntervalFrom2Levels ( input_smooth, 2.0, 1, 8.0, 1, 0, "equi")
```

Siehe auch:

IntervalFromLevel, IntervalGetStatist, Hyst

IntervalFromLevel

Verfügbar ab: Professional Edition

Unmittelbar aneinander grenzende Intervalle werden aus einem Signal gebildet, wobei die Grenzen über Durchgänge des Signals durch eine Schwelle bestimmt werden. Nur wenn Intervalle nicht den Nebenbedingungen an Amplitude und Breite entsprechen, werden sie aussortiert und es entstehen Lücken.

Deklaration:

```
IntervalFromLevel ( Eingangsdaten, Level, LowerLevel, UpperLevel, Slope, MinWidth, MaxWidth [, Runden] [, ResultFormat] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Eingangsdaten
Level	Das Signal muss durch diese Schwelle, die einen y-Wert bzw. Amplitude darstellt, hindurchgehen, um Beginn oder Ende eines Intervalls festzulegen.
LowerLevel	Ein Intervall wird nur dann als gültig erkannt, wenn innerhalb des Intervalls mindestens ein Wert genauso niedrig oder niedriger als dieser Wert ist. Wird auf Level gesetzt, falls nicht zu überprüfen.
UpperLevel	Ein Intervall wird nur dann als gültig erkannt, wenn innerhalb des Intervalls mindestens ein Wert genauso hoch oder höher als dieser Wert ist. Wird auf Level gesetzt, falls nicht zu überprüfen.
Slope	Welche Flanke soll beachtet werden, die steigende (positive) oder fallende (negative)?
	1 : Positive Flanke (vom Punkt weg). Am Übergang gilt: y[k] <= Level < y[k+1]
	2 : Positive Flanke (zum Punkt hin). Am Übergang gilt: y[k] < Level <= y[k+1]
	3 : Negative Flanke (vom Punkt weg). Am Übergang gilt: y[k] >= Level > y[k+1]
	4 : Negative Flanke (zum Punkt hin). Am Übergang gilt: y[k] > Level >= y[k+1]
	5 : Beliebige Flanke (vom Punkt weg). Am Übergang gilt: y[k] <= Level < y[k+1] oder y[k] >= Level > y[k+1]
	6 : Beliebige Flanke (zum Punkt hin). Am Übergang gilt: y[k] < Level <= y[k+1] oder y[k] > Level >= y[k+1]
MinWidth	Minimale Breite des Intervalls in x-Einheiten. Kürzere Intervalle werden aussortiert. 0, falls nicht zu überprüfen.
MaxWidth	Maximale Breite des Intervalls in x-Einheiten. Längere Intervalle werden aussortiert. 0, falls nicht zu überprüfen.
Runden	Sollen die gefundenen Positionen gerundet werden? (optional , Standardwert: 0)
	0 : Automatisch: Präzisen Wert benutzen, aber bei digitalen Signalen aufgerundet.
	1 : Präzisen Wert benutzen, das Signal kann linear interpoliert zwischen den Stützstellen gedacht werden.
	2 : Die gefundene Position wird auf die Stützstelle des Signals gerundet, die am nächsten am interpolierten x-Wert liegt. Bei Signalen mit diskreten Werten.
	3 : Die gefundene Position wird auf die nächste Stützstelle des Signals abgerundet, die <= dem interpolierten x-Wert liegt.
	4 : Die gefundene Position wird auf die nächste Stützstelle des Signals aufgerundet, die >= dem interpolierten x-Wert ist. Bei digitalen Signalen.
ResultFormat	Format des Ergebnisses (optional , Standardwert: "")
	"" : automatisch: Erzeugt ein äquidistantes Ergebnis bei äquidistanten Eingangsdaten, ein XY-Ergebnis bei XY-Eingangsdaten mit monotoner Zeitspur, sonst ein Intervall-Code über Parameter.
	"equi" : äquidistant. Nur für äquidistante Eingangsdaten. Auch für Daten mit Segmenten
	"xy" : XY-Paare, Intervall-Code mit Zeitstempel. Nur für äquidistante Eingangsdaten oder XY-Eingangsdaten mit monotoner Zeitspur. Nicht für Daten mit Segmenten. Falls Daten in Treppenstufen vorliegen, ist zu beachten, dass bei späteren Zurückgewinnen eines Wertes aus dem Zeitstempel (falls am Stützpunkt liegend) durch Rundungsfehler zufällig das vorherige oder das Sample am Stützpunkt selbst benutzt wird.
	"para equi" : Äquidistanter Datensatz mit Intervall-Code aufgetragen über dem Parameter der zweikomponentigen Daten.
	"para xy" : XY-Datensatz mit Intervall-Code und Wert des Parameters der zweikomponentigen Daten. Nicht für Daten mit Segmenten
Ergebnis	Intervalldatensatz

Beschreibung:

Die Funktion kann auf Eingangsdaten mit Events oder Segmenten angewendet werden.

Bei XY-Daten, die mal monoton, mal nicht monoton ausfallen können, sollte automatisches Resultformat vermieden werden.

Auch bei allgemeinen XY-Daten wird die Breite stets positiv ermittelt. Die Breite ist der Absolutbetrag aus der Differenz aus x am Ende und am Anfang eines Intervalls.

Bei verrauschtem Signal kann es sinnvoll sein, vorher zu glätten.

Verhalten

Die Funktion erkennt Anfang und Ende von wiederkehrenden Zyklen bzw. Übergänge zwischen den Zyklen.

Dabei wird davon ausgegangen, dass die Signale mit einem Rauschen beaufschlagt sind, dessen Höhe bekannt ist. Die Parameter UpperLevel und LowerLevel sollten einen Abstand vom Level haben, der deutlich größer als das Rauschen ist.

Bei eingestellter positiver Flanke wird erwartet, dass das Signal nach oben bis mindestens auf UpperLevel ansteigt, danach mindestens bis auf LowerLevel abfällt, bevor es wieder einen Durchgang mit eingestellter Flanke gibt. Entsprechend umgekehrt bei negativer Flanke.

Ein gültiges Intervall kommt nur zustande, wenn die genannten Amplitudenbedingungen erfüllt sind und die angegebenen Zeitbedingungen bezüglich der Intervallbreite.

Perioden des Signals, die die Bedingungen nicht erfüllen, führen zu Lücken zwischen den Intervallen.

Algorithmus

Zunächst wird ein erster Durchgang durch die gegebene Schwelle mit gegebener Richtung (Flanke) gesucht. Anschließend wird ein zweiter Durchgang gesucht.

Die Differenz zwischen beiden in x-Richtung wird auf minimale und maximale Breite geprüft. Falls außerhalb, wird der erste Durchgang verworfen.

Bei eingestellter positiver Flanke muss zunächst UpperLevel, erst danach LowerLevel erreicht werden, andernfalls wird der erste Durchgang verworfen. Entsprechend umgekehrt bei negativer Flanke.

Ist der erste Durchgang nicht verworfen, aber nicht beide LowerLevel und UpperLevel erreicht, wird der zweite Durchgang verworfen. Das verhindert das vorzeitige Ende an der falschen Flanke.

Gibt es beide Durchgänge, bilden sie ein Intervall.

Falls der zweite Durchgang nicht verworfen ist, wird er zum ersten erklärt, sonst bleibt der erste bestehen. Anschließend wird von vorn wiederholt.

Intervallformat äquidistant

Eine Folge von Codes, die andeuten, an welchen Stellen Intervalle beginnen und enden.

0: außerhalb eines Intervalls

1: innerhalb eines Intervalls

2+Nachkomma: Ende eines Intervalls. Die Nachkommastellen bezeichnen die relative Position innerhalb des Abtastintervalls

3+Nachkomma: Beginn eines Intervalls. Die Nachkommastellen bezeichnen die relative Position innerhalb des Abtastintervalls

4+Nachkomma: Übergang zwischen 2 benachbarten Intervallen. Die Nachkommastellen bezeichnen die relative Position innerhalb des Abtastintervalls

5+Nachkomma: Punktförmiges Intervall. Die Nachkommastellen bezeichnen die relative Position innerhalb des Abtastintervalls

Intervallformat XY

Die x-Koordinate gibt die Position an, die y-Koordinate hat folgende Bedeutung:

2: Ende eines Intervalls

3: Beginn eines Intervalls

4: Übergang zwischen 2 benachbarten Intervallen

5: Punktförmiges Intervall

Beispiele:

Periodische Intervalle bestimmen. Nulldurchgänge eines periodischen Signals, dessen Messwerte -2.0 und +2.0 übersteigen und dessen Periode zwischen 0.001 und 0.2 liegt.

```
ivl = IntervalFromLevel( input, 0.0, -2.0, 2.0, 1, 0.001, 0.2, 0, "equi")
```

Periodische Intervalle bestimmen. Das Eingangssignal ist verrauscht.

```
input = 10*sin (Ramp(0,1e-3,400)*PI2*10+1.1) + Random ( 400, 2, 0, 0, 35 )
input_smooth = Smo(input, 0.01)
ivl = IntervalFromLevel( input_smooth, 0.0, -3.0, 3.0, 1, 0.01, 0.3, 0, "equi")
```

Siehe auch:

IntervalFrom2Levels, IntervalGetStatist

IntervalGetStatist

Verfügbar ab: Professional Edition

Für jedes Intervall eines Datensatzes werden statistische Kennwerte wie Min, Max etc. berechnet.

Deklaration:

```
IntervalGetStatist ( Eingangsdaten, Intervalldaten, Intervallformat, Berechnung [, Interpolation] [, ResultFormat] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Eingangsdaten. Äquidistant oder XY mit X monoton.
Intervalldaten	Intervalldaten
Intervallformat	Bedeutung der Intervalldaten: Sind die Intervalle X-Positionen oder als Parameter zu deuten? Letzteres ist wichtig, wenn die Eingangsdaten vom Typ XY sind.
	"" : automatisch: Bei äquidistanten Eingangsdaten und bei XY-Eingangsdaten mit monotoner X-Spur bedeuten die Intervalldaten X-Positionen. Sonst Deutung als Parameter. Nicht empfohlen, wenn die Eingangsdaten XY sind, die X-Spur aber mal monoton, mal nicht monoton sein kann.
	"equi" : äquidistant. Nur für äquidistante Eingangsdaten.
	"xy" : XY-Paare, Intervall-Code mit Zeitstempel. Nur für äquidistante Eingangsdaten oder XY-Eingangsdaten mit monotoner Zeitspur.
	"para equi" : Äquidistanter Datensatz mit Intervall-Code aufgetragen über dem Parameter der zweikomponentigen Daten.
	"para xy" : XY-Datensatz mit Intervall-Code und Wert des Parameters der zweikomponentigen Daten.
Berechnung	Berechnung
	"len" : Anzahl von Samples, die das Intervall breit ist.
	"width" : Breite des Intervalls in x-Einheiten, Differenz aus x am Ende und x am Anfang.
	"1/width" : Kehrwert des Betrags der Breite des Intervalls. Stellt das Intervall eine Periode dar, ist das die Frequenz.
	"min" : Minimum
	"max" : Maximum
	"mean" : Mittelwert. Bei linearer Interpolation der wirkliche Mittelwert des Polygonzugs.
	"sum" : Summe der y-Werte. Am Intervallrand anteilig addiert. Signal stets als Treppen gedeutet.
	"int" : Gesamtes Integral von y über x.
	"rms" : Effektivwert, basierend auf quadratischer Summe von y-Werten. Gewichtung entsprechend Abstand zwischen den Samples. Randwerte werden anteilig gezählt. Signal stets als Treppen gedeutet.
	"stdev" : Standardabweichung. Alle Samples werden gleich gewichtet (so als ob äquidistant). Randwerte, die kürzer als der Sample-Abstand sind, werden nicht gezählt. Signal stets als Treppen gedeutet.
	"minposF" : Position des Minimums. Falls derselbe Wert mehrfach auftritt, wird das erste Auftreten bestimmt.
	"maxposF" : Position des Maximums. Falls derselbe Wert mehrfach auftritt, wird das erste Auftreten bestimmt.
	"minposL" : Position des Minimums. Falls derselbe Wert mehrfach auftritt, wird das letzte Auftreten bestimmt.
	"maxposL" : Position des Maximums. Falls derselbe Wert mehrfach auftritt, wird das letzte Auftreten bestimmt.
	"x begin" : x-Wert bei Beginn des Intervalls
	"x end" : x-Wert bei Ende des Intervalls
	"y begin" : y-Wert bei Beginn des Intervalls
	"y end" : y-Wert bei Ende des Intervalls
Interpolation	Interpolation (optional , Standardwert: 0)
	0 : Automatisch: Linear interpoliert, aber bei digitalen Signalen wie Treppenstufen.
	1 : Das Signal wird linear interpoliert zwischen den Stützstellen gedacht.
	2 : Das Signal wird konstant zwischen den Stützstellen fortgesetzt gedacht. Jeder Wert wird als Treppenstufe bis zur nächsten Stützstelle gehalten.
ResultFormat	Format des Datensatzes mit Ergebnissen. (optional , Standardwert: "list")

	"list" : Liste: Erzeugt eine einfache Liste der Ergebniswerte, ein Ergebniswert pro Intervall. Ohne Zeitstempel und ohne jede Zeitinformation
	"equi" : äquidistant. Das Ergebnis hat dieselbe Länge wie die Eingangsdaten. Auch für Daten mit Segmenten. Der Ergebniswert pro Intervall wird während des gesamten Intervalls (also wiederholt) ausgegeben. Bei Intervallformat "para" ist äquidistant über dem Parameter der zweikomponentigen Daten aufgetragen.
	"xy" : XY-Paare, Ergebniswert mit Zeitstempel. Nicht für Daten mit Segmenten. Bei Intervallformat "para" Paare aus Ergebniswert und Parameter der zweikomponentigen Daten.
Ergebnis	Ergebnis

Beschreibung:

Die Funktion kann auf Eingangsdaten mit Events oder Segmenten angewendet werden.

Eingangsdaten und Intervalldaten müssen nicht dieselbe Abtastzeit haben. Vielmehr werden die Abtastzeit und der x-Offset sogar beachtet. Die Intervalldaten legen Zeiten fest, die auf die Eingangsdaten angewendet werden. Eine unterschiedliche absolute Zeit wird ignoriert.

Für unvollständige Intervalle und für Intervalle, die (teilweise) außerhalb der Eingangsdaten liegen, wird kein Ergebniswert gebildet.

Bei äquidistanten Resultaten bzw. Intervallen kann nur 1 Beginn oder Ende eines Intervalls pro Abtastschritt angegeben werden.

Die Berechnungen gehen vom Intervall-Beginn bis zum Ende. Wenn Beginn und Ende nicht gerade genau an den Positionen von Samples sind, werden anteilige Samples berücksichtigt.

Enthält das Ergebnis eine Zeitspur, wird der Intervallbeginn angegeben.

Wenn das Signal treppenförmig interpoliert wird und Intervallgrenzen an den Messpunkten des Signals liegen, kommt es durch kleinste Rechenungenauigkeiten zu einem undefinierten Verhalten: Der Messwert liegt mal links, mal auch rechts von der Intervallgrenze. Folglich gehört der Messwert mal zum Intervall, mal nicht. Das kann sich z.B. bei einer Min/Max-Suche stark auswirken. Abhilfe ist möglich, indem die Intervallgrenzen vorher deutlich weg von den Messpunkten gelegt werden, z.B. in die Mitte zwischen den Messpunkten.

Beispiele:

Effektivwerte von einzelnen Schwingungen eines Signals bestimmen. Das Eingangssignal ist verrauscht.

```
input = 10*sin (Ramp(0,1e-3,400)*PI2*10+1.1) + Random ( 400, 2, 0, 0, 35 ) ; test data
input_smooth = Smo(input, 0.01)
ivl = IntervalFromLevel( input_smooth, 0.0, -3.0, 3.0, 1, 0.01, 0.3, 0, "equi")
st = IntervalGetStatist ( input, ivl, "", "rms", 1, "list")
```

Siehe auch:

IntervalFrom2Levels, IntervalFromLevel

IntEx

Verfügbar ab: Professional Edition

Integration, Bildung des Integrals

Deklaration:

```
IntEx ( Datensatz [, Berechnung] [, Format] [, Resetoption] [, Resetkanal] ) -> Integral
```

Parameter:

Datensatz	Zu integrierender Datensatz
Berechnung	Wie wird das Integral berechnet, nach welcher Regel. (optional , Standardwert: "rect")
	"rect" : Rechteck-Regel. Jeder Messwert wird als Rechteck mit der Breite der Abtastzeit aufgefasst.
	"trapez" : Trapez-Regel. Die Messwerte werden linear interpoliert gedacht. Damit entsteht bei N Messwerten eine Folge von N-1 Trapezen mit der Breite der Abtastzeit. Das Ergebnis ist um 1 Wert kürzer als bei der Rechteckregel.
	"trapez+" : Trapez-Regel. Die Messwerte werden linear interpoliert gedacht. Damit entsteht bei N Messwerten eine Folge von N-1 Trapezen mit der Breite der Abtastzeit. Zusätzlich wird der allerletzte Messwert als ein Rechteck mit der Breite der Abtastzeit gedeutet. Das Ergebnis ist genauso lang wie bei der Rechteckregel.
Format	Format des Ergebnisses. (optional , Standardwert: "zero")
	"zero" : Vorn mit null. Der erste Wert des Integrals ist eine 0. Damit wird ausgedrückt, dass bis genau zum Beginn des ersten Messwertes noch keine Fläche vorhanden ist. Der Verlauf des Integrals ergibt sich folgerichtig verzögert. Das Ergebnis ist um einen Wert länger.
	"nozero" : Vorn ohne null. Die 0, die sich eigentlich beim Integrieren als erster Wert des Ergebnisses ergibt, wird weggelassen. Damit erscheint das Ergebnis nicht mehr verzögert. Das Ergebnis ist also verfälscht, was beim Interpretieren zu beachten ist.
	"total" : Gesamtintegral. Nur der letzte Wert des Integrals wird zurückgegeben. Das Ergebnis hat die Länge 1.
Resetoption	Soll ein Reset ausgeführt werden? (optional , Standardwert: "no")
	"no" : Niemals Reset. Der Parameter Resetkanal wird nicht angegeben oder ist 0.
	"reset" : Harter Reset. Das Integral wird an ausgewählten Stellen unmittelbar auf 0 gesetzt. Der Resetkanal muss angegeben werden.
	"soft" : Weicher Reset. Das Integral bewegt sich an ausgewählten Stellen sanft gegen 0. Der Resetkanal muss angegeben werden.
Resetkanal	Resetkanal. Wird je nach "Resetoption" angegeben und benutzt. (optional)
Integral	Integral

Beschreibung:

Der übergebene Datensatz wird punktweise integriert.

Der Datensatz darf Events und Segmente haben, er muss äquidistant sein.

Die Einheit des integrierten Datensatzes ist das Produkt aus der x- und y-Einheit des übergebenen Datensatzes.

Bei XY-Daten wird Int() angewendet.

Resetkanal

Falls Wert = 0: Fortsetzen der Integration. Falls Wert <> 0: Reset des Integrals auf 0.

Der Ergebnisdatensatz wird genau an den Stellen, an denen der Resetkanal <> 0 ist, auch auf 0 gesetzt. Je nachdem, ob das Ergebnis vorn eine 0 hat oder nicht, ergibt sich damit ein unterschiedlicher Verlauf.

Der Resetkanal muss hinsichtlich Segmenten und Events dieselbe Struktur aufweisen wie der zu integrierende Datensatz.

Der Resetkanal hat dieselbe Länge wie der berechnete Verlauf des Integrals. Wird z.B. bei Rechteckregel vorn eine 0 ergänzt, ist der berechnete Verlauf ein Messwert länger als der übergebene Datensatz. Mit dem Resetkanal soll an jeder Stelle des berechneten Verlaufs ein Reset auf 0 erzielt werden können. Also ist der Resetkanal dann auch einen Wert länger.

Ist der Resetkanal nicht lang genug, wird er mit Nullen hinten aufgefüllt gedacht.

Falls das Gesamtintegral gemeinsam mit einem Resetkanal benutzt wird, erfolgt das interne Aufintegrieren ohne vorn angefügte 0.

Harter, weicher Reset

Beim harten Reset wird das Integral, das eine beliebige Höhe haben kann, auf 0 gesetzt. Im Ergebnis drückt sich das als ein Sprung auf 0 aus.

Beim weichen Reset wird eine Gerade ermittelt. Diese Gerade verbindet das Integral vom vorauslaufenden Reset ausgehend mit dem aktuell ermittelten Wert des Integrals. Diese Gerade wird vom Integral abgezogen. Damit ergibt sich typisch ein sanfter Verlauf Richtung 0. Gibt es keinen vorausgehenden Reset, wird der Signalbeginn benutzt.

Sei beispielsweise die Wertefolge nach dem letzten Reset: Eingangsdaten=[7,3,17], Resetkanal=[0,0,1]. Es ergibt sich die Summe von Eingangswerten zu 7+3+17=27. Bei den vorliegenden 3 Werten ein Mittelwert von 27/3=9. Modifizierte Eingangsfolge abzüglich des Mittelwertes

ist [7-9,3-9,17-9] = [-2,-6,8]. Aufsummieren führt auf [-2,-8,0]. Der letzte Wert ist 0, passend zum letzten Wert des Resetkanals.

Das Integral hinter dem letzten Reset bleibt beim Reset unverändert.

Der weiche Reset wirkt, als ob der Mittelwert des Signals hinter dem letzten Reset bis einschließlich dem aktuellen vor der Integration subtrahiert worden wäre.

Alle Methoden des Resets bekämpfen störende Offsets, die ein Wegdriften des Integrals verursachen.

Beispiele:

Integration der Beschleunigung a, um die Geschwindigkeit v zu erhalten.

```
v = IntEx ( a, "trapez")
```

Zahlenbeispiele für die Wirkung der unterschiedlichen Berechnungen und Formate

```
A=[3,5,9]
B=IntEx(A)
;B=[0,3,8,17]

B=IntEx(A,"rect", "zero")
;B=[0,3,8,17]

B=IntEx(A,"rect", "nozero")
;B=[3,8,17]

B=IntEx(A,"rect", "total")
;B=17

B=IntEx(A,"trapez", "zero")
;B=[0,4,11]

B=IntEx(A,"trapez", "nozero")
;B=[4,11]

B=IntEx(A,"trapez", "total")
;B=11

B=IntEx(A,"trapez+", "zero")
;B=[0,4,11,20]

B=IntEx(A,"trapez+", "nozero")
;B=[4,11,20]

B=IntEx(A,"trapez+", "total")
;B=20
```

Zahlenbeispiele für die Wirkung des Resets

```
A=    [ 7, 3,17, 9,11, 7, 3, 9]
Reset=[ 0, 0, 0, 1, 0, 0, 1, 0]
B=IntEx(A,"rect", "zero", "reset", Reset )
;B=   [ 0, 7,10, 0, 9,20, 0, 3,12]

B=IntEx(A,"rect", "nozero", "reset", Reset )
;B=   [ 7,10,27, 0,11,18, 0, 9]

B=IntEx(A,"rect", "zero", "soft", Reset )
;B=   [ 0,-2,-8, 0, 0, 2, 0, 3,12]

B=IntEx(A,"rect", "nozero", "soft", Reset )
;B=   [-2,-8, 0, 0, 4, 4, 0, 9]
```

Siehe auch:

Int, GInt

IPol

Interpolation mit kubischen Splines

Deklaration:

```
IPol ( Daten, EwFaktor ) -> Interpoliert
```

Parameter:

Daten	Datensatz, der mit kubischen Splines interpoliert werden soll. [ND],[XY]
EwFaktor	Faktor, um den der Datensatz zu vergrößern ist.
Interpoliert	Spline-interpolierter Datensatz

Beschreibung:

Der übergebene Datensatz wird mit einem kubischen Spline interpoliert. Dabei gibt der 2. Parameter den Faktor an, um den der Datensatz zu vergrößern ist.

Bei der Spline-Interpolation wird durch die Punkte des Datensatzes ein natürlicher kubischer Spline gelegt. Eine entsprechend hohe Abtastung ergibt den zu erzeugenden Datensatz mit höherer Punktedichte. Eine Interpolation mit kubischen Splines verleiht einem Datensatz ein sehr glattes, rundes Erscheinungsbild ohne Kanten. Kubische Splines sind gut geeignet, um Kurven mit geringer Punktedichte für eine gut aussehende grafische Darstellung aufzubereiten.

Es ist zu beachten, dass kubische Splines leicht überschwingen.

Es werden bei der Interpolation nur Zwischenpunkte berechnet, es wird nicht extrapoliert.

- Bei XY-Daten muss die X-Spur streng monoton wachsend sein.
- Die Einheiten bleiben unverändert.
- Die Abtastzeit des erzeugten Datensatzes ist um den übergebenen Faktor kleiner.
- Der übergebene Faktor darf nur eine ganze Zahl > 1 und < 8192 sein.
- Da nicht extrapoliert wird, wird der Datensatz in seiner Länge nicht ganz um den übergebenen Faktor vergrößert. Es fehlen (Faktor -1) Werte.
- Vor allem bei rechteckförmigen Sprüngen ist das Überschwingen von kubischen Splines störend. Zur Abhilfe können Sie das Signal vorher glätten oder in Abschnitte zerlegen, die Sie gegebenenfalls linear interpolieren.
- Wenn Sie hochfrequent verrauschte Signale interpolieren, erhalten Sie im Allgemeinen schlechte Ergebnisse. Es ist dann zweckmäßig, z. B. durch Glätten das Rauschen vor der Interpolation zu unterdrücken.
- Bei äquidistanten Daten macht die Funktion Red() eine Interpolation rückgängig.

Beispiele:

Die Punktedichte eines Datensatzes wird um den Faktor 10 erhöht, womit der wahrscheinliche Verlauf des Signals wesentlich besser nachgebildet wird. Eine grafische Darstellung sieht besser aus, nachfolgende mathematische Operationen, wie Differentiation, werden zuverlässiger.

```
data2 = IPol(data, 10)
```

Siehe auch:

Lip, MatrixIPol, Red, Leng

IsCplx

Die Funktion prüft, ob es sich beim Parameter um einen komplexen Datensatz handelt.

Deklaration:

```
IsCplx ( Daten ) -> EwErgebnis
```

Parameter:

Daten	Zu prüfender Datensatz
EwErgebnis	Ergebnis
	0 : Kein komplexer Datensatz.
	1 : Komplexer Datensatz in Real-/Imaginärteil-Darstellung [RI].
	2 : Komplexer Datensatz in Betrag-/Phase-Darstellung [BP].
	3 : Komplexer Datensatz in Betrag(Dezibel)-/Phase-Darstellung [DP].

Beschreibung:

Die Funktion prüft, ob der übergebene Datensatz komplex ist. Falls dies der Fall ist, wird die Form, in der der Datensatz vorliegt, ermittelt.

Beispiele:

Eine Datei wird geladen und geprüft, ob der erste enthaltene Datensatz komplex ist:

```
idFile = FileOpenDSF("c:\dat\test.dat", 0)
IF idFile >= 1
   test = FileObjRead(idFile, 1)
   IF IsCplx(test) = 0
      BoxMessage("Achtung", "Datensatz ist komplex", "!1")
   END
   ret = FileClose(idFile)
END
```

Siehe auch:

Cmp1, Cmp2, Compl

IsXY

Die Funktion prüft, ob es sich um einen XY-Datensatz handelt.

Deklaration:

```
IsXY ( Daten ) -> EwErgebnis
```

Parameter:

Daten	Zu prüfender Datensatz
EwErgebnis	Ergebnis
	0 : Kein XY-Datensatz
	1 : XY-Datensatz

Beispiele:

Eine Datei wird geladen und geprüft, ob der erste enthaltene Datensatz vom Typ XY ist. Falls ja, wird er vor der Weiterverabeitung in einen äquidistant abgetasteten Datensatz gewandelt:

```
idFile = FileOpenDSF("c:\dat\test.dat", 0)
IF idFile >= 1
   test = FileObjRead(idFile, 1)
   IF IsXY(test) = 0
      test = XYdt(test, 0.1)
   END
   ...
   ret = FileClose(idFile)
END
```

Siehe auch:

CmpX, CmpY, XYof

Join

Bindet 2 Datensätze aneinander
Alternativer Name: **Binde**

Die Funktion ist veraltet, statt dessen sollte die leistungsfähigere Funktion JoinEx() verwendet werden.

Deklaration:

```
Join ( DatenVorn, DatenHinten ) -> DatenVerbunden
```

Parameter:

DatenVorn	Erster Datensatz. Erlaubte Typen: [ND].
DatenHinten	Zweiter Datensatz. Erlaubte Typen: [ND].
DatenVerbunden	Ergebnis der Aneinanderreihung der beiden Datensätze.

Beschreibung:

Ein typischer Einsatz dieser Funktion ist das Bilden eines Datensatzes aus vielen Einzelwerten. Um dieses Aneinanderhängen auch in einer Schleife sinnvoll starten zu können, ist ein leerer Datensatz erforderlich, an den dann nach und nach Werte angehängt werden können. Der leere Datensatzes kann z. B. mit der symbolischen Konstante EMPTY angegeben werden.

Die Konstante EMPTY ist besonders sinnvoll bei Sequenzen zu benutzen, die die Funktion Join() in einer Schleife aufrufen.

- Wenn zwei Einzelwerte zu einem normalen Datensatz zusammengebunden werden, werden folgende Standardwerte für den Datensatz gewählt: Abtastintervall = 1, x-Offset = 0.
- Wenn ein Einzelwert und ein normaler Datensatz zusammengebunden werden, werden die Einheiten des normalen Datensatzes übernommen. Sinnvollerweise sollte die Einheit des Einzelwertes mit der y-Einheit des normalen Datensatzes übereinstimmen.
- Wenn zwei normale Datensätze aneinander gebunden werden, werden Einheiten, Abtastzeit und x-Offset des ersten Datensatzes übernommen. Für eine sinnvolle Anwendung der Funktion sollten die Einheiten und Abtastzeit übereinstimmen.
- Für ein zeit- bzw. x-richtiges Verbinden bzw. Verschmelzen von Datensätzen können Sie die Funktion Append() verwenden.

Beginnend mit Version 7.6 steht die leistungsfähigere Funktion JoinEx() zur Verfügung. Die Funktionalität der Join()-Funktion lässt sich außerdem alternativ mit Initialisierungslisten nachbilden. Die folgenden 3 Aufrufe sind nahezu äquivalent:

```
concat = Join(front, back) ;(1)
concat = JoinEx(front, back) ;(2)
concat = [front, back] ;(3)
```

Der einzige Unterschied ist, dass bei (1) das Ergebnis immer in ein reelles Datenformat konvertiert wird, während bei (2) und (3) das Zahlenformat der Parameter nach Möglichkeit erhalten wird.

Beispiele:

Es ist ein Datensatz zu erstellen, der eine Liste von mehreren Maxima enthält:

```
NEliste = EMPTY   ;Ein leerer Datensatz wird erzeugt.
NEliste = Join(NEliste, EWmax1)
; An die leere Liste wird eine Zahl (erstes Maximum) gehängt, NEliste wird hier zu einem gewöhnlichen Einzelwert.
NEliste = Join(NEliste, EWmax2)
; An die bestehende Liste wird ein weiterer Wert angehängt (zweites Maximum), NEliste wird jetzt
ein normaler Datensatz mit der Länge 2) usw.
```

Die aufgenommenen Daten vor dem Trigger werden zusammengefügt mit den Daten nach dem Trigger:

```
NDcomplete = Join(NDpreTrigger, NDpostTrigger)
```

Siehe auch:

JoinEx, Append, AppendLoop, Cut

JoinEx

Bindet Datensätze aneinander

Deklaration:

```
JoinEx ( DatenVorn, DatenHinten [, DH2] [, DH3] [, DH4] [, DH5] [, DH6] [, DH7] [, DH8] [, DH9] [, DH10] [, DH11] [, DH12] [, DH13] [, DH14] ) -> DatenVerbunden
```

Parameter:

DatenVorn	Erster Datensatz. Erlaubte Typen: [ND].
DatenHinten	Anzuhängender Datensatz. Erlaubte Typen: [ND].
DH2	2. anzuhängender Datensatz (optional). (optional)
DH3	3. anzuhängender Datensatz (optional). (optional)
DH4	4. anzuhängender Datensatz (optional). (optional)
DH5	5. anzuhängender Datensatz (optional). (optional)
DH6	6. anzuhängender Datensatz (optional). (optional)
DH7	7. anzuhängender Datensatz (optional). (optional)
DH8	8. anzuhängender Datensatz (optional). (optional)
DH9	9. anzuhängender Datensatz (optional). (optional)
DH10	10. anzuhängender Datensatz (optional). (optional)
DH11	11. anzuhängender Datensatz (optional). (optional)
DH12	12. anzuhängender Datensatz (optional). (optional)
DH13	13. anzuhängender Datensatz (optional). (optional)
DH14	14. anzuhängender Datensatz (optional). (optional)
DatenVerbunden	Ergebnis der Aneinanderreihung aller Parameter.

Beschreibung:

Als Parameter können Einzelwerte oder einfache Datensätze (äquidistant abgetastet, keine Events, keine Segmente) verwendet werden.

- Einheiten und weitere Kennwerte (wie Pretrigger/x-Offset, Abtastzeit/x-Delta) "erbt" der neue Datensatz vom ersten Parameter, der eine Länge >= 2 hat. Für eine sinnvolle Anwendung der Funktion sollten die Einheiten und das Abtastintervall aller Parameter übereinstimmen.
- Wenn nur Einzelwerte angegeben sind, werden die Einheiten von [DatenVorn] verwendet. Das Ergebnis hat dann ein Abtastintervall von 1.0 und einen x-Offset von 0.0.
- Wenn alle Parameter dasselbe Datenformat aufweisen, wird dieses auch für das Ergebnis verwendet. Ansonsten wird in das Zahlenformat "Reell 8 Byte" (double) konvertiert.
- Für ein zeit- bzw. x-richtiges Verbinden bzw. Verschmelzen von Datensätzen können Sie die Funktion Append() verwenden.
- Wenn Sie wiederholt immer wieder Daten an einen Datensatz anhängen wollen (z.B. in einer Schleife), sollten Sie die für diesen Anwendungsfall optimierte Funktion AppendLoop() verwenden.

Statt der Funktion JoinEx() können Sie in den meisten Fällen auch eine **Initialisierungsliste** (komma-getrennte Initialisierer in eckigen Klammern) verwenden. Die beiden folgenden Zeilen sind z.B. äquivalent:

```
Mean3Days = Mean(JoinEx(BeforeYesterday, Yesterday, Today))
Mean3Days = Mean([BeforeYesterday, Yesterday, Today])
```

Beispiele:

Ein Datensatz wird um eine führende und eine schließende 0 ergänzt:

```
Data = JoinEx(0, Data, 0)
```

Die Messdaten von 3 aufeinanderfolgenden Tagen werden zu einem Datensatz zusammengefügt:

```
Data3Days = JoinEx(BeforeYesterday, Yesterday, Today)
```

In einem eventierten Datensatz werden die Maxima aller Events bestimmt und diese in einen neuen Datensatz kopiert.

```
allMax = EMPTY
FOREACH EVENT ev IN data
   allMax = JoinEx(allMax, max(ev))
```

```
END
```

Hinweis: Für solche Anwendungen ist die Funktion AppendLoop() oft besser geeignet, da diese bei wiederholtem, häufigem Anhängen wesentlich schneller ist.

Siehe auch:

Join, Append, AppendLoop, Cut

KBT

KB-Bewertung und Takt-Maximal-Werte

Deklaration:

```
KBT ( Daten, EwFrequenzUnten, EwFrequenzOben, EwIntegrationen, EwZeitbewertung, EwTakt, EwNormierung, EwMinimum, Null ) -> KBDaten
```

Parameter:

Daten	Zu analysierender Datensatz.
EwFrequenzUnten	Die untere Grenzfrequenz des Bandpasses (>0!), i. a. auf 1 setzen für 1Hz. Intern wird automatisch eine um 0.8 kleinere Frequenz für den Bandpass gewählt, also entsprechend dann 0.8Hz. Sind die obere und untere Grenz-Frequenz kleiner als Null angegeben, werden keine Filterungen (Bandpass und Hochpass) und Integrationen durchgeführt
EwFrequenzOben	Die obere Grenzfrequenz des Bandpasses (>0!), i. a. auf 80 zu setzen für 80Hz. Intern wird automatisch eine um 1/0.8 vergrößerte Frequenz benutzt, also entsprechend 100Hz.
EwIntegrationen	Anzahl der auszuführenden Integrationen. Haben Sie z. B. ein Beschleunigungs-Signal, muss es einmal integriert werden, um ein Geschwindigkeits-Signal zu erhalten. Die Integrationen bzw. Ableitungen werden nur durchgeführt, wenn auch eine Bandpass-Filterung durchgeführt wird
	0 : Keine Integration (Standard)
	1 : Eine Integration
	2 : Zweifache Integration
	-1 : Erste Ableitung
	-2 : Zweite Ableitung
EwZeitbewertung	Mittelungsdauer für die Bildung des gleitenden Effektivwertes (Zeitbewertung).
	>0 : Freie Mittelungsdauer
	0 : Keine Effektivwert-Bildung
	-1 : FAST (0.125s)
EwTakt	Der Takt, in dem Takt-Maximal-Werte berechnet werden.
	0 : Keine Bildung von Takt-Maximal-Werten
	>0 : Ein von Ihnen angegebener Takt, z. B. 30 für 30 Sekunden. Ist der Takt kleiner als die Abtastzeit, wird die Abtastzeit als Takt benutzt. Ist der Takt länger als die Dauer des Quell-Datensatzes, wird der komplette Quell-Datensatz benutzt. Wenn der Datensatz mehr als einen Takt lang ist und der letzte Takt nicht mehr vollständig im Datensatz enthalten ist, werden nur die verfügbaren Messpunkte benutzt. Es liegt dann an Ihnen, den letzten Takt-Maximal-Wert noch zu berücksichtigen.
EwNormierung	Mit diesem Faktor wird das Ergebnis zur Normierung multipliziert. (0 bedeutet keine Normierung, die Multiplikation wird eingespart.)
EwMinimum	Ergebnisse unterhalb von diesem Wert werden auf Null gesetzt. Dieses Wegschneiden von kleinen Werten wird nur durchgeführt, wenn auch eine gleitende Effektivwert-Bildung durchgeführt wird. Sinnvollerweise sollte kein negative Normierung benutzt werden! Da nach der Effektivwert-Bildung keine negativen Zahlen auftreten können, wird [Minimum]=0 gesetzt, wenn nichts weggeschnitten werden soll.
Null	Reservierter Parameter, auf 0 zu setzen.
KBDaten	Ergebnis der Bewertung

Beschreibung:

Die KB-Bewertung dient zur Beurteilung von Erschütterungen vor allem im Frequenz-Bereich von 1Hz bis 80Hz. Diese Schwingungen werden von Menschen wahrgenommen und als unangenehm empfunden. Vor allem im Immissions-Schutz wird die KB-Bewertung zu einer normgerechten Auswertung benutzt.

Die Funktion ist dafür ausgelegt, eine komplette KB-Bewertung inklusive Bestimmung der Takt-Maximal-Werte durchzuführen.

Die KB-Bewertung wird entsprechend DIN durchgeführt. Dabei werden folgende Teilschritte nacheinander ausgeführt:

- Es wird davon ausgegangen, dass ein unbewertetes Messsignal (Schnelle-Signal, Geschwindigkeit über der Zeit) vorliegt.
- Dieses Signal wird bandbegrenzt, typischerweise zwischen 1Hz und 80Hz oder aber alternativ auch im Nahbereich von Sprengungen von 1Hz bis 315Hz oder bei Schienen-Verkehrswegen 4Hz bis 80Hz. DIN45669 Teil1 schreibt dafür einen Butterworth-Bandpass 4. Ordnung vor, wobei die untere Grenz- Frequenz noch mit dem Faktor 0.8 verkleinert wird, die obere mit dem Faktor 1/0.8 vergrößert wird, so dass die tatsächlich benutzten Grenz-Frequenzen bei 0.8Hz und 100Hz liegen.
- Das bandbegrenzte Signal wird mit einem Hochpass 1.Ordnung und der Grenz-Frequenz 5.6Hz gefiltert. Man erhält damit das frequenz-

bewertete Erschütterungs-Signal.

- Das Signal kann nun noch integriert oder differenziert werden. Das ist nach der bereits durchgeführten Filterung eine besonders günstige Stelle, da hochfrequentes Rauschen bereits entfernt ist (also Ableitung gut bestimmbar) und auch niederfrequente Driften und Offsets eliminiert sind (also Integration gut durchführbar). Das ist nicht in der DIN beschrieben, aber von hoher praktischer Bedeutung. Integrationen und Differentiationen werden durchgeführt, indem das digitale Filter für den Bandpass um Pol- bzw. Nullstellen bei der Frequenz Null erweitert wird.
- Vom frequenz-bewerteten Signal wird die bewertete Schwingstärke durch Bildung des gleitenden Effektivwertes gebildet. Der gleitende Effektivwert wird auf folgende Weise berechnet: Das Signal wird quadriert, dann exponentiell gemittelt, dann wird wieder die Wurzel gezogen. Quadrierung, Mittelung und anschließende Radizierung sind die Merkmale einer Effektivwert-Bildung. Beim gewöhnlichen Effektivwert wird ein gleichmäßig gewichtetes Mittel aller Quadrate gebildet, während hier beim gleitenden Effektivwert eine zeitliche Bewertung durchgeführt wird. Die benutzte exponentielle Mittelung bewirkt ein "Vergessen". Sie kann als Filterung mit einem Tiefpass 1.Ordnung verstanden werden. Die Zeitkonstante für den gleitenden Effektivwert liegt laut DIN4150 Teil2 bei 0.125s.
- Die bewertete Schwingstärke (Ergebnis der Effektivwert-Berechnung) wird nun auf Maxima hin untersucht. Dabei wird in jedem Intervall der Länge 30s der Maximalwert gesucht. Das sind dann die Takt-Maximal-Werte.
- Nun wird noch eine Normierung durchgeführt. Das Signal wird mit einem Faktor multipliziert, um z.B. eine bestimmte Skalierung zu berücksichtigen.
- Alle Takt-Maximal-Werte unterhalb einer gewissen Schwelle werden auf Null gesetzt. Für eine eventuell nachfolgende Bildung der Takt-Maximal-Effektiv-Werte werden nämlich nur Werte größer 0.1 berücksichtigt, die anderen werden auf Null gesetzt.

Anwendung zur alleinigen Bildung von Takt-Maximal-Werten, die z. B. von Terz-Spektren berechnet werden sollen. Setzen Sie dazu:

```
EwFrequenzUnten = -1
EwFrequenzOben = -1
EwIntegrationen = 0
EwZeitBewertung = 0
EwTakt = Ihr Takt, z. B. 30s
EwNormierung = 0
EwMinimum = 0
```

Anwendung als Bandpass-Funktion (Achtung: auch der Hochpass mit 5.6 Hz Grenzfrequenz ist in der Filterung enthalten!). Setzen Sie dazu:

```
EwFrequenzUnten = Untere Grenz-Frequenz * 0.8
EwFrequenzOben = Obere Grenz-Frequenz / 0.8
EwIntegrationen = 0
EwZeitBewertung = 0
EwTakt = 0
EwNormierung = 0
EwMinimum = 0
```

Anwendung zur Berechnung eines gleitenden Effektivwertes. Setzen Sie dazu:

```
EwFrequenzUnten = -1
EwFrequenzOben = -1
EwIntegrationen = 0
EwZeitBewertung = Ihre Mittelungs-Zeit-Konstante oder -1 für 0.125s
EwTakt = 0
EwNormierung = 0
EwMinimum = 0
```

Die Algorithmen entsprechen:

- DIN 4150 Teil 2, Erschütterungen im Bauwesen, Einwirkungen auf Menschen in Gebäuden
- DIN 45669 Teil 1, Messung von Schwingungs-Immissionen, Schwingungsmesser, Anforderungen, Prüfung
- Die Bildung des Takt-Maximal-Effektiv-Wertes ist in dieser Funktion nicht enthalten. Sie kann mit der Funktion RMS() durchgeführt werden.

Achtung, Zeitkonstante!

Wenn Sie die gleitende Effektiv-Wert-Bildung benutzen, beachten Sie, dass die Wirkung anders ist als die eines einfachen Tiefpasses. Die Zeitkonstante, die Sie angeben, ist nicht direkt ablesbar. Denn das gefilterte Signal wird noch anschließend radiziert. Sie können aber die Zeitkonstante gut ablesen, wenn Sie das Ergebnis quadrieren! Die Zeitkonstante lässt sich z. B. gut bei rechteckigem Eingangssignal ablesen, wenn keine Filterung benutzt wird.

Abtastzeiten

Wenn die Funktion eine Filterung durchführt, wird ein digitales Filter benutzt. Die Grenzfrequenzen dieses digitalen Filters können nur kleiner als die halbe Abtastfrequenz sein. Höhere Grenzfrequenzen sind nicht möglich.

Genauigkeit der Filterung

Der Entwurf der digitalen Filter, die für den Bandpass und Hochpass benutzt werden, ist nur für Frequenzen deutlich unterhalb der halben Abtastfrequenz gut. Um überhaupt den Bandpass ausnutzen zu können, sollte die Abtastfrequenz bei einer oberen Grenz-Frequenz von 80Hz wenigstens 1KHz betragen.

Gleitender Effektivwert

Die Mittelungszeit darf größer oder kleiner als die Abtastzeit sein. Gerechnet wird mit einer Approximation nullter Ordnung (treppen-förmiges Signal).

Einschwingen

Beachten Sie beim Anwenden von Bandpässen das Einschwingen, dessen Dauer mit 1/Frequenz-Differenz abgeschätzt werden kann. Besonders bei der Wahl von Integrationen ist die untere Grenz-Frequenz entscheidend für das Einschwingen. Während dem Einschwingen der Filter ist das Ergebnis noch nicht aussagekräftig. Ignorieren Sie ggf. die ersten Werte des Ergebnis-Datensatzes. Das Einschwingen ist kein spezielles Problem von imc FAMOS oder von digitalen Filtern, sondern von Filtern ganz allgemein.

Zahlenbereich

Beachten Sie den zulässigen Zahlenbereich, wenn Sie einen Normierungsfaktor wählen. Bei Überschreitungen des Zahlenbereichs wird der entsprechende Wert des Ergebnis-Datensatzes auf Null gesetzt.

Einheit

Die x-Einheit des Quell-Datensatzes wird in Sekunden erwartet. Die der Funktion übergebenen Zeit-Angaben werden in derselben Einheit erwartet. Die Frequenz-Angaben werden in Hz erwartet. Das Ergebnis trägt stets die y-Einheit der Quell-Daten, auch bei Integrationen.

Datentyp

Der Typ des Ergebnisses ist ein normaler Datensatz. Wenn ein einziger Wert berechnet wird (ein Takt-Maximal-Wert), wird ein Einzelwert zurückgegeben.

Y-Skalierung

Die berechneten Werte werden stets linear angegeben. Wenn Sie eine Darstellung in dB wünschen, dann können Sie die mathematische Funktion dB() oder an den Kurvenfenstern eine entsprechende Darstellung wählen.

Beispiele:

Die Standard-Anwendung der Funktion, wobei ein Datensatz a mit Beschleunigungs-Messwerten vorliegt. Seine Abtastzeit beträgt 1ms. Ein Frequenz-Bereich von 1Hz bis 80Hz wird gewählt. Da das Signal von einem Beschleunigungs-Sensor kommt, ist eine einmalige Integration notwendig, um die Schnelligkeit zu erhalten. Die Fast-Bewertung (0.125s) wird für die Effektivwert-Bestimmung benutzt. Takt-Maximal-Werte werden im Abstand von 30s berechnet. Es wird nicht weiter normiert. Takt-Maximalwerte unterhalb von 0.1 werden auf 0 gesetzt. Anschließend wird mit einem weiteren Funktionsaufruf der Takt-Maximal-Effektivwert berechnet.

```
KBFTI = KBT(a, 1.0, 80.0, 1, -1, 30, 0, 0.1, 0)
KBFTM = RMS(KBFTI)
```

Um Zwischen-Ergebnisse betrachten zu können, kann der kompakte Aufruf der KBT-Funktion auch aufgebrochen werden, wobei zuerst die Filterung (Bandpass und Hochpass), dann die gleitende Effektivwert-Bildung und dann die Bestimmung der Takt-Maximal-Werte erfolgt.

```
FrRating = KBT(a, 1.0, 80.0, 1, 0, 0, 0, 0, 0)
rms = KBT(FrRating, -1, -1, 0, 0.125, 0, 0, 0.1, 0)
KBFTI = KBT(rms, -1, -1, 0, 0, 30, 0, 0, 0)
KBFTM = RMS(KBFTI)
```

Siehe auch:

ExpoRMS, RMS, OctA, ABCRating, MInt

KlsDurch

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Klassendurchgangsverfahren

Deklaration:

```
KlsDurch ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Bezug, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Bezug	Lage des Bezugsniveaus
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : Bezugsniveau automatisch
	3 : offene Randklassen, Bezugsniveau automatisch
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Klassendurchgangsverfahren (Level crossing counting) nach DIN 45667.

Dabei werden in den Klassen oberhalb der Bezugslinie alle von der Bezugslinie weg ins Positive gerichteten Klassendurchgänge in der dazugehörigen Klasse gezählt.

In den Klassen unterhalb der Bezugslinie werden alle von der Bezugslinie weg ins Negative gerichteten Klassendurchgänge in der dazugehörigen Klasse gezählt.

Beispiele:

```
Häufig = KlsDurch( Daten, 3, -3, 12, 0, 0.25, 1 )
```

Siehe auch:

ClsTimeAtLevel

KlsMaxSp

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Maximalwert-Speicherverfahren

Deklaration:

```
KlsMaxSp ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Stichprobenabstand, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Stichprobenabstand	
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Maximalwert-Speicherverfahren nach DIN 45667.

Das Maximalwert-Speicherverfahren ist eine Abwandlung des Stichprobenverfahrens.

Nach DIN 45667 wird jeweils der größte Abtastwert innerhalb eines bestimmten Zeitintervalls ermittelt und klassiert.

Im Klassier-Kit wird jeweils der Maximalwert von so vielen Abtastwerten bestimmt und klassiert, wie als Parameter Stichprobenabstand eingegeben werden.

Beispiele:

```
Häufig = KlsMaxSp( Daten, 12, 0, 12, 5, 1 )
```

Siehe auch:

ClsTimeAtLevel

KlsRFlow

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Rain-Flow-Verfahren

Deklaration:

```
KlsRFlow ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : Amplituden-Mittelwert-Darstellung
	3 : offene Randklassen, Amplituden-Mittelwert-Darstellung
Ergebnis	Histogramm

Beschreibung:

Zum Abschluss wird das Residuum in die Matrix gezählt.

Beispiele:

```
Häufig = KlsRFlow( Daten, 25, 10, 30, 0.5, 3 )
setseglang ( Häufig, 30 )
```

Standard-Aufruf. Die Matrix ist nicht skaliert.

```
_NumberClasses = 30
_Min = -5
_Max = 5
_option = 2+1 ;( +1 offene Rand, +2 Amp-Mean )
_Hysteresis = ( _Max - _Min ) / _NumberClasses
RainflowMatrix = KlsRFlow ( data_chan1, _Max, _Min, _NumberClasses, _Hysteresis, _option)
setseglang ( RainflowMatrix, _NumberClasses )
xoffset RainflowMatrix _Min
_dx = ( _Max - _Min ) / _NumberClasses
xdelta RainflowMatrix _dx
wenn _option >= 2
   ; Mittelwert entlang x-Achse
   ; Amplitude entlang z-Achse
   setzdel ( RainflowMatrix, _dx / 2 )
   setzoff ( RainflowMatrix, 0 )
   seteinheit ( RainflowMatrix, "Mean [Nm]", 0 ) ; x
   seteinheit ( RainflowMatrix, "Ampl [Nm]", 2 ) ; z
sonst
   setzdel ( RainflowMatrix, _dx )
   setzoff ( RainflowMatrix, _Min )
   seteinheit ( RainflowMatrix, "Nm", 0 ) ; x
   seteinheit ( RainflowMatrix, "Nm", 2 ) ; z
ende
seteinheit ( RainflowMatrix, "Count", 1 ) ; y
```

Die Rainflow-Matrix wird bestimmt.

Ein segmentierter Datensatz mit Skalierung wird erzeugt.

Siehe auch:

ClsOffRainflowInit1

KlsSMitt

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Spannen-Mittelwert-Verfahren

Deklaration:

```
KlsSMitt ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : Amplituden-Mittelwert-Darstellung
	3 : offene Randklassen, Amplituden-Mittelwert-Darstellung
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Spannen-Mittelwert-Verfahren (Range-mean counting) nach DIN 45667.

Als Spannen oder Bereiche werden die Abstände zwischen zwei benachbarten Extremwerten des Signals bezeichnet.

Die Spannen zwischen einem Minimum und dem nachfolgenden Maximum werden als positive Spannen und die Spannen zwischen einem Maximum und dem nachfolgenden Minimum werden als negative Spannen bezeichnet.

Im Gegensatz zum einparametrigen Spannenverfahren werden positive und negative Spannen nicht mehr getrennt, sondern gleichwertig klassiert, und außerdem wird der jeder Spanne zugehörige Mittelwert berücksichtigt.

Bei der Funktion KlsSMitt kann als Ergebnis der Klassierung entweder die Korrelationsmatrix oder die Amplituden-Mittelwert-Darstellung gewählt werden. Standardmäßig wird die Korrelationsmatrix als Ergebnis ausgegeben.

In der Korrelationsmatrix wird der Anfangswert jeder Spanne spaltenweise und der dazugehörige Endwert zeilenweise klassiert. Im Schnittpunkt von der Klasse, der der Anfangswert zugeordnet wird, und der Klasse, der der Endwert zugeordnet wird, erfolgt pro Spanne eine Zählung in der Korrelationsmatrix.

Bei der Amplituden-Mittelwert-Darstellung wird zu jeder Spanne deren Größe und der dazugehörige Mittelwert bestimmt.

In der Ergebnismatrix wird spaltenweise die Größe der Spanne und zeilenweise der dazugehörige Mittelwert klassiert. Im Schnittpunkt von der Klasse, der die Spannengröße zugeordnet wird, und der Klasse, der der dazugehörige Mittelwert zugeordnet wird, erfolgt pro Spanne eine Zählung in der Ergebnismatrix.

Spannen, die größer als 0 und kleiner oder gleich einer Klassenbreite sind, werden spaltenweise in der Klasse 1 gezählt.

Spannen, die größer als eine Klassenbreite und kleiner oder gleich zwei Klassenbreiten sind, werden spaltenweise in der Klasse 2 gezählt usw.

Beispiele:

```
Häufig = KlsSMitt( Daten, 50, 0, 50, 1, 1 )
```

Siehe auch:

ClsTimeAtLevel, KlsSpann, KlsRFlow

KlsSPaar

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Spannenpaarverfahren

Deklaration:

```
KlsSPaar ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : Präzise Berechnung
	3 : Präzise Berechnung mit Startwert
	4 : Spannen zwischen Extremwerten
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Spannenpaarverfahren (Range-pair counting) nach DIN 45667

Dabei werden von Talwerten aufwärts positive Spannen und von Gipfelwerten abwärts negative Spannen erfasst.

Eine Zählung wird aber erst dann vorgenommen, wenn zu einer positiven Spanne eine gleich große negative Spanne, also ein Spannenpaar, aufgetreten ist.

Eine kleinere Spanne kann mehrmals gezählt werden, bevor die Zählung einer größeren Spanne ausgelöst wird.

Bei der präzisen Berechnung ist der Algorithmus leicht optimiert. Dabei werden offene Randklassen angewendet.

Ist die präzise Berechnung nicht gewählt, ist der Algorithmus kompatibel zu den Vorgängerversionen.

Bei Berechnung mit Startwert wird der allererste Wert der Eingangsdaten auch als Extremwert angesehen. Ist diese Option nicht gewählt, wird mit einer Suche nach dem ersten echten relativen Extremwert begonnen.

Bei der Option "Spannen zwischen Extremwerten" wird eine präzise Berechnung mit Startwert durchgeführt. Aber die Spannen werden nur zwischen den Extremwerten gezählt. Liegt z.B. eine einzige Schwingung vor, so wird im Ergebnis auch nur eine einzige 1 geliefert. Nicht lauter Einsen bis hin zu dieser Klasse. Das entspricht nicht DIN 45667.

Beispiele:

```
Häufig = KlsSPaar( Daten, 24, -12, 36, 1, 3 )
```

Siehe auch:

ClsTimeAtLevel, KlsRFlow, KlsSpann, ClsOffFromRainflowGetRangePair

KlsSpann

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Spannenverfahren

Deklaration:

```
KlsSpann ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : negative Spannen
	3 : offene Randklassen, negative Spannen
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Spannenverfahren (Range counting) nach DIN 45667.

Als Spannen (Bereiche) werden die Abstände zwischen zwei benachbarten Extremwerten der Schwingung bezeichnet. Die Spannen zwischen einem Minimum und dem nachfolgenden Maximum (positive Spannen) und die Spannen zwischen einem Maximum und dem nachfolgenden Minimum (negative Spannen) werden getrennt klassiert, d.h. man erhält zwei Verteilungen.

Im Klassier-Kit wird entweder die Verteilung der positiven oder die Verteilung der negativen Spannen ermittelt.

Standardmäßig werden positive Spannen klassiert.

Beispiele:

```
Häufig = KlsSpann( Daten, 13, -1, 28, 0.5, 3 )
```

Siehe auch:

ClsTimeAtLevel, KlsSMitt, KlsSPaar

KlsSpit1

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Spitzenwertverfahren 1

Deklaration:

```
KlsSpit1 ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Bezug, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Bezug	Lage des Bezugsniveaus
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : Bezugsniveau automatisch
	3 : offene Randklassen, Bezugsniveau automatisch
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Spitzenwertverfahren 1 (Zero-crossing peak counting) nach DIN 45667.

Dabei werden nur die dem Betrag nach größten Spitzen zwischen zwei Durchgängen durch die Bezugslinie erfasst und der dazugehörigen Klasse zugeordnet.

Beispiele:

```
Häufig = KlsSpit1( Daten, 15, 5, 20, 0, 0.5, 3 )
```

Siehe auch:

ClsTimeAtLevel, KlsSpit2, KlsSpit3, ClsOffFromRainflowGetZeroCrossingPeak

KlsSpit2

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Spitzenwertverfahren 2

Deklaration:

```
KlsSpit2 ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Bezug, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Bezug	Lage des Bezugsniveaus
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : Bezugsniveau automatisch
	3 : offene Randklassen, Bezugsniveau automatisch
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Spitzenwertverfahren 2 (Peak counting) nach DIN 45667.

Es werden im Bereich oberhalb der Bezugslinie alle Maxima und im Bereich unterhalb der Bezugslinie alle Minima erfasst.

Beispiele:

```
Häufig = KlsSpit2( Daten, 28, 12, 32, 0, 0.5, 1 )
```

Siehe auch:

ClsTimeAtLevel, KlsSpit1, KlsSpit3, ClsOffFromRainflowGetPeak

KlsSpit3

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Spitzenwertverfahren 3

Deklaration:

```
KlsSpit3 ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : negative Spitzen
	3 : offene Randklassen, negative Spitzen
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Spitzenwertverfahren 3 nach DIN 45667.

Dabei werden alle Maxima und alle Minima unabhängig voneinander klassiert.

Als Ergebnis erhält man zwei Verteilungen.

Im Klassier-Kit wird entweder die Verteilung der Maxima oder die Verteilung der Minima ermittelt. Standardmäßig werden die Maxima klassiert.

Beispiele:

```
Häufig = KlsSpit3( Daten, 0.6, 0.1, 25, 0.02, 3 )
```

Siehe auch:

ClsTimeAtLevel, KlsSpit1, KlsSpit2, ClsOffFromRainflowGetMinMaxPeak

KlsSpSti

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Spitzenwert-/Stichprobenverfahren

Deklaration:

```
KlsSpSti ( Datensatz 1, Datensatz 2, Maximalwert, Minimalwert, Klassenanzahl, Hysterese, Optionen ) -> Ergebnis
```

Parameter:

Datensatz 1	Zu klassierender Datensatz 1
Datensatz 2	Zu klassierender Datensatz 2
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Hysterese	Hysterese zur Ausschaltung kleinerer Schwankungen
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Spitzenwert-/Stichprobenverfahren.

Dabei wird der Datensatz 1 mit dem Spitzenwertverfahren 3 bearbeitet.

Hier werden allerdings alle Maxima und alle Minima gleichzeitig klassiert.

Jedes Mal, wenn ein zu zählender Extremwert (Maximum oder Minimum) auftritt, wird gleichzeitig der diesem Extremwert des Datensatzes 1 entsprechende Abtastwert des Datensatzes 2 klassiert.

Ist beispielsweise der fünfte Abtastwert des ersten Datensatzes ein Extremum, wird dieses Extremum klassiert und gleichzeitig auch der fünfte Abtastwert des zweiten Datensatzes.

Anschließend wird der Datensatz 2 mit dem Spitzenwertverfahren 3 bearbeitet, wobei sowohl die Maxima als auch die Minima klassiert werden, und jedes Mal, wenn ein zu zählender Extremwert auftritt, wird auch der diesem Extremwert des Datensatzes 2 entsprechende Abtastwert des Datensatzes 1 klassiert.

Die Klassierung erfolgt in der Korrelationsmatrix, wobei spaltenweise der Datensatz 1 und zeilenweise der Datensatz 2 klassiert wird. Im Schnittpunkt von der Klasse, der der Extremwert des Datensatzes 1 bzw. 2 zugeordnet wird, und der Klasse, der der entsprechende Abtastwert des Datensatzes 2 bzw. 1 zugeordnet wird, erfolgt pro Extremwert des Datensatzes 1 bzw. 2 eine Zählung in der Korrelationsmatrix.

Beispiele:

```
Häufig = KlsSpSti( Daten, 10, -10, 40, 0.5, 1 )
```

Siehe auch:

ClsTimeAtLevel, KlsSpit3, KlsStSti

KlsStich

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Stichprobenverfahren

Deklaration:

```
KlsStich ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Stichprobenabstand, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Stichprobenabstand	
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : Stichprobenabstand zufällig
	3 : offene Randklassen, Stichprobenabstand zufällig
Ergebnis	Verteilung

Beschreibung:

Klassierung von Messdaten mit dem Stichprobenverfahren nach DIN 45667.

Dabei wird der Momentanwert der schwingenden Größe in konstanten oder in zwischen bestimmten Grenzen regellos schwankenden Zeitabständen festgestellt und klassenweise gezählt.

Beispiele:

```
Häufig = KlsStich( Daten, 20, -20, 40, 3, 3 )
```

Siehe auch:

ClsTimeAtLevel, KlsMaxSp

KlsStSti

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Stichproben-/Stichprobenverfahren

Deklaration:

```
KlsStSti ( Datensatz 1, Datensatz 2, Maximalwert, Minimalwert, Klassenanzahl, Stichprobenabstand, Optionen ) -> Ergebnis
```

Parameter:

Datensatz 1	Zu klassierender Datensatz 1
Datensatz 2	Zu klassierender Datensatz 2
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Stichprobenabstand	
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
	2 : Stichprobenabstand zufällig
	3 : offene Randklassen, Stichprobenabstand zufällig
Ergebnis	Verteilung

Beschreibung:

Die Messdaten werden mit dem Stichproben-/Stichprobenverfahren klassiert.

Dabei wird der Datensatz 1 mit dem Stichprobenverfahren bearbeitet. Beim Stichprobenverfahren wird in bestimmten Abständen der Momentanwert des Signals klassiert.

Jedes Mal, wenn ein Abtastwert des Datensatzes 1 klassiert wird, wird gleichzeitig der diesem Abtastwert entsprechende Abtastwert des Datensatzes 2 klassiert.

Wird beispielsweise der fünfte Abtastwert des ersten Datensatzes klassiert, dann wird gleichzeitig auch der fünfte Abtastwert des zweiten Datensatzes klassiert.

Die Klassierung erfolgt in der Korrelationsmatrix, wobei spaltenweise der Datensatz 1 und zeilenweise der Datensatz 2 klassiert wird. Im Schnittpunkt von der Klasse, der der zu klassierende Abtastwert des Datensatzes 1 zugeordnet wird (spaltenweise), und der Klasse, der der entsprechende Abtastwert des Datensatzes 2 zugeordnet wird (zeilenweise), erfolgt eine Zählung in der Korrelationsmatrix.

Beispiele:

```
Häufig = KlsStSti( Daten1, Daten2, 3, -3, 24, 5, 3 )
```

Siehe auch:

ClsTimeAtLevel, KlsSpSti, KlsStich

KlsVWeil

Verfügbar ab: Enterprise Edition (ClassCounting-Kit)

Klassierung mit dem Verweildauerverfahren

Deklaration:

```
KlsVWeil ( Datensatz, Maximalwert, Minimalwert, Klassenanzahl, Optionen ) -> Ergebnis
```

Parameter:

Datensatz	Zu klassierender Datensatz
Maximalwert	Obere Grenze des zu klassierenden Wertebereichs
Minimalwert	Untere Grenze des zu klassierenden Wertebereichs
Klassenanzahl	
Optionen	
	0 : keine Optionen
	1 : offene Randklassen
Ergebnis	Histogramm

Beschreibung:

Anstelle dieser Funktion sollte die neuere Funktion ClsTimeAtLevel() benutzt werden!

Klassierung von Messdaten mit dem Verweildauerverfahren nach DIN 45667.

Dabei wird die Summe der Zeiten, die die Schwingung innerhalb der Grenzen der einzelnen Klasse verbringt, für jede Klasse getrennt ermittelt.

Im Klassier-Kit wird jeder Abtastwert des Signals der zugehörigen Klasse zugeordnet.

Das Verweildauerverfahren ist der Spezialfall des Stichprobenverfahrens mit dem konstanten Stichprobenabstand von 1.

Das Verweildauerverfahren entspricht der klassischen Amplitudenverteilung bzw. dem Histogramm.

Beispiele:

```
Häufig = KlsVWeil( Daten, 10, -10, 40, 1 )
```

Siehe auch:

Histo, ClsOff2ChannelHistogram, ClsTimeAtLevel

KvAttrib

Anwendungsbereich: Kurvenfenster

Für ein Kurvenfenster wird ein Attribut zur Darstellung definiert.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvAttrib ( NDFensterDatenSatz, EWAttribut )
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EWAttribut	EWAttribut

Beschreibung:

Bitte benutzen Sie CwDisplaySet.

EWAttribut

0:Standard

1: Kurven übereinander

Addieren Sie

0: kein Gitter

2: Gitter

Addieren Sie

0: Beschriftung der Achsen

4: keine Beschriftung der Achsen

Addieren Sie

00: direktes Zeichen auf Schirm

10: Bitmap-Zeichnen. (Flackern reduzieren)

Addieren Sie

00: Trigger nicht anzeigen

20: Trigger anzeigen

Addieren Sie

00: automatisches Anpassen der Kurve

40: kein automatisches Anpassen. (wird ignoriert, nur noch imc FAMOS 2.x-kompatibel)

Addieren Sie

000:normales x-Achse

100: x-Achse für Roll-Mode (wird ignoriert, nur noch imc FAMOS 2.x-kompatibel)

Addieren Sie

000: x-Achse in Sekunden

200: x-Achse mit relativer Zeit (Stunden, Minuten,...)

Addieren Sie

000: x-Achse in Sekunden

400: x-Achse in absoluter Zeit (Datum, Uhrzeit)

Addieren Sie

1000*Anzahl Symbole (0..99) pro Darstellung. Nur für Symbole.

0 ist Standard für Symbol an jedem Messwert

Beispiele:

```
KvAttrib ( x, 1+400 )
```

Siehe auch:

CwDisplaySet, CwAxisSet

KvCursor

Anwendungsbereich: Kurvenfenster

Für ein Messwertfenster wird zu einem Cursor die Cursor-Position zurückgegeben.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvCursor ( NDFensterDatenSatz, EWCursor ) -> EWxWert
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EWCursor	EWCursor
	1 : x-Koordinate Cursor zu linker Maustaste
	2 : x-Koordinate Cursor zu rechter Maustaste
	3 : der (aktuell) linke Cursor (kleinerer Parameter)
	4 : der (aktuell) rechte Cursor (größerer Parameter)
	5 : Rückgabe = 1, wenn Messwertfenster existiert, sonst 0
	6 : Rückgabe = 1, wenn das Kurvenfenster existiert, sonst 0
	7 : Rückgabe = 1, wenn das Übersichtsfenster existiert, sonst 0
	8 : Rückgabe = 1 (x-Achse in Einheit der Daten), 2 (abs. Zeit), 3 (rel. Zeit), 4 (Terz), > 4 möglich. 0 (nicht vorhanden)
	9 : Rückgabe = 1 (Standard), 2 (y-Achsen übereinander), 3 (Wasserfall), 4 (Spektralansicht), 5 (Farbkarte), > 5 möglich. 0 (nicht vorhanden)
	10 : Rückgabe = xmin
	11 : Rückgabe = xmax
	12 : y-Koordinate Cursor zu linker Maustaste
	13 : y-Koordinate Cursor zu rechter Maustaste
	14 : z-Koordinate Cursor zu linker Maustaste
	15 : z-Koordinate Cursor zu rechter Maustaste
	16 : Parameter Cursor zu linker Maustaste
	17 : Parameter Cursor zu rechter Maustaste
EWxWert	EWxWert

Beschreibung:

Bitte benutzen Sie CwDisplayGet, CwIsWindow für die Option 5, CwAxisGet für die Optionen 10, 11.

Ist kein Messwertfenster vorhanden, wird 0 zurückgegeben.

Beispiele:

```
xLinks = KvCursor ( x, 1 )
```

Siehe auch:

CwDisplayGet

KvCursorSetzen

Anwendungsbereich: Kurvenfenster

Ein Messcursor wird an eine bestimmte Position gesetzt.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvCursorSetzen ( NDFensterDatenSatz, EWx, EWPara1, EWPara2, EWRechtsLinks ) -> EWFehler
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EWx	Die x-Koordinate, an die der Cursor zu setzen ist. Diese Koordinate wird ggf. auf das nächste Pixel gerundet. Die x-Achse muss bereits diese Koordinate darstellen. Ist die x-Achse in absoluter Zeit dargestellt, muss die x-Koordinate auch als solche vorgegeben werden. Bei XY-Darstellungen ist anstelle der x-Koordinate der Parameter vorzugeben. Dann wird ggf. auf einen groben Bruchteil des Parameter-Inkrements gerundet.
EWPara1	EWPara1
EWPara2	EWPara2
EWRechtsLinks	EWRechtsLinks
	1 : Cursor zu linker Maustaste
	2 : Cursor zu rechter Maustaste
EWFehler	Rückgabe ist EwFehler (optional)

Beschreibung:

Bitte benutzen Sie CwDisplaySet den Parametern measure.*.

Dazu muss das Messwertfenster bereits vorhanden sein. Ein nicht abgeschlossenes KvUpdate(0) darf vorher nicht aufgerufen worden sein.

EWPara1 = EWPara2 = 0, falls nicht anders vermerkt

Bei Farbspektraldarstellung ist EWPara1 die y-Koordinate. Ist das Fenster in dB skaliert, wird trotzdem nicht in dB vorgegeben.

Rückgabe:

0: wenn ok

1: wenn der Messcursor (gerade) nicht auf den Wert gesetzt werden kann.

Beispiele:

```
err = KvCursorSetzen ( Kanal1, 0.1, 0, 0, 1 )
```

Siehe auch:

CwDisplaySet

KvFenst

Anwendungsbereich: Kurvenfenster

Für ein gegebenes Kurvenfenster wird ein Messwert- oder Übersichtsfenster dargestellt bzw. geschlossen.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvFenst ( NDFensterDatenSatz, EWAuftrag )
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EWAuftrag	EWAuftrag
	1 : Messwertfenster anzeigen
	2 : Messwertfenster schließen
	3 : Übersichtsfenster anzeigen
	4 : Übersichtsfenster schließen
	5 : Kurvenfenster schließen, nicht für Kurven in Dialogen!
	6 : Kurvenfenster zum Symbol, nicht für Kurven in Dialogen!
	7 : Kurvenfenster zum Vollbild, nicht für Kurven in Dialogen!
	8 : Kurvenfenster nach Sinnbild oder Vollbild zur normalen Größe
	9 : Messwertfenster unsichtbar zeigen, nur Cursoren sichtbar
	10 : Navigator-Fenster anzeigen
	11 : Kommunikator-Fenster anzeigen

Beschreibung:

Bitte benutzen Sie CwAction.

Beispiele:

```
KvFenst ( x, 1 )
```

Siehe auch:

CwAction

KvGlobaleEinstellungLaden

Anwendungsbereich: Kurvenfenster

Die Funktion lädt eine globale Einstellung in den Kurvenmanager.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvGlobaleEinstellungLaden ( TXDateiname, EWEinstellung, EWParameter )
```

Parameter:

TXDateiname	TXDateiname
EWEinstellung	EWEinstellung
	1 : Alle Kurvenfenster werden ab sofort auf dem Bildschirm in den neuen Farben dargestellt. Dazu gehören alle Einstellungen des Farben-Dialogs.
	2 : Für alle folgenden Druckvorgänge, das Übertragen in den Reportgenerator, das Kopieren in die Ablage und den Grafikexport werden die neuen Farben übernommen.
	3 : Alle Einstellungen des Dialogs <Einstellungen Ablage> werden ab sofort gültig. Für alle folgenden Druckvorgänge, das Übertragen in den Reportgenerator, das Kopieren in die Ablage und den Grafikexport werden diese Einstellungen benutzt. Allerdings bleiben davon Reports unberührt, bei denen die aktuellen Einstellungen beim Transfer nicht übernommen werden sollen.
	4 : Einige Einstellungen des Dialogs <Kurvenfenster, Voreinstellungen...> werden geladen. Diese Einstellungen gelten für alle Kurvenfenster gemeinsam. Dazu zählen die Schrift- und Symbolgröße auf dem Bildschirm.
EWParameter	EWParameter

Beschreibung:

Bitte benutzen Sie CwLoadSettings.

Einstellungen werden aus einer Datei mit dem Namen TXDateiname geladen. Die Datei wird standardmäßig im CCV-Verzeichnis bei imc FAMOS erwartet. Einige der Optionen haben direkte Auswirkungen auf alle angezeigten Kurvenfenster. Wenn nicht anders vermerkt, dann EWParameter = 0 setzen.

Beispiele:

Die Datei c:\imc\set\farben.set wurde vom Kurvenfenster aus gespeichert.

```
KvGlobaleEinstellungLaden ( "..\set\farben.set", 1, 0 )
```

Siehe auch:

CwLoadSettings

KvKanalErsetzen

Anwendungsbereich: Kurvenfenster

Kanal ersetzen
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvKanalErsetzen ( NDFensterDatenSatz, NDErsatz, TXAlteBezeichnung ) -> AnzahlErsetzt
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
NDErsatz	Ersatz
TXAlteBezeichnung	Alte Bezeichnung
AnzahlErsetzt	Rückgabe ist die Anzahl der ersetzten Kanäle. (optional)

Beschreibung:

Bitte benutzen Sie CwReplace.

In einem Kurvenfenster wird ein Kanal mit der Bezeichnung TXAlteBezeichnung dargestellt. Dieser Kanal soll nun durch den Kanal NDErsatz in diesem Kurvenfenster ersetzt werden. Anschließend wird anstelle des Kanals mit der Bezeichnung TXAlteBezeichnung der Kanal NDErsatz dargestellt. Taucht der Kanal mehrfach auf, wird auch mehrfach ersetzt. Ist der Kanal gar nicht vorhanden, bleibt die Funktion ohne Wirkung. Gruppennamen werden mit ":" getrennt angegeben. Die Funktion wird angewendet, wenn nach dem Laden einer Kurvenfenster-Konfiguration andere Kanäle in dieser Konfiguration dargestellt werden sollen als einst dargestellt waren, als die Konfiguration gesichert wurde.

Beispiele:

Als x.ccv erstellt wurde, wurden im Kurvenfenster die Kanäle x1 und x2 dargestellt.

```
Fehler = KvKonfig ( x, "x.ccv")
Ersetzt = KvKanalErsetzen ( x, Kanal1, "x1" )
Ersetzt = KvKanalErsetzen ( x, Kanal2, "x2" )
```

Jetzt werden Kanal1 und Kanal2 im Kurvenfenster dargestellt.

Siehe auch:

CwReplace

KvKonfig

Anwendungsbereich: Kurvenfenster

Zeigt Datensatz mit in der Datei TXDateiname abgelegter Kurvendarstellung an.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvKonfig ( NDFensterDatenSatz, TXDateiname ) -> TXFehler
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
TXDateiname	TXDateiname
TXFehler	Rückgabe ist ein leerer Text, im Fehlerfall eine Fehlermeldung. (optional)

Beschreibung:

Bitte benutzen Sie CwLoadCCV.

Beispiele:

```
Fehler = KvKonfig ( x, "x.ccv" )
```

Siehe auch:

CwLoadCCV

KvMarkerAnhängen

Anwendungsbereich: Kurvenfenster

Ein Marker wird in ein Kurvenfenster gesetzt.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvMarkerAnhängen ( NDFensterDatenSatz, EWx, EWy, EWr, EWKoordinaten, EWKurvenIndex, TXText, EW1, EW2 ) -> EwFehler
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EWx	Die x-Koordinate
EWy	Die y-Koordinate
EWr	EWr
EWKoordinaten	EWKoordinaten
	1 : EWx und EWy in physikalischen Einheiten
	2 : EWx und EWy in Prozent der Achsenlängen
EWKurvenIndex	1, 2, 3, ... : Der Index der Linie, der der Marker in der Verwaltung zugeordnet wird. Bei Darstellung in mehreren Koordinatensystemen oder nach Umskalierung der Achsen von Bedeutung. Bei Zweifel = 1.
TXText	Diesen Text erhält der Marker.
EW1	EW1
EW2	EW2
EwFehler	Rückgabe ist EwFehler (optional)

Beschreibung:

Bitte benutzen Sie CwNewElement mit dem Parameter marker, anschließend CwMarkerSet..

Dieser neue Marker wird an die bereits bestehende Liste von Markern angehängt.

EWr = EW1 = EW2 = 0, wenn nicht anders angegeben.

Rückgabe:

0: ok

1: Fehler. Der Marker konnte nicht angelegt werden.

Beispiele:

```
err = KvMarkerAnhängen ( Kanal1, 10, 20, 0, 2, 1, "Einschaltvorgang", 0, 0 )
```

Siehe auch:

CwNewElement, CwMarkerSet

KvPosi

Anwendungsbereich: Kurvenfenster

Einstellung von Position und Größe eines Kurvenfensters
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvPosi ( NDFensterDatenSatz, EWX, EWY, EWdX, EWdY )
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EWX	EWX
EWY	EWY
EWdX	EWdX
EWdY	EWdY

Beschreibung:

Bitte benutzen Sie CwPosition oder CwDisplaySet mit den Parametern win.x, win.dx etc..

EWX, EWY - Obere, linke Ecke des Fensters

EWdX, EWdY - Größe des Fensters

Beispiele:

```
KvPosi ( x, 0, 0, 640, 480 )
```

Siehe auch:

CwPosition

KvRefDB

Anwendungsbereich: Kurvenfenster

Der Bezugswert für Darstellungen in DB wird definiert.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvRefDB ( NDFensterDatenSatz, EwDB, Auftrag )
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EwDB	EwDB
Auftrag	Auftrag
	0 : NDFensterDatenSatz ist 0 zu setzen. EwDB wird lediglich gemerkt und beim Erzeugen weiterer Kurvenfenster gleich in diese Kurvenfenster als Bezugswert eingetragen.
	1 : NDFensterDatenSatz gibt einen Datensatz an, der als Kurvenfenster dargestellt wird. EwDB wird in diesem Kurvenfenster als neuer Bezugswert benutzt. Ist kein Kurvenfenster dargestellt, hat die Funktion keine Wirkung.

Beschreibung:

Bitte benutzen Sie CwDisplaySet.

Vor dem ersten Aufruf der Funktion wird der Wert entsprechend den Voreinstellungen der Kurvenfenster benutzt. Es dürfen nur Zahlen größer null angegeben werden. EwDB ist der neue Bezugswert. Dieser Bezugswert wird lediglich in Kurvenfenstern benutzt, nicht aber bei mathematischen Berechnungen.

Beispiele:

```
KvRefDB ( 0, 0.075, 0 )
KvRefDB( ANSTIEG, 378.0, 1 )
```

Siehe auch:

CwDisplaySet

KvSicher

Anwendungsbereich: Kurvenfenster

Die Konfiguration dieses Kurvenfensters wird in der Datei Dateiname gesichert.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvSicher ( NDFensterDatenSatz, TXDateiname ) -> TXFehler
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
TXDateiname	TXDateiname
TXFehler	Rückgabe ist ein leerer Text, im Fehlerfall eine Fehlermeldung. (optional)

Beschreibung:

Bitte benutzen Sie CwSaveCCV.

Standard-Verzeichnis ist das Kurven-Konfigurationsverzeichnis (ccv) von imc FAMOS, Standard-Dateinamen-Erweiterung ist ".CCV". Der Rückgabewert ist ein leerer Text, wenn kein Fehler aufgetreten ist, sonst eine Fehlermeldung.

Beispiele:

```
Fehler = KvSicher ( ANSTIEG, "an.ccv" )
```

Siehe auch:

CwSaveCCV

KvTitel

Anwendungsbereich: Kurvenfenster

Titel setzen
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvTitel ( NDFensterDatenSatz, TXTitel, EwAuftrag )
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
TXTitel	Titel
EwAuftrag	Auftrag
	1 : Falls der Datensatz NDDatensatz als Kurvenfenster dargestellt ist, wird die Titelleiste des Kurvenfensters auf TXTitel gesetzt. Diese Einstellung gilt nur, solange das Kurvenfenster vorhanden ist. Wenn das Fenster geschlossen wird, ist diese Einstellung verloren.
	2 : Der Name der Variablen NDDatensatz wird für alle Anzeigen verändert. Wenn die Variable NDDatensatz angezeigt, gesichert oder exportiert wird, wird stets der neu definierte Name benutzt.

Beschreibung:

Bitte benutzen Sie CwDisplaySet mit dem Parameter title für den Auftrag 1, rename für den Auftrag 2.

Beispiele:

```
KvTitel ( x, "Ergebnis der Spektralanalyse:", 1 )
```

Siehe auch:

CwDisplaySet

KvTitelAbfrage

Anwendungsbereich: Kurvenfenster

Titel abfragen
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvTitelAbfrage ( NDFensterDatenSatz, EwAuftrag ) -> TXTitel
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EwAuftrag	EwAuftrag
	1 : Falls der Datensatz NDDatensatz als Kurvenfenster dargestellt ist, wird die Titelleiste des Kurvenfensters abgefragt.
	2 : Der Name der Variablen NDDatensatz wird abgefragt. Der Name kann vorher mit der Funktion KvTitel() verändert worden sein. Eine eventuelle Gruppenzugehörigkeit wird ignoriert.
TXTitel	TXTitel

Beschreibung:

Bitte benutzen Sie CwDisplayGetText mit dem Parameter title für den Auftrag 1, CwDataGetText für den Auftrag 2.

Beispiele:

```
xTitel = KvTitelAbfrage ( x, 1)
```

Siehe auch:

CwDisplayGetText

KvVar

Anwendungsbereich: Kurvenfenster

Variable tauschen
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvVar ( NDAlt, NDNeu )
```

Parameter:

NDAlt	NDAlt
NDNeu	NDNeu

Beschreibung:

Die Variable NDAlt wird vor Aufruf der Funktion als Kurvenfenster dargestellt. Nach Aufruf dieser Funktion wird anstelle der Variable NDAlt nun die Variable NDNeu im selben Kurvenfenster dargestellt. Das Kurvenfenster wird also einer neuen Variablen zugeordnet. Falls der erste Datensatz im Fenster NDAlt war, wird er durch NDNeu ersetzt.

Beispiele:

```
KvVar ( ANSTIEG, BOGEN )
```

KvXAchse

Anwendungsbereich: Kurvenfenster

Einstellung des dargestellten X-Achsen-Bereichs.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvXAchse ( NDFensterDatenSatz, EWxMin, EWxMax, EWAttribut )
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
EWxMin	xMin
EWxMax	xMax
EWAttribut	EWAttribut

Beschreibung:

Bitte benutzen Sie CwAxisSet.

EWAttribut:

-1: altes Attribut bleibt bestehen

0: linear

1: logarithmisch

Addieren Sie

00: feste Skalierung mit xMin, xMax

10: xMin, xMax runden

20: x-Achse auto, alle Kurven darstellen. xMin, xMax werden ignoriert

30: x-Achse auto, alle Kurven darstellen (runden). xMin, xMax werden ignoriert

40: wie 20

50: wie 30

Addieren Sie

100*Markierungen (Markierungen >= 2 und <= 99, Markierungen=0 für automatisch)

Falls die x-Achse in absoluter Zeit dargestellt ist, arbeitet die Funktion kompatibel zu lange zurück liegenden Versionen: Ist die Variable, die das Fenster identifiziert, ein Datensatz, addiert die Funktion dessen absolute Zeit zu den angegebenen Werten von xMin und xMax. Ist dieses Verhalten nicht gewünscht, so kann die abs. Zeit des Bezugsdatensatzes auf null gesetzt werden. Oder eine Textvariable wird als Bezug verwendet.

Beispiele:

```
KvXAchse ( x, 0, 1, 0+10+100*5 )
```

Siehe auch:

CwAxisSet

KvYAchse

Anwendungsbereich: Kurvenfenster

Eine y-Achse eines Kurvenfensters wird skaliert.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten!

Deklaration:

```
KvYAchse ( NDFensterDatenSatz, NDDatenSatz, NDKomponente2, EWyMin, EWyMax, EWAttribut, EWPosition )
```

Parameter:

NDFensterDatenSatz	identifiziert das Kurvenfenster
NDDatenSatz	der Datensatz, der ergänzt bzw. skaliert wird.
NDKomponente2	Für XY- Darstellung die 2. Komponente.
EWyMin	yMin
EWyMax	yMax
EWAttribut	EWAttribut
EWPosition	EWPosition

Beschreibung:

Bitte benutzen Sie CwAxisSet.

EWAttribut:

-1 altes Attribut bleibt bestehen

-2 DatenSatz wird aus dem Kurvenfenster entfernt. Position egal

sonst:

0: reell

1: y-Achse in XY-Darstellung. x-Achse ist letzter Datensatz, der x-Datensatz muss bereits im Fenster dargestellt sein! Sollte nicht mehr benutzt werden. Stattdessen 3 und 4. Nur noch zur Kompatibilität!

2: Ortskurve, NDDatensatz muss komplex sein, NDKomponente2 = 0

3: XY-Darstellung: NDDatensatz ist x-Achse, NDKomponente2 ist y-Achse

4: XY-Darstellung: NDDatensatz ist y-Achse, NDKomponente2 ist x-Achse

Addieren Sie

00: linear

10: logarithmisch

20: logarithmisch in Dezibel

Addieren Sie

000: feste Skalierung

100: yMin, yMax werden gerundet

200: automatisch.

300: automatisch mit Null-Linie

400: wie vorhergehende y-Achse

Addieren Sie

0000: Linien

1000: Punkte

2000: Treppen

3000: Balken

4000: Symbol (Quadrat, etc.) mit Linien

5000: Symbol ohne Linien

Addieren Sie

00000: Quadrat

10000: Kreis

20000: Dreieck1

30000: Dreieck2

40000: Raute

50000: +

60000: x

70000: dicker Punkt

80000: horiz. Linie

Addieren Sie

100000*Markierungen (Markierungen >= 2 und <= 99, Markierungen=0 für automatisch)

Position:

-1 Position bleibt erhalten

-2 als letzte Kurve anhängen

sonst:

1, 2, 3, 4... Index der Kurve. An dieser Stelle erscheint der neue Datensatz. (1=Basiskurve)

Beispiele:

Datensatz a im Fenster a wird mit 5 Ticks von -10 ... +10 dargestellt.

```
KvYAchse ( a, a, 0, -10.0, 10.0, 5*100000, 1 )
```

Anhängen von Datensatz b an Kurvenfenster a

```
KvYAchse ( a, b, 0, 0, 0, 200, -2 )
```

Anhängen von xy-Darstellung aus x und y an Kurvenfenster a

```
KvYAchse ( a, x, y, 0, 0, 200+3, -2 )
```

Siehe auch:

CwAxisSet, CwLineSet

LAYOUT

Reportgenerator-Fernsteuerung
Alternativer Name: **FORMULAR**

Der Befehl ist veraltet, statt dessen sollten die leistungsfähigeren Befehle des Reportgenerator-Kits wie RgDocOpen(), RgDocPrint() und RgCurveSet()/RgTextSet() verwendet werden.

Deklaration:

```
LAYOUT Aufgabe Name Variable
```

Parameter:

Aufgabe	Auszuführendes Kommando
	PRINT : Der aktuelle Report wird gedruckt
	LOAD : Laden einer Report-Datei
	OBJECT : Eine Variable wird ein Report-Objekt übertragen
Name	Für LOAD und OBJECT: Dateiname bzw. Objekttitel
Variable	Für OBJECT: Zu übertragende Variable.

Beschreibung:

```
LAYOUT PRINT
Der aktuelle Report wird ausgedruckt.
```

```
LAYOUT LOAD
```

Eine Report-Konfigurationsdatei wird in den Reportgenerator geladen. Wenn Sie im Report Platzhalter benutzen, rufen Sie diesen Befehl auf, um alle Platzhalter wieder herzustellen.

Der Reportgenerator wird als Symbol gestartet. Da Sequenzen den Reportgenerator nur in Zusammenhang mit zu ladenden Konfigurationen benötigen, braucht er auch nicht separat aufgerufen werden.

Der angegebene Dateiname darf auch ein vollständiger Pfadname sein, falls sich die gewünschte Konfigurationsdatei nicht im Standardverzeichnis befindet. Das Standardverzeichnis für Reporte stellen Sie entweder am Reportgenerator selbst oder im Dialog "Optionen"/ "Verzeichnisse" ein.

```
LAYOUT OBJECT
```

Eine Variable wird in ein Report-Objekt übertragen. Das gewünschte Objekt im Report wird durch seinen Titel (Namen) ausgewählt. Variablen können in Text- und Kurvenobjekte übertragen werden. Vor dem Übertragen eines Kurven-Objektes muss der Datensatz in einem Kurvenfenster dargestellt sein.

Beispiele:

```
LOAD Daten.dat
SHOW Daten
Txt = "Erstellt durch E.Mustermann"
LAYOUT LOAD c:\imc\drb\Drb1.drb
LAYOUT OBJECT Kurve Daten
LAYOUT OBJECT Name Txt
LAYOUT PRINT
```

Ein Report wird geladen. Ein Kurven- und ein Textobjekt werden zum Report übertragen und danach ausgedruckt.

Siehe auch:

RgDocOpen, RgDocPrint, RgCurveSet, RgTextSet

LDIR

Verzeichnis zum Laden von Dateien setzen

Deklaration:

```
LDIR Verzeichnis
```

Parameter:

Verzeichnis	Vollständiger Pfadname des gewünschten Verzeichnisses.

Beschreibung:

Statt dem Kommando LDIR sollte in neu zu erstellenden Sequenzen die Funktion SetOption() verwendet werden.

Das Verzeichnis zum Laden von Dateien wird neu gesetzt.

Nach Ausführung dieses Befehls wird dieses Verzeichnis zum Laden von Dateien benutzt.

Der Verzeichnisname darf auch in Anführungszeichen angegeben werden. Dies ist dann zwingend notwendig, wenn Leerzeichen im Namen enthalten sind.

Dieses Kommandos kann auch ohne Parameter (also ohne Verzeichnisangabe) aufgerufen werden. Dann wird das unter "Optionen/Verzeichnisse" eingestellte Verzeichnis wieder als aktueller Standard verwendet.

Das hier gewählte Verzeichnis bleibt gültig bis:

- der Befehl LDIR erneut aufgerufen wird
- die Funktion SetOption("Dir.DataFiles",...) aufgerufen wird
- mit dem Dialogfeld "Laden" im Menü "Datei" eine Datei aus einem anderen Verzeichnis geladen wird
- mit dem Dialog "Optionen"/ "Verzeichnisse" ein neues Verzeichnis zum Laden von Dateien definiert wird

Multithreading: Der Befehl wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
LOAD DATEN1
LDIR C:\VERSUCH
LOAD DATEN2
```

Aus dem aktuell eingestellten Verzeichnis wird die Datei DATEN1 im imc/FAMOS-Format geladen, anschließend die Datei C:\VERSUCH\DATEN2.DAT.

```
LDIR "c:\Meine Versuche vom 12.1.98"
```

Der Pfadname enthält Leerzeichen und muss deshalb in Anführungszeichen geschrieben werden.

Siehe auch:

SetOption, LOAD, SDIR, MDIR, FileLoad, FileOpenDSF

Leng

Setzt die Anzahl der Werte eines Datensatzes
Alternativer Name: **Lang**

Deklaration:

```
Leng ( Daten, EwLang ) -> Ergebnis
```

Parameter:

Daten	Datensatz, dessen Länge verändert werden soll.
EwLang	Länge des zu bildenden Datensatzes
Ergebnis	Datensatz mit geänderter Länge.

Beschreibung:

Die Länge (Punktezahl) eines Datensatzes wird auf einen neuen Wert gesetzt. Ist die neue Punktezahl kleiner als die alte, wird der Datensatz abgeschnitten (verkürzt). Ist die neue Punktezahl größer, wird der Datensatz mit Nullen bis zur neuen Länge aufgefüllt (bei skalierbaren ganzzahligen Datenformaten wird der unskalierte Zahlenwert auf 0 gesetzt).

- Der zweite Parameter sollte keine Einheit haben, da er eine reine Zahl darstellt.
- Die neue Punktezahl des Datensatzes kann nur eine ganze Zahl >= 0 sein.
- Die neue Länge des Datensatzes ist eine Punktezahl, keine x-Koordinate/-Ausdehnung.

Beispiele:

Die ersten 200 Punkte eines Datensatzes bilden einen neuen Datensatz:

```
NDnew = Leng(NDold, 200)
```

Ein 600 Punkte langer Datensatz wird auf 512 Punkte abgeschnitten und danach mit Nullen auf 1024 Punkte verlängert. Wird dieser Datensatz in einer AKF oder KKF benutzt, wird das Signal als einmalig, d. h. nicht periodisch interpretiert.

```
NDHalfZeroes = Leng(Leng(NDdata, 512), 1024)
```

Ein Datensatz mit 1024 Punkten wird mit einem sin(x) / x - Interpolator auf 2048 Punkte erweitert. Das wird erreicht, indem das Spektrum mit Nullen verlängert und wieder rücktransformiert wird. Benutzen Sie das Rechteckfenster!

```
NDintpol = iFFT(Leng(FFT(NDdata), 1025))
```

Siehe auch:

Leng?, MatrixChangeDim, Cut, XDel, XOff, Red, IPol

Leng?

Ermittelt die Länge (Anzahl der Werte) eines Datensatzes
Alternativer Name: **Lang?**

Deklaration:

```
Leng? ( Daten ) -> EwLänge
```

Parameter:

Daten	Datensatz, dessen Länge ermittelt werden soll.
EwLänge	Länge (Punktezahl) des Parameters.

Beschreibung:

Die Länge (Punktezahl) eines Datensatzes wird ermittelt.

Es wird eine Punktezahl zurückgegeben, keine x-Koordinaten-Differenz.

Beispiele:

Die Dauer eines Datensatzes wird bestimmt (Produkt aus Abtastzeit und Punktezahl):

```
duration = XDel?(NDdata) * Leng?(NDdata)
```

Ein Histogramm wird über die Anzahl der Punkte des untersuchten Datensatzes auf 100% normiert:

```
NDnormhi= 100'%' * Histo(NDdata, 0, 0)/Leng?(NDdata)
```

Die Länge eines Einzelwertes wird zu Eins definiert. Ein leerer Datensatz hat die Länge Null.

```
EwEins = Leng?(100)
EwNull = Leng?(EMPTY)
```

Siehe auch:

Leng, XDel?, XOff?, Time?, MatrixInfo

LFit

Lineare Regression, Anpassung an Gerade
Alternativer Name: **Regr**

Deklaration:

```
LFit ( Daten ) -> Gerade
```

Parameter:

Daten	Datensatz, zu dem eine anpassende Gerade gefunden werden soll. [ND],[XY]
Gerade	Regressionsgerade

Beschreibung:

Es wird ein Datensatz entsprechend der Gleichung

```
f(x) = A * x + B
```

bestimmt, der den gegebenen Datensatz am besten annähert. Die Koeffizienten A und B werden entsprechend bestimmt. Zugrunde liegt der Algorithmus der linearen Regression, der im Sinne kleinster Fehler-Quadrate eine optimale Gerade bestimmt.

Nach Beendigung der Funktion wird die ermittelte Gleichung in die Ausgabebox im FAMOS-Hauptfenster ausgegeben. Die Gleichung wird mit Einheiten gezeigt und hat zum Beispiel die Form:

```
f(x) = 5.834 [Ohm] * x + 12.29 [V]
```

Ein Aufruf der Funktion mit einem leeren Text als Parameter liefert die zuletzt berechneten Koeffizienten A, B in Form eines Datensatzes mit 2 Werten.

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Normaler Datensatz [ND]

Der erzeugte Datensatz hat die Länge und Einheit des übergebenen Datensatzes, aber maximal die Länge 100. Bei Kürzung der Länge wird die Abtastzeit verändert, so dass der erzeugte Datensatz über demselben Intervall definiert ist.

XY-Datensatz [XY]

Der erzeugte Datensatz hat die Länge 2. Er ist über demselben x-Intervall wie der Quelldatensatz definiert.

Beispiele:

Die Regressionsgerade für einen Datensatz wird berechnet. Anschließend werden noch Anstieg und Offset der Regressionsgeraden ermittelt:

```
RegressionLine = LFit(Daten)
coeff = LFit("")
factor= coeff[1]
offset = coeff[2]
```

Die Abweichung zur Bestgeraden wird ermittelt. Damit die Differenz gebildet werden kann, wird mit der Funktion RSamp die Abtastzeit angeglichen:

```
NDerror = NDdata - RSamp(LFit(NDdata), NDdata)
```

Siehe auch:

eFit, RSampEx, Appro, ApproNonLin, Poly

Lip

Lineare Interpolation
Alternativer Name: **IPli**

Deklaration:

```
Lip ( Daten, EwFaktor ) -> Interpoliert
```

Parameter:

Daten	Datensatz, der linear interpoliert werden soll. [ND],[XY]
EwFaktor	Faktor, um den der Datensatz zu vergrößern ist.
Interpoliert	Linear interpolierter Datensatz

Beschreibung:

Der übergebene Datensatz wird linear interpoliert. Dabei gibt der 2. Parameter den Faktor an, um den der Datensatz zu vergrößern ist. Bei einer linearen Interpolation wird angenommen, dass die Punkte des Datensatzes durch schräge Geraden verbunden sind. Es werden nur Zwischenwerte berechnet. Es wird nicht extrapoliert.

- Die Einheiten bleiben unverändert.
- Bei äquidistant abgetasteten Daten [ND] ist die Abtastzeit des erzeugten Datensatzes um den übergebenen Faktor kleiner. Die Interpolation kann hier durch die Funktion Red() rückgängig gemacht werden.
- Der übergebene Faktor darf nur eine ganze Zahl größer 1 sein.
- Da nicht extrapoliert wird, wird der Datensatz in seiner Länge nicht ganz um den übergebenen Faktor vergrößert. Es fehlen (Faktor -1) Werte. Bei grafischer Darstellung von Kurven wird i. a. linear interpoliert, so dass diese Funktion für eine grafische Darstellung nicht benutzt zu werden braucht.

Beispiele:

Ein Datensatz wird in seiner Punktedichte um den Faktor 3 erhöht:

```
data2 = Lip(data, 3)
```

Siehe auch:

IPol, MatrixIPol, Red, Leng

ln

Natürlicher Logarithmus zur Basis e (Eulersche Zahl).

Deklaration:

```
ln ( Daten ) -> Ergebnis
```

Parameter:

Daten	Daten. Erlaubte Typen: [ND],[XY].
Ergebnis	Natürlicher Logarithmus des Parameters.

Beschreibung:

Der Logarithmus zur Basis e (Eulersche Zahl) wird gebildet, e = 2.718.... Dieser Logarithmus wird als natürlicher Logarithmus bezeichnet.

Anmerkungen

- Die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Der Parameter der Funktion sollte keine Einheit haben. Hat er jedoch eine, bleibt sie erhalten (Warnung wird erzeugt).
- Der Parameter der Funktion ln() sollte stets größer Null sein.
- Die Funktionen exp() und ln() sind Umkehrfunktionen.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Der natürliche Logarithmus der vordefinierten Konstante e ist gleich 1:

```
one = ln(e)
```

Die folgende Formel verändert die Zahl nicht. Beachten Sie jedoch den eingeschränkten Definitionsbereich der Funktion exp():

```
number = ln(exp(number))
```

Siehe auch:

log, exp, ^(hoch)

LOAD

Datei laden
Alternativer Name: **LADEN**

Deklaration:

```
LOAD Dateiname Variablenname
```

Parameter:

Dateiname	Name der zu ladenden Datei
Variablenname	Name, unter dem die Variable in die Variablenliste aufgenommen wird.

Beschreibung:

Dieser Befehl ist veraltet, statt dessen können Sie auch die leistungsfähigere und bequemer anzuwendende Funktion FileLoad() verwenden.

Die angegebene Datei wird im aktuell eingestellten Dateiformat (FAMOS, BINARY oder ASCII) geladen und die enthaltenen Datenobjekte in die Variablenliste eingetragen. Es kann ein vollständiger Dateiname inklusive Pfad angegeben werden. Fehlt der zweite Parameter, wird als Variablenname der Dateiname ohne Verzeichnisangabe und ohne Erweiterung benutzt.

Bei imc FAMOS-Dateien, die mit der Version 3.0 erstellt wurden, sind die Variablennamen mit in der Datei abgespeichert. Die Variablen werden dann beim Laden nach imc FAMOS unter dem gleichen Namen wieder angelegt. Wenn der 2. Parameter angegeben ist und die Datei genau 1 Datenobjekt enthält, wird dieses unter dem angegebenen Namen angelegt. Bei mehreren Datenobjekten werden alle Objekte gelesen und der angegebene Variablenname ignoriert.

Wildcards

Sie können im Dateinamen Jokerzeichen (Wildcards) angeben, um eine Reihe von Dateien zu laden. Das Jokerzeichen '?' steht dabei für genau ein beliebiges Zeichen, das Jokerzeichen '*' für eine unbestimmte Anzahl von beliebigen Zeichen.

```
LOAD c:\imc\*.*
```

Im Verzeichnis c:\imc werden alle Dateien geladen.

```
LOAD a??.*
```

Alle Dateien im aktuellen Verzeichnis werden geladen, deren Name aus 3 Zeichen, beginnend mit einem 'a', besteht. Die Dateinamens-Erweiterung ist beliebig.

```
LOAD *a1*.dat
```

Alle Dateien mit der Erweiterung 'dat', in deren Namen die Zeichenfolge '1a' (auch zu Beginn oder am Ende) vorkommt, werden geladen.

Indirekter Aufruf und Dateinamen mit Sonderzeichen

Steht der Dateiname in einer Textvariablen erfolgt der Aufruf in spitzen Klammern:

```
MeinDateiname= "C:\Daten\Signal.dat"
LOAD <MeinDateiname>
```

Befinden sich im Dateinamen Leerzeichen, Punkte oder andere Sonderzeichen ist der indirekte Aufruf über eine Textvariable zwingend. Zusätzlich muss die Variable mit weiteren Anführungsstrichen umschlossen werden:

```
MeinDateiname= """C:\Daten\1.1 Signal bei 20°C.dat"""
LOAD <MeinDateiname>
```

- Ist gerade das imc/FAMOS-Dateiformat eingestellt, braucht keine Dateinamen-Erweiterung angegeben zu werden. Die Erweiterung ".DAT" wird automatisch ergänzt.
- Möchten Sie eine Datei ohne Erweiterung angeben, beenden Sie den Dateinamen mit einem Punkt.
- Wenn Sie keinen Pfad angeben, wird der für das Laden von Dateien eingestellte Pfad benutzt. Dieser Pfad steht nach dem Aufstart von FAMOS auf dem im Dialog "Optionen"/ "Verzeichnisse" eingestellten Standardpfad und kann z. B. mit dem Befehl LDIR oder der Funktion SetOption() umgestellt werden.
- Der Dateiname darf auch in Anführungszeichen angegeben werden. Dies ist z. B. dann notwendig, wenn der Pfad Leerzeichen enthält.
- Um Dateien zu laden, deren Format Sie mit dem imc-Datei-Assistenten definiert haben, verwenden Sie bitte die Funktion FileOpenFAS() oder den Befehl FASLOAD.
- Zum Laden von Dateien in beliebigen Formaten können Sie auch die Funktionen FileOpen*() benutzen. Diese sind wesentlich leistungsfähiger als der Befehl LOAD, insbesondere beim Laden von Dateien, die mehrere Datensätze enthalten.

Beispiele:

```
FAMOS
LOAD c:\imc\dat\Daten var1
```

Die Datei DATEN.DAT wird im imc FAMOS-Format geladen und die resultierende Variable unter dem Namen "var1" in die Variablenliste

aufgenommen.

```
ASCII
LOAD DBAS.ASC
```

Die angegebene ASCII-Datei wird geladen und unter dem Namen DBAS in die Variablenliste eingetragen.

```
BINARY
LDIR c:\BINDATA
LOAD *.BIN
```

Im angegebenen Verzeichnis werden alle Dateien mit der Erweiterung 'BIN' im Binärformat geladen.

```
LOAD "c:\imc\Meine Daten\Daten"
```

Der Dateinamen enthält Leerzeichen und muss deshalb in Anführungszeichen angegeben werden.

Siehe auch:

FileLoad, FileOpenDSF, LDIR, FAMOS, BINARY, ASCII

LOCAL

Die Anweisung dient zur Deklaration lokaler Variablen

Deklaration:

```
LOCAL Variablenname
```

Parameter:

Variablenname	Ein oder mehrere Variablennamen (ggf. mit Jokerzeichen)

Beschreibung:

Variablen können innerhalb einer Sequenz als 'lokal' deklariert werden. Lokale Variablen sind nur wärend der Ausführung der aktuellen Sequenz gültig und können nur innerhalb dieser verwendet werden, am Ende der Sequenzabarbeitung werden solche Variablen automatisch gelöscht.

Die Deklaration kann entweder direkt bei der Zuweisung erfolgen:

```
LOCAL temp =  Ramp(0, 1, 100)
```

Oder per Vorwärtsdeklaration:

```
LOCAL temp
temp =  Ramp(0, 1, 100)
```

Bei der Vorwärtsdeklaration können auch Jokerzeichen ('*' - beliebig viele beliebige Zeichen, '?' - genau ein beliebiges Zeichen) verwendet werden:

```
LOCAL ?    ; alle Variablen, deren Name genau ein Zeichen lang ist
LOCAL #*   ; alle Variablen, deren Name mit '#' beginnt
```

Es können auch mehrere Namensmuster, durch Leerzeichen oder Komma getrennt, angegeben werden:

```
LOCAL ?,#*
```

Benennung

Der intern verwendete (komplette) Name von lokalen Variablen wird durch [Name]@![Sequenzname] gebildet. Unter Ausnutzung des Messungskonzeptes in FAMOS werden die lokalen Variablen also einer 'virtuellen' Messung zugeordnet, deren Name aus dem Namen der aktuellen Sequenz abgeleitet wird. In der Variablensliste werden sie auch unter dem vollen Namen gelistet (z.B. bei der Ausführung der Sequenz im Einzelschrittmodus oder wenn eine Sequenzausführung wegen eines Fehlers abgebrochen wurde).

Namensdopplungen mit permanenten (globalen) Variablen:

- Bei der Erzeugung von lokalen Variablen wird eine eventuell bereits existierende permanente Variable mit dem selben Namen ignoriert.
- Bei der Suche nach Variablen 'gewinnt' die lokale Variable über eine eventuell vorhandene permanente Variable gleichen Namens.

```
X = 1  ; permanente Variable
LOCAL X
X = 2  ; neue lokale Variable
Y = X  ; Y hat den Wert 2 (wie die lokale Variable)
```

Besonderheiten und Beschränkungen lokaler Variablen:

- Lokale Variablen werden von den den Funktionen zur Abfrage der Variablenliste (VarGetInit, VarGetInit2, VarExist?) ignoriert.
- Sie können nicht mit Panel- oder Dialogelementen verknüpft werden.
- Sie sind für Abfragen per DDE nicht sichtbar.
- Sie werden in der Variablenliste/Messungsansicht nicht angezeigt.
- Sie werden bei Projekten nicht mit gespeichert.
- Die Deklaration einer lokalen Variablen ist nur im Kontext einer Sequenz sinnvoll, im 'Eingabe'-Fenster wird der LOCAL-Befehl daher ignoriert.
- Die Vorwärts-Deklaration gilt nur für Variablen, die direkt durch eine Zuweisung entstehen. Variablen, die durch Befehle (z.B. LOAD, REQUEST etc) entstehen, werden niemals lokal.

Siehe auch:

log

Logarithmus zur Basis 10.
Alternativer Name: **lg**

Deklaration:

```
log ( Daten ) -> Ergebnis
```

Parameter:

Daten	Daten. Erlaubte Typen: [ND],[XY].
Ergebnis	Zehnerlogarithmus des Parameters.

Beschreibung:

Der Zehnerlogarithmus wird gebildet.

Anmerkungen

- Die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Der Parameter der Funktion sollte keine Einheit haben. Hat er jedoch eine, bleibt sie erhalten (Warnung wird erzeugt).
- Der Parameter der Funktion log() sollte stets größer Null sein.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

```
zero = log(1)
```

Es ist z.B. log(0.1)= -1, log(1)= 0, log(10)= 1

Siehe auch:

ln, dB, ^(hoch)

LogSetup

Protokoll-Ausgabe konfigurieren

Deklaration:

```
LogSetup ( TxZiel [, EwReset] [, TxZeitStempel] [, TxOutputLevel] )
```

Parameter:

TxZiel	Bestimmt das Ziel für nachfolgende LogTrace()-Ausgaben. Wenn ein leerer String angegeben wird, werden die Nachrichten in das Ausgabefenster im FAMOS-Hauptfenster geschrieben. Ansonsten wird hier ein Dateiname erwartet.
EwReset	Anzeige/Datei zurücksetzen (optional , Standardwert: 0)
	0 : Nicht löschen
	1 : Ausgabefenster löschen bzw. Datei leeren.
TxZeitStempel	Nachrichten mit führendem Zeitstempel ausgeben. (optional , Standardwert: "")
	"" : Kein Zeitstempel
	"hh:mm:ss" : Uhrzeit (Genauigkeit: 1 Sekunde)
	"hh:mm" : Uhrzeit (Genauigkeit: 1 Minute)
	"hh:mm:ss,0" : Uhrzeit (Genauigkeit: 1/10 Sekunde)
	"d hh:mm:ss" : Datum + Uhrzeit (Genauigkeit: 1 Sekunde)
	"d hh:mm" : Datum + Uhrzeit (Genauigkeit: 1 Minute)
	"d hh:mm:ss,0" : Datum + Uhrzeit (Genauigkeit: 1/10 Sekunde)
TxOutputLevel	LogTrace()-Befehle, die mit einer kleinerer Wichtigkeit konfiguriert sind, werden ignoriert. (optional , Standardwert: "*")
	"-" : Alle LogTrace()-Ausgaben ignorieren.
	"e" : Nur wichtige Fehler ausgeben.
	"e2" : Alle Fehler (e, e2) ausgeben.
	"w" : Alle Fehler und wichtige Warnungen (e, e2, w) ausgeben
	"w2" : Alle Fehler und alle Warnungen (e, e2, w, w2) ausgeben.
	"i" : Alle Fehler, alle Warnungen und wichtige Informationen (e, e2, w, w2, i) ausgeben.
	"*" : Alles ausgeben.

Beschreibung:

Wenn für [TxZiel] ein Dateiname ohne Verzeichnis angegeben ist, wird das Standard-Sequenzverzeichnis verwendet. Wenn keine Dateierweiterung angegeben ist, wird ".log" verwendet.

Die Ausgabe-Datei wird als Textdatei im UTF-8-Zeichensatz geschrieben.

Bei Ausgabe in eine Datei werden Fehlermeldungen und Warnungen, die durch FAMOS vom Formelinterpreter oder bei der Ausführung von Funktionen generiert werden, ebenfalls protokolliert.

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Beispiele:

Eine Sequenz lädt alle Messwert-Dateien mit einem vorgegebenen Namensmuster aus einem Verzeichnis. Für jede Datei wird ein Protokolleintrag generiert und an die Protokolldatei angehängt. Jeder Eintrag besteht aus Zeitstempel, Dateiname und Status (Erfolg/Fehler) der Operation.

```
logLevel = "*"
LogSetup("c:\log\protocol.txt", 0, "d hh:mm:ss,0", logLevel)
SetOption("Func.ErrorBoxes", "No")  ; Keine Fehlerboxen bei Dateifunktionen!
fileNames = FsGetFileNames("c:\imc\dat","a*.dat", 0, 0, 0)
FOREACH ELEMENT name in fileNames
   IF FileLoad(name, "", 0) >= 0
      LogTrace("Load file " + name + " => Success")
   ELSE
      LogTrace("Load file " + name + " => Failure", "e")
   END
END
```

Hinweis: Um die Protokollierung zentral komplett auszuschalten, setzen Sie die Variable [logLevel] auf "-". Um nur die Fehlermeldungen auszugeben, setzen Sie [logLevel] auf "e".

Die im vorherigen Beispiel erzeugte Protokoll-Datei wird später wieder geladen und im Ausgabefenster angezeigt:

```
LogSetup("", 1)
LogTrace("--- Inhalt der Logdatei ---")
idFile = FileOpenASCII("c:\log\protocol.txt", 0)
TxZeile = ""
ok = FileLineRead(idFile, TxZeile, 0)
WHILE ok = 0
   LogTrace(TxZeile)
   ok = FileLineRead(idFile, TxZeile, 0)
END
FileClose(idFile)
```

Siehe auch:

LogTrace, BoxOutput

LogTrace

Ausgabe eines Textes in das Ausgabefenster oder eine Datei.

Deklaration:

```
LogTrace ( TxLogText [, TxLevel] )
```

Parameter:

TxLogText	Auszugebender Text
TxLevel	Beschreibt die Wichtigkeit der Ausgabe, wird gegen das in LogSetup() angegebene Ausgabelevel geprüft - falls [TxLevel] kleiner (weniger wichtig) ist, wird der Befehl ignoriert. (optional , Standardwert: "i2")
	"e" : Fehler
	"e2" : Fehler niederer Priorität
	"w" : Warnung
	"w2" : Warnung niederer Priorität
	"i" : Information
	"i2" : Information niederer Priorität

Beschreibung:

Die Funktion schreibt den angegebenen Text in das Protokoll. Ausgabemedium (Text-Datei oder Ausgabefenster) und -Format (z.B. führender Zeitstempel) wird mit der Funktion LogSetup() konfiguriert.

Bei Ausgabe in eine Datei wird dem Text abhängig von [TxLevel] ein "(E)", "(W)" oder "(I)" vorangestellt. "E" steht dabei für Fehler (Error), "W" für Warnung, "I" für Information. Bei Protokollierung im Ausgabefenster wird das Level durch ein entsprechendes Symbol gekennzeichnet.

Beispiele:

Eine Sequenz lädt alle Messwert-Dateien mit einem vorgegebenen Namensmuster aus einem Verzeichnis. Für jede Datei wird ein Protokolleintrag generiert und an die Protokolldatei angehängt. Jeder Eintrag besteht aus Zeitstempel, Dateiname und Status (Erfolg/Fehler) der Operation.

```
logLevel = "*"
LogSetup("c:\log\protocol.txt", 0, "d hh:mm:ss,0", logLevel)
SetOption("Func.ErrorBoxes", "No")  ; Keine Fehlerboxen bei Dateifunktionen!
fileNames = FsGetFileNames("c:\imc\dat","a*.dat", 0, 0, 0)
FOREACH ELEMENT name in fileNames
   IF FileLoad(name, "", 0) >= 0
      LogTrace("Load file " + name + " => Success")
   ELSE
      LogTrace("Load file " + name + " => Failure", "e")
   END
END
```

Hinweis: Um die Protokollierung zentral komplett auszuschalten, setzen Sie die Variable [logLevel] auf "-". Um nur die Fehlermeldungen auszugeben, setzen Sie [logLevel] auf "e".

Die im vorherigen Beispiel erzeugte Protokoll-Datei wird später wieder geladen und im Ausgabefenster angezeigt:

```
LogSetup("", 1)
LogTrace("--- Inhalt der Logdatei ---")
idFile = FileOpenASCII("c:\log\protocol.txt", 0)
TxZeile = ""
status = FileLineRead(idFile, TxZeile, 0)
WHILE status = 0
   LogTrace(TxZeile)
   status = FileLineRead(idFile, TxZeile, 0)
END
FileClose(idFile)
```

Siehe auch:

LogSetup, BoxOutput, FileLineWrite

LostValueFill

Verfügbar ab: Professional Edition

Füllen der Lücken zwischen den Events, wobei die Events entstanden sind durch vorhandene Lost Value, Overflow, Not a Number, Übersteuerung, Fühlerbruch.

Deklaration:

```
LostValueFill ( Eingangsdaten [, Referenz] [, WieErsetzen] [, Ersatzwert] [, Zeit_Option] [, MaxLen] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Eingangsdaten
Referenz	Referenz. Falls ein leerer Text "", dann gibt es keine Referenz. Die Berechnung des Starts und Endes und der Länge der Lücken erfolgt allein durch die x0 und die Triggerzeiten. Bei vorhandenem Referenzkanal werden Start und Ende durch den Referenzkanal festgelegt. Damit kann der eigentliche Start vor dem Beginn des ersten Events liegen und das Ende hinter dem Ende des letzten Events. Der Referenzkanal muss dieselbe Abtastzeit aufweisen wie die Eingangsdaten. (optional , Standardwert: "")
WieErsetzen	Wie soll ersetzt werden? (optional , Standardwert: 0)
	0 : linear interpoliert: zwischen dem letzten gültigen vor der Lücke und dem ersten gültigen danach.
	1 : konstant fortgesetzt
	2 : konstant nächster: Der am nächsten liegende gültige Randwert wird benutzt.
	3 : fester Wert, der als folgender Parameter Ersatzwert angegeben wird
Ersatzwert	Der Ersatzwert, falls mit einem fest definierten Wert ersetzt werden soll. Sonst 0. (optional , Standardwert: 0)
Zeit_Option	Zeit_Option (optional , Standardwert: 0)
	0 : Die Triggerzeiten der Events werden nicht berücksichtigt.
	1 : Die Triggerzeiten der Events werden berücksichtigt.
MaxLen	Maximale Länge des Ergebnisses in Samples; = 0, falls keine Begrenzung. Eine Begrenzung ist sinnvoll, wenn die Eingangsdaten möglicherweise nicht aus einer sinnvollen Messung stammen. Dann könnten z.B. durch ungünstige Werte der Triggerzeiten oder auch x-Offsets (x0) der Events riesige Datenmengen entstehen. Droht das Ergebnis länger zu werden als die hier definierte Länge, bricht die Funktion mit einer Fehlermeldung ab. (optional , Standardwert: 0)
Ergebnis	Ergebnis

Beschreibung:

Lost Values in einem Datensatz können dazu geführt haben, dass der Datensatz in mehrere Events aufgeteilt wurde. Dabei enthalten die einzelnen Events keine Lost Values. Die Lost Values kann man sich in den Lücken zwischen den Events denken.

Für einige Auswertungen kann es von Vorteil sein, wenn alle Events zeitrichtig wieder zu einem Datensatz ohne Events aneinandergehängt werden. Dabei werden die Lücken zwischen den Events durch neu eingefügte Werte ersetzt.

Lost Value

Der Lost Value bezeichnet einen verloren gegangenen oder ungültigen oder "eigentlich nicht vorhandenen" Wert in einem Datensatz. Der Messwert kann z.B. durch den Überlauf (Overflow) eines Pufferspeichers oder bei fehlerhafter Übertragung über ein Bus-System verloren gegangen sein. Der korrekte Messwert kann z.B. wegen eines Fühlerbruchs (abgebrochenes Thermoelement) oder der Übersteuerung eines Messverstärkers nicht bestimmbar sein. Wenn ein äquidistant abgetasteter Messkanal einen Wert benötigt, aber kein Wert vorliegt, wird stattdessen ein Lost Value verwendet.

Bei reellen Zahlenformaten ist ein Lost Value häufig als 1e100 dargestellt; oder als Not a Number, ebenfalls als 1e100 dargestellt. Bei ganzzahligen Zahlenformaten wird meist ein Wert am Ende des entsprechenden Zahlenbereichs gewählt. So z.B. -32768 bei 16 Bit ganzen Zahlen mit Vorzeichen.

Bei der Übersteuerung von Messverstärkern entstehen meist Messwerte, die (leicht) außerhalb des Messbereichs liegen. Sie sind typisch durch Hardware oder Firmware begrenzt worden.

Bei Fühlerbruch eines Thermoelements wird von vielen Messgeräten ein Ersatzwert von etwa -2000 in den Temperaturkanal geschrieben. Eine Prüfung auf <=-1000 würde diese Fühlerbruchwerte erkennen.

Not a Number (NaN) ist der binäre Code, den der Computer beim Datentyp reelle Zahl benutzt, wenn der vorliegende Wert keine Zahl ist oder für ihn kein Zahlenwert vorhanden ist. Der Wert wird in imc FAMOS meist als 1e100 dargestellt. Not a Number kann benutzt werden, um einen Lost Value darzustellen. Not a Number kann auch durch arithmetische Operationen entstehen, wenn der zulässige Zahlenbereich überschritten wird.

Parameter

Die Eingangsdaten sind äquidistant und sollten Events haben. Eingangsdaten ohne Events werden wie Daten mit einem Event behandelt.

Die WieErsetzen Option und der Ersatzwert legen fest, wie ersetzt wird.

Beispiele:

Eine Prüfstandssoftware erzeugte bei einem eigentlich kontinuierlichen Datenstrom mehrere Events, nachdem aufgrund von (einigen wenigen) Netzwerkstörungen Daten verloren gingen. Alle Events erhalten dieselbe Triggerzeit, die Verschiebungen werden durch x0 des Events ausgedrückt. Die auswertende Software benötigt einen einzigen zusammenhängenden Datensatz ohne Events. Die Gesamtmessung lässt ca. 2e6 Messwerte erwarten.

```
T = LostValueFill( T_Evn, "", 0, 0, 0, 1e7)
```

Bei einem Dauerversuch gibt es die beiden Temperaturkanäle T1 und T2, die gleich abgetastet sind während der gesamten Messung. Aber T2 wurde wegen Overflow in mehrere Events aufgeteilt. Für die folgende Auswertung müssen beide Kanäle dieselbe zeitliche Ausdehnung haben.

```
T2_NoEvn = LostValueFill( T2, T1, 0, 0, 0, 1e7)
Delta = T2_NoEvn - T1
```

Siehe auch:

LostValueReplace, LostValueGaps, PTastEx

LostValueGaps

Verfügbar ab: Professional Edition

Vorhandene Lost Value, Overflow, Not a Number, Übersteuerung, Fühlerbruch werden durch Lücken ersetzt. Lücken werden über Events oder XY-Format realisiert.

Deklaration:

```
LostValueGaps ( Eingangsdaten, ErgebnisFormat, WieFinden [, Limit] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Eingangsdaten
ErgebnisFormat	Format des Ergebnisses: Sollen die Lücken durch Events oder über XY-Werte realisiert werden.
	0 : Events werden erzeugt. Jedes Event enthält eine kompakte Folge von Werten ohne Lost Values.
	1 : Der Datentyp XY wird erzeugt. Wertepaare mit Lost Value erscheinen nicht im Ergebnis.
WieFinden	Bedingung, die angibt, wie Lost Values im Datensatz gefunden werden
	0 : =NaN (Not a Number)
	1 : >=Limit
	2 : <=Limit
	3 : >Limit
	4 : <Limit
	5 : >=Limit oder <=-Limit
	6 : <Limit und >-Limit
Limit	Grenze, die in der Bedingung WieFinden auftaucht. (optional , Standardwert: 1e100)
Ergebnis	Ergebnis

Beschreibung:

Wenn in einem Datensatz Lost Values auftauchen, ist bei Auswertungen und Darstellungen darauf zu reagieren. Denn die Lost Values selbst enthalten eine wichtige Information. Andererseits fehlen an diesen Stellen die richtigen Werte.

Es gibt Situationen, in denen die Lost Values weggeschnitten werden sollen und einfach Lücken in den Daten bleiben. Die Anwendung deutet die Lücken entsprechend.

Die Lücken können realisiert werden, indem ein XY-Format im Ergebnis erzwungen wird. Die XY-Paare, die einen Lost Value enthalten, erscheinen nicht.

Das XY-Format kann sinnvoll erzeugt werden, egal wie häufig oder unregelmäßig auch immer LostValues verteilt sind. Eine Darstellung der Messpunkte im Kurvenfenster zeigt stets deutlich, an welchen Stellen wirkliche Messwerte vorliegen.

Alternativ können die Lücken realisiert werden, indem die Abschnitte ohne LostValues in Events eingeteilt werden. Ein Event enthält dann stets eine kompakte Folge von gültigen Werten. Die Lost Values wären dann in den Lücken zwischen den Events zu denken.

Das Erzeugen von Events ist nur sinnvoll, wenn es wenige Stellen gibt, an denen kompakte Häufungen von LostValues auftreten. Umgekehrt gesehen gibt es dann nur wenige kompakte Bereiche, in denen keine Lost Values auftreten. Es sollte aus Gründen der Effektivität vermieden werden, unnötig viele und kurze Events zu erzeugen.

Eine andere Technik im Umgang mit Lost Values ist mittels der Funktion LostValueReplace() realisiert. Die Anwendung entscheidet über die sinnvolle Vorgehensweise.

Lost Value

Der Lost Value bezeichnet einen verloren gegangenen oder ungültigen oder "eigentlich nicht vorhandenen" Wert in einem Datensatz. Der Messwert kann z.B. durch den Überlauf (Overflow) eines Pufferspeichers oder bei fehlerhafter Übertragung über ein Bus-System verloren gegangen sein. Der korrekte Messwert kann z.B. wegen eines Fühlerbruchs (abgebrochenes Thermoelement) oder der Übersteuerung eines Messverstärkers nicht bestimmbar sein. Wenn ein äquidistant abgetasteter Messkanal einen Wert benötigt, aber kein Wert vorliegt, wird stattdessen ein Lost Value verwendet.

Bei reellen Zahlenformaten ist ein Lost Value häufig als 1e100 dargestellt; oder als Not a Number, ebenfalls als 1e100 dargestellt. Bei ganzzahligen Zahlenformaten wird meist ein Wert am Ende des entsprechenden Zahlenbereichs gewählt. So z.B. -32768 bei 16 Bit ganzen Zahlen mit Vorzeichen.

Bei der Übersteuerung von Messverstärkern entstehen meist Messwerte, die (leicht) außerhalb des Messbereichs liegen. Sie sind typisch durch Hardware oder Firmware begrenzt worden.

Bei Fühlerbruch eines Thermoelements wird von vielen Messgeräten ein Ersatzwert von etwa -2000 in den Temperaturkanal geschrieben. Eine Prüfung auf <=-1000 würde diese Fühlerbruchwerte erkennen.

Not a Number (NaN) ist der binäre Code, den der Computer beim Datentyp reelle Zahl benutzt, wenn der vorliegende Wert keine Zahl ist oder für ihn kein Zahlenwert vorhanden ist. Der Wert wird in imc FAMOS meist als 1e100 dargestellt. Not a Number kann benutzt werden, um einen

Lost Value darzustellen. Not a Number kann auch durch arithmetische Operationen entstehen, wenn der zulässige Zahlenbereich überschritten wird.

Parameter

Die Eingangsdaten können äquidistant sein oder im XY-Format mit monoton wachsender x-Achse vorliegen. Events sind möglich, Segmente nicht.

Die WieFinden Bedingung und der Vergleichswert Limit legen fest, welche Werte zu ersetzen sind.

Beispiele:

Ein mit 10Hz abgetasteter Temperaturverlauf, der bei Fühlerbruch den Wert -2000 enthält, wird für eine folgende Auswertung korrigiert: Ein XY-Format wird erzeugt, wobei alle Wertepaare mit Fühlerbruch nicht auftauchen.

```
T_xy = LostValueGaps(T, 0, 4, -1000)
```

Die Drehzahl wurde an einem Prüfstand im Langzeitversuche gemessen. Mehrmals kam es dabei zu Netzwerkstörungen, so dass für mehrere Minuten keine Messwerte übertragen wurden und Puffer übergelaufen sind. Die speichernde Software fügte an diesen Stellen NaN (Not a Number) Werte in die Messkanäle ein. Ein Datensatz mit mehreren Events soll daraus erzeugt werden, wobei alle NaN in den Lücken zwischen den Events liegen sollen.

```
Speed_Evn = LostValueGaps(Speed, 1, 0, 0)
```

Siehe auch:

LostValueReplace, LostValueFill

LostValueReplace

Verfügbar ab: Professional Edition

Vorhandene Lost Value, Overflow, Not a Number, Übersteuerung, Fühlerbruch werden im Datensatz ersetzt.

Deklaration:

```
LostValueReplace ( Eingangsdaten, WieFinden [, Limit] [, WieErsetzen] [, Ersatzwert] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Eingangsdaten
WieFinden	Bedingung, die angibt, wie Lost Values im Datensatz gefunden werden
	0 : =NaN (Not a Number)
	1 : >=Limit
	2 : <=Limit
	3 : >Limit
	4 : <Limit
	5 : >=Limit oder <=-Limit
	6 : <Limit und >-Limit
Limit	Grenze, die in der Bedingung WieFinden auftaucht. (optional , Standardwert: 1e100)
WieErsetzen	Wie soll ersetzt werden? (optional , Standardwert: 0)
	0 : linear interpoliert: zwischen dem letzten gültigen vor den Lost Values und dem ersten gültigen danach. Falls am Rand, wird konstant fortgesetzt.
	1 : konstant fortgesetzt
	2 : konstant nächster: Der am nächsten liegende gültige Randwert wird benutzt.
	3 : fester Wert, der als folgender Parameter Ersatzwert angegeben wird
Ersatzwert	Der Ersatzwert, falls mit einem fest definierten Wert ersetzt werden soll. Sonst 0. (optional , Standardwert: 0)
Ergebnis	Ergebnis

Beschreibung:

Wenn in einem Datensatz Lost Values auftauchen, ist bei Auswertungen und Darstellungen darauf zu reagieren. Denn die Lost Values selbst enthalten eine wichtige Information. Andererseits fehlen an diesen Stellen die richtigen Werte.

Es gibt Situationen, in denen wenige Lost Values auftauchen und der Datensatz ausgewertet und dargestellt werden soll, ohne dass die Lost Values stören. Dazu sollen die Lost Values ersetzt werden durch gültige Messwerte aus ihrer Umgebung. Diese Technik kann zulässig sein, wenn sich das Signal langsam ändert. Damit wird dann ein geschätzter Verlauf des Signals ermittelt. Das resultierende Ergebnis enthält also richtige Werte und teilweise geschätzte.

Eine andere Technik im Umgang mit Lost Values ist mittels der Funktion LostValueGaps() realisiert. Die Anwendung entscheidet über die sinnvolle Vorgehensweise.

Lost Value

Der Lost Value bezeichnet einen verloren gegangenen oder ungültigen oder "eigentlich nicht vorhandenen" Wert in einem Datensatz. Der Messwert kann z.B. durch den Überlauf (Overflow) eines Pufferspeichers oder bei fehlerhafter Übertragung über ein Bus-System verloren gegangen sein. Der korrekte Messwert kann z.B. wegen eines Fühlerbruchs (abgebrochenes Thermoelement) oder der Übersteuerung eines Messverstärkers nicht bestimmbar sein. Wenn ein äquidistant abgetasteter Messkanal einen Wert benötigt, aber kein Wert vorliegt, wird stattdessen ein Lost Value verwendet.

Bei reellen Zahlenformaten ist ein Lost Value häufig als 1e100 dargestellt; oder als Not a Number, ebenfalls als 1e100 dargestellt. Bei ganzzahligen Zahlenformaten wird meist ein Wert am Ende des entsprechenden Zahlenbereichs gewählt. So z.B. -32768 bei 16 Bit ganzen Zahlen mit Vorzeichen.

Bei der Übersteuerung von Messverstärkern entstehen meist Messwerte, die (leicht) außerhalb des Messbereichs liegen. Sie sind typisch durch Hardware oder Firmware begrenzt worden.

Bei Fühlerbruch eines Thermoelements wird von vielen Messgeräten ein Ersatzwert von etwa -2000 in den Temperaturkanal geschrieben. Eine Prüfung auf <=-1000 würde diese Fühlerbruchwerte erkennen.

Not a Number (NaN) ist der binäre Code, den der Computer beim Datentyp reelle Zahl benutzt, wenn der vorliegende Wert keine Zahl ist oder für ihn kein Zahlenwert vorhanden ist. Der Wert wird in imc FAMOS meist als 1e100 dargestellt. Not a Number kann benutzt werden, um einen Lost Value darzustellen. Not a Number kann auch durch arithmetische Operationen entstehen, wenn der zulässige Zahlenbereich überschritten wird.

Parameter

Die Eingangsdaten können äquidistant sein oder im XY Format vorliegen. Segmente und Events sind möglich.

Die WieFinden Bedingung und der Vergleichswert Limit legen fest, welche Werte zu ersetzen sind.

Die WieErsetzen Option und der Ersatzwert legen fest, wie ersetzt wird.

Bei WieErsetzen gleich konstant wird bei Lücke vor dem ersten Wert der erste Wert nach vorn verlängert.

Bei XY Eingangsdaten nur konstante Fortsetzung und fester Wert möglich.

Für einen nicht benutzten Ersatzwert muss 0 angegeben werden.

Das Fortsetzen von Werten bzw. auch das Interpolieren geschieht nur innerhalb eines Events bzw. Segments, nie über die Zwischenräume hinweg.

Beispiele:

Ein mit 10Hz abgetasteter Temperaturverlauf, der bei Fühlerbruch den Wert -2000 enthält, wird für eine folgende Auswertung korrigiert: Die Fühlerbruch-Werte werden durch lineare Interpolation der umliegenden gültigen Werte ersetzt.

```
T = LostValueReplace(T_orig, 4, -1000, 0)
```

Aus dem Import einer ASCII-Datei stammt der Drehzahlkanal Rev, der NaN (Not a Number) Werte enthält. Diese stören die Optik beim Darstellen des Kanals. Da der Kanal keine negativen Werte enthält, wird -1 als Ersatzwert benutzt.

```
Rev_ = LostValueReplace(Rev, 0, 0, 3, -1)
```

Ein Messkanal liegt vor, der den Gang eines Automatik-Getriebes darstellt und über den CAN-Bus übertragen wurde. Die Firmware des loggenden Messgerätes findet eine -3 vor, falls der Gang nicht bestimmt werden konnte, eine -4, falls die Übertragung am CAN-Bus fehlerhaft war. Für eine Auswertung wird stets ein gültiger Gang gebraucht. Die Fehlersituationen sind selten. Der zum Ausfallzeitpunkt am nächsten liegende gültige Wert soll als Ersatz benutzt werden.

```
Gear = LostValueReplace(Gear_CAN, 2, -3, 2)
```

Siehe auch:

RangeSet, Setze, LostValueGaps, LostValueFill

LoudnessLevel

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Aus einem Terzspektrum wird die Lautheit bzw. der Lautstärkepegel nach DIN 45631/A1:2010-03 bzw. ISO 532-1:2017 berechnet. Das Verfahren nach E. Zwicker wird benutzt.

Deklaration:

```
LoudnessLevel ( TerzSpektrumDB, Schallfeld, Einheit ) -> Ergebnis
```

Parameter:

TerzSpektrumDB	Die Eingangsdaten. Ein Terzspektrum, skaliert in dB. Wertebereich -60..+120dB. Was außerhalb ist, wird auf diesen Bereich beschränkt. Die Terzen von 25Hz bis 12.5kHz sollten enthalten sein. Was nicht enthalten ist, wird als Minimum des Wertebereichs angenommen. Der Datensatz ist in x-Richtung in Terzen skaliert, also: x=14 entspricht 25Hz, x=15 entspricht 31.5Hz, ... x=41 entspricht 12.5kHz. Falls dieser Datensatz segmentiert ist, wird eine Folge von Werten (Datensatz)zurückgegeben.
Schallfeld	Schallfeld
	2 : frei, Freifeld, Aufnahme im Freien
	3 : diffus, Aufnahme z.B. in einem (kleinen) Raum wie auch Fahrzeuginnenraum
Einheit	In welcher Einheit soll das Ergebnis angegeben werden?
	0 : phon (Der Lautstärkepegel LN wird bestimmt)
	1 : sone (Die Lautheit N wird bestimmt)
Ergebnis	Das Ergebnis erhält als Einheit entsprechend Schallfeld und Typ die Einheit "phonGF", "phonGD", "soneGF", "soneGD".

Beschreibung:

Der Algorithmus behandelt stationäre Geräusche, keine zeitvarianten.

Nicht mehr zu benutzende Parameter, nur noch kompatibel zu imc FAMOS 7.2 und früher:
Schallfeld=0: eben, frontal

Schallfeld=1: diffus

Der wesentliche Unterschied ist eine unwesentliche Rundung, die die DIN 45631:1991 im Anhang A nahelegte und in den neuen Normen nicht mehr erscheint.

Beispiele:

Die Lautheit bzw. der Lautstärkepegel sollen aus einem Terzspektrum ThirdsDB bestimmt werden:

```
phonGF = LoudnessLevel(ThirdsDB,2,0) ; phon, Freifeld
soneGF = LoudnessLevel(ThirdsDB,2,1) ; sone, Freifeld
phonGD = LoudnessLevel(terzen,3,0) ; phon, diffus
soneGD = LoudnessLevel(terzen,3,1) ; sone, diffus
```

Aus einem Schalldruckverlauf soll die Lautheit bestimmt werden (1 Zahlenwert für die gesamte Messung).

Beispiel-Datensatz Einzelton 1kHz, 70dB. Diese Zeilen können benutzt werden, falls keine Beispieldaten vorliegen:

```
Microphone= 0.08944*sin ( ramp(0,1/40000, 50000)*PI2 * 1000 )
yeinheit Microphone Pa
xeinheit Microphone s
```

Terzanalyse

```
OctI(0, 0, 0, -2, 0, 0, 0)
ThirdsTotal = OctA(Microphone,25,12500)
```

Das Einschwingen soll übersprungen werden. Die Funktion OctA bestimmt den Effektivwert über alles, auch während des Einschwingens. Falls das Einschwingen keine Rolle spielt, kann Thirds = ThirdsTotal stattdessen ausgeführt werden.

```
Settle = 0.5 's' / xdel?(Microphone) ; Annahme: 0.5s Einschwingdauer der Terzfilter
Length = leng? ( Microphone)
Wenn Length > Settle + 0.1
   ThirdsSettle = OctA(leng( Microphone, Settle ),25,12500)
   Thirds = wurz ( ( Length * quad ( ThirdsTotal ) - Settle * quad ( ThirdsSettle ) ) / ( length - Settle ) )
   entf ThirdsSettle
sonst
   Thirds = ThirdsTotal
ende
entf ThirdsTotal
```

```
entf Length
entf Settle
ThirdsDB = db ( Thirds / 2e-5'Pa' ) ; Das liefert Schalldruckpegel
```

Lautheitsbestimmung

```
soneGF = LoudnessLevel ( ThirdsDB, 2, 1 )
```

Aus einem Verlauf des Terzspektrums über der Zeit soll ein Zeitverlauf der Lautheit ermittelt werden.

Terzanalyse in Abhängigkeit von der Zeit. Schalldruck liegt im Signal Microphone vor, skaliert in Pa über s.

```
Interval = 0.1's' ; Das ist das Zeitintervall
OctI(0, 0.125, Interval/xdel?(Microphone), -2, 0, 0, 0)
Thirds = OctA(Microphone,25,12500)
ThirdsDB = db ( Thirds / 2e-5'Pa' )
SetSegLen( ThirdsDB, 28 ) ; segmentieren
SetZoff(ThirdsDB,xoff? ( Microphone)) ; die z-Koordinate definieren
SetZDel(ThirdsDB,Interval)
SetUnit(ThirdsDB,Unit?(Microphone,0),2)
```

Lautheitsbestimmung

```
soneGF = LoudnessLevel ( ThirdsDB, 2, 1)
```

Siehe auch:

LoudnessSpectrum, SoundIndex

LoudnessSpectrum

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Aus einem Terzspektrum wird das Lautheits-Tonheits-Muster nach DIN 45631/A1:2010-03 bzw. ISO 532-1:2017 berechnet. Dabei ist die spezifische Lautheit N' als Funktion der Tonheit z in Bark dargestellt. Das Verfahren nach E. Zwicker wird benutzt.

Deklaration:

```
LoudnessSpectrum ( TerzSpektrumDB, Schallfeld, Typ ) -> Ergebnis
```

Parameter:

TerzSpektrumDB	Die Eingangsdaten. Ein Terzspektrum, skaliert in dB. Wertebereich -60..+120dB. Was außerhalb ist, wird auf diesen Bereich beschränkt. Die Terzen von 25Hz bis 12.5kHz sollten enthalten sein. Was nicht enthalten ist, wird als Minimum des Wertebereichs angenommen. Der Datensatz ist in x-Richtung in Terzen skaliert, also: x=14 entspricht 25Hz, x=15 entspricht 31.5Hz, ... x=41 entspricht 12.5kHz.Falls dieser Datensatz segmentiert ist, wird eine Folge von Spektren ebenfalls als segmentierter Datensatz zurückgegeben.
Schallfeld	Schallfeld
	2 : frei, Freifeld, Aufnahme im Freien
	3 : diffus, Aufnahme z.B. in einem (kleinen) Raum wie auch Fahrzeuginnenraum
Typ	Typ
	0 : Flankenlautheit, das führt auf die spezifische Lautheit.
	1 : Kernlautheit. Nur in Sonderfällen zur Analyse von Verdeckungseffekten
Ergebnis	Das Ergebnis ist in sone/Bark über Bark skaliert.

Beschreibung:

Das Ergebnis liegt im Bereich 0 bis 24 Bark mit einer Auflösung von 0.1 Bark vor.

Der Algorithmus behandelt stationäre Geräusche, keine zeitvarianten.

Nicht mehr zu benutzende Parameter, nur noch kompatibel zu imc FAMOS 7.2 und früher:

Schallfeld=0: eben, frontal

Schallfeld=1: diffus

Der wesentliche Unterschied ist eine Verschiebung des Ergebnisses um 1 Wert, früher also 241 Werte, nun 240 Werte.

Beispiele:

Spezifische Lautheit aus einem Mikrofonsignal mic, gemessen in Pa über der Zeit. Aufnahme im Freien

```
Thd = SpecThirds_1( mic, 25, 12500, 0)
Thd_dB = dB ( Thd / 2e-5)
N = LoudnessSpectrum( Thd_dB, 2, 0 )
```

Die Kern- bzw. Flankenlautheiten sollen aus einem Terzspektrum ThirdsDB bestimmt werden:

```
N_KernF = LoudnessSpectrum( ThirdsDB,2,1)     ; Kernlautheit, Freifeld
N_FlankenF = LoudnessSpectrum( ThirdsDB,2,0)  ; Flankenlautheit, Freifeld
N_KernD = LoudnessSpectrum( ThirdsDB,3,1)     ; Kernlautheit, diffus
N_FlankenD = LoudnessSpectrum( ThirdsDB,3,0)  ; Flankenlautheit, diffus
```

Aus einem Schalldruckverlauf soll die Flankenlautheit bestimmt werden (1 Spektrum für die gesamte Messung). Für Beispieldaten und das Eliminieren der Einschwing-Effekte siehe Beispiel in LoudnessLevel()

Terzanalyse

```
OctI(0, 0, 0, -2, 0, 0, 0)
Thirds = OctA(Microphone,25,12500)
ThirdsDB = db ( Thirds / 2e-5'Pa' ) ; Das liefert Schalldruckpegel
```

Flankenlautheitsbestimmung

```
N_FlankenF = LoudnessSpectrum( ThirdsDB,2,0)
```

Aus einem Verlauf des Terzspektrums über der Zeit soll ein Zeitverlauf des Lautheits-Tonheits-Muster ermittelt werden.

Terzanalyse in Abhängigkeit von der Zeit. Schalldruck liegt im Signal Microphone vor, skaliert in Pa über s.

```
Interval = 0.1's' ; Das ist das Zeitintervall
OctI(0, 0.125, Interval/xdel?(Microphone), -2, 0, 0, 0)
Thirds = OctA(Microphone,25,12500)
ThirdsDB = db ( Thirds / 2e-5'Pa' )
```

```
SetSegLen( ThirdsDB, 28 ) ; segmentieren
SetZoff(ThirdsDB,xoff? ( Microphone)) ; die z-Koordinate definieren
SetZDel(ThirdsDB,Interval)
SetUnit(ThirdsDB,Unit?(Microphone,0),2)
```

Flankenlautheitsbestimmung

```
N_FlankenF = LoudnessSpectrum( ThirdsDB,2,0)
N_FlankenF_T = MatrixTranspose( N_FlankenF )
```

Siehe auch:

LoudnessLevel, OctI, SpecThirds, Sharpness

LowerValue

Liefert den jeweils kleineren Wert der beiden Parameter.

Deklaration:

```
LowerValue ( Parameter1, Parameter2 ) -> Ergebnis
```

Parameter:

Parameter1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Parameter2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
Ergebnis	Der jeweils kleinere Wert der beiden Parameter.

Beschreibung:

Die Funktion hat zwei praktische Anwendungen. Bei der Parameterkombination Datensatz/Einzelwert findet eine obere Begrenzung des Datensatzes auf den Wert des 2. Parameters statt. Falls beide Parameter Datensätze sind, ist das Ergebnis die untere Hüllkurve.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Der Eingangskanal wird in Dezibel umgerechnet und auf eine obere Grenze von 100 dB begrenzt.

```
Channel_01_dB = LowerValue(db(Channel_01), 100)
```

Die untere und obere Einhüllende zweier Datensätze wird bestimmt.

```
Env_L = LowerValue(Channel_01, Channel_02)
Env_H = UpperValue(Channel_01, Channel_02)
```

Siehe auch:

UpperValue, <, RangeSet

MatrixAdd

Verfügbar ab: Professional Edition

Addition und Subtraktion zweier Matrizen oder Vektoren, auch ihrer Transponierten

Deklaration:

```
MatrixAdd ( Matrix A, Matrix B [, Berechnung] ) -> Ergebnis
```

Parameter:

Matrix A	Matrix bzw. Vektor A
Matrix B	Matrix bzw. Vektor B
Berechnung	Sollen die Matrizen vor der Verrechnung transponiert werden? (optional , Standardwert: "A+B")
	"A+B" : A + B
	"AT+B" : A transponiert + B
	"A+BT" : A + B transponiert
	"AT+BT" : A transponiert + B transponiert
	"A-B" : A - B
	"AT-B" : A transponiert - B
	"A-BT" : A - B transponiert
	"AT-BT" : A transponiert - B transponiert
Ergebnis	Ergebnis

Beschreibung:

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Als Vektor wird in diesem Zusammenhang eine einreihige Matrix verstanden, also eine Matrix mit nur einer Zeile oder mit nur einer Spalte.

Ein Spaltenvektor ist eine Matrix mit nur einer Spalte. Er ist ein Datensatz ohne Segmente.

Ein Spaltenvektor kann auch durch einen segmentierten Datensatz mit genau 1 Segment dargestellt werden.

Ein Zeilenvektor ist ein segmentierter Datensatz mit Segmentlänge 1.

Das Ergebnis der Funktion ist im allgemeinen Fall eine Matrix und damit ein segmentierter Datensatz. Ist das Ergebnis ein Spaltenvektor, ist es ein Datensatz ohne Segmente. Ist das Ergebnis eine einzige Zahl, wird sie als Spaltenvektor gedeutet und hat damit auch keine Segmente.

Subtrahieren mittels "A-B" etc. oder durch Addition der negativen Matrix.

Einheiten und Abtastzeiten werden nicht beachtet.

In vielen Fällen kann auch anstelle der Funktion das Zeichen "+" oder "-" zur Verrechnung eingesetzt werden.

Die Addition kann nur erfolgen, wenn nach ggf. durchgeführter Transponierung Zeilen- und Spaltenanzahl übereinstimmen.

Addieren und Subtrahieren von Zahlen für jedes Element der Matrix mit den Zeichen "+" und "-", z.B. Matrix+1.

Beispiele:

Summe und Differenz von Matrizen

```
A = ramp(1, 1, 6 ) ; test data
setsegLen( A, 2)
; A:
; 1 3 5
; 2 4 6
B = MatrixInit ( 2, 3, "I" )
; B:
; 1 0 0
; 0 1 0
C_sum = MatrixAdd( A, B, "A+B" )
; C_sum:
; 2 3 5
; 2 5 6
C_diff = MatrixAdd( A, B, "A-B" )
; C_diff:
; 0 3 5
; 2 3 6
C_diff = MatrixAdd( A, -B, "A+B" )
; C_diff:
```

```
; 0 3 5
; 2 3 6
```

Siehe auch:

MatrixMult

MatrixChangeDim

Verfügbar ab: Professional Edition

Einfügen und Löschen von Zeilen oder Spalten

Deklaration:

```
MatrixChangeDim ( Matrix, Start, Anzahl, Option ) -> Matrix
```

Parameter:

Matrix	Matrix
Start	Ab diesem Startindex, = 1 für das erste Element
Anzahl	Anzahl
Option	Wie soll geändert werden?
	"insert col" : Spalten einfügen
	"insert row" : Zeilen einfügen
	"remove col" : Spalten entfernen
	"remove row" : Zeilen entfernen
Matrix	Matrix

Beschreibung:

Das Ergebnis ist eine Matrix veränderter Dimension wie Matrix A.

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Als Vektor wird in diesem Zusammenhang eine einreihige Matrix verstanden, also eine Matrix mit nur einer Zeile oder mit nur einer Spalte.

Ein Spaltenvektor ist eine Matrix mit nur einer Spalte. Er ist ein Datensatz ohne Segmente.

Ein Spaltenvektor kann auch durch einen segmentierten Datensatz mit genau 1 Segment dargestellt werden.

Ein Zeilenvektor ist ein segmentierter Datensatz mit Segmentlänge 1.

Das Ergebnis der Funktion ist im allgemeinen Fall eine Matrix und damit ein segmentierter Datensatz. Ist das Ergebnis ein Spaltenvektor, ist es ein Datensatz ohne Segmente. Ist das Ergebnis eine einzige Zahl, wird sie als Spaltenvektor gedeutet und hat damit auch keine Segmente.

Beim Einfügen werden Nullen (0.0) eingefügt. Allerdings bleibt das Datenformat vorzugsweise erhalten. Wenn dabei z.B. bei skalierten ganzen Zahlen die 0.0 nicht darstellbar ist, wird ein möglichst naher Wert benutzt.

Einfügen ab Startindex=0 hängt an.

Die Matrix darf nicht leer sein. Um eine vorher leere Matrix auf eine Dimension > 0 zu bringen, wird MatrixInit() benutzt.

Das Ergebnis übernimmt die x- und z-Koordinate.

Beispiele:

3 Zeilen vorn bzw. oben einfügen

```
B = MatrixChangeDim ( A, 1, 3, "insert row" )
```

2 Zeilen anhängen

```
B = MatrixChangeDim ( A, 0, 2, "insert row" )
```

1. Spalte entfernen

```
B = MatrixChangeDim ( A, 1, 1, "remove col" )
```

Siehe auch:

MatrixPart, MatrixInit, SetSegLang

MatrixCut

Ein beliebiger gerader Schnitt wird durch eine Matrix gelegt. Die Schnittgerade wird durch 2 Punkte definiert.

Deklaration:

```
MatrixCut ( Matrix, EwX1, EwZ1, EwX2, EwZ2, EwBerechnung, EwSkalierungsachse, EwInkrement ) -> Ergebnis
```

Parameter:

Matrix	Durch diese Matrix wird der Schnitt gelegt.
EwX1	x-Koordinate des 1. Punktes
EwZ1	z-Koordinate des 1. Punktes
EwX2	x-Koordinate des 2. Punktes
EwZ2	z-Koordinate des 2. Punktes
EwBerechnung	Nach welcher Methode sollen Zwischenwerte berechnet werden?
	0 : Automatisch
	1 : Lineare Interpolation
	2 : Konstante Fortsetzung
	3 : Nahester Wert
EwSkalierungsachse	Sollen zur Beschriftung des Ergebnisses die x- oder z-Koordinaten der Matrix bevorzugt werden?
	0 : Automatisch
	1 : x-Koordinaten
	2 : z-Koordinaten
EwInkrement	Inkrement (Delta-X, dx) des Resultats. Kann automatisch gewählt oder fest vorgegeben werden
	0 : Automatisch
	>0 : Feste Vorgabe
Ergebnis	Schnittgerade

Beschreibung:

Die Matrix kann eine einfache Matrix sein (segmentierter Datensatz) oder eine Matrix mit erweiterten Eigenschaften (Datensatz mit Events, Beschreibung weiter unten).

Koordinaten:
Jedes Segment bildet eine Spalte der Matrix. Entlang der Spalte der Matrix stehen die x-Koordinaten. Damit legen die x-Koordinaten die Zeile fest. Die z-Koordinaten lassen einzelne Spalten unterscheiden. Bei segmentierten Datensätzen wird die z-Koordinate durch z0 (z-Offset) und Delta-Z (z-Inkrement) festgelegt.

Für einen horizontalen Schnitt ist z1=z2.

Für einen vertikalen Schnitt ist x1=x2.

Für einen schrägen Schnitt sind x1 und x2 verschieden, ebenso z1, z2.

Nicht erlaubt ist die Kombination der Eingangskoordinaten x1=x2 und z1=z2. Damit ist keine Gerade festgelegt, sondern nur ein Punkt.

Berechnung:
"Automatisch" und "Lineare Interpolation" interpolieren linear zwischen den Stützstellen (zuerst zwischen benachbarten x-Werten, dann zwischen benachbarten z-Werten). "Automatisch" ist empfohlen. Bei der Berechnung "Konstante Fortsetzung" wird ein Treppenstufen-Verhalten (vgl. Treppendarstellung am Kurvenfenster) nachgebildet. In Richtung steigender Indizes wird dabei ein Wert solange konstant angenommen, bis ein neuer vorliegt. Bei der Berechnung "Nahester Wert" wird der jeweils am nächsten zur Sollposition liegende Wert der Matrix benutzt.

Skalierungsachse:
Bei einem schrägen Schnitt kann das Ergebnis in x- oder z-Koordinaten der Matrix skaliert sein (d.h. so wird die x-Einheit des Ergebnisses gebildet). Bei horizontalem Schnitt kann nur über der x-Koordinate skaliert werden, bei vertikalem Schnitt nur über der z-Koordinate. Bei automatischer Wahl wird die z-Koordinate bevorzugt; nur wenn sie nicht möglich ist, wird die x-Koordinate gewählt.

Inkrement:
Automatische Wahl ist empfohlen. Dann wird stets ein Inkrement gewählt, bei dem keine wesentliche Information verlorengeht. Bei fester Wahl hingegen werden Punkte an Stellen ganzer Vielfacher des Inkrements erzeugt. Das feste Inkrement sollte günstig gewählt werden, damit keine Information verlorengeht und die Datenmenge nicht zu groß wird. Ein festes Inkrement sollte nur benutzt werden, wenn auch die Skalierungsachse fest vorgegeben ist.

Ergebnis:

Das Ergebnis ist ein gleichmäßig abgetasteter Datensatz. Je nach Schräglage der Linie wird eine passende Anzahl von Stützstellen gewählt, so dass keine Information verlorengeht. Ist eine andere Anzahl von Stützstellen gewünscht, kann die Funktion XYdt() anschließend angewendet werden.

Eventierte Eingangsdaten:
Ist die Matrix ein Datensatz mit Events, so gelten folgende Regeln: Jedes Event bildet eine Spalte der Matrix. Die z-Koordinate der Matrix muß streng monoton steigen. Alle Events müssen dieselbe Länge, denselben x-Offset (x0) und dieselbe Abtastzeit Delta-X (dx) haben. Das Ergebnis ist bei automatischem Inkrement (=0) ein XY-Datensatz. Da bei Events die z-Koordinate i.a. nicht gleichmäßig steigt, wird die Schnittgerade auch i.a. nicht gleichmäßig abgetastet sein. Ist hingegen ein festes Inkrement (ungleich Null) vorgegeben, wird ein gleichmäßig abgetasteter Datensatz erzeugt.

Beispiele:

Ein drehzahlabhängiges Spektrum "Drehzahlspektrum" liegt vor. In x-Richtung ist die Frequenz aufgetragen, in z-Richtung die Drehzahl. Das Spektrum bei der Drehzahl 2000 U/min soll herausgeschnitten werden.

```
Spektrum2000 = MatrixCut(Drehzahlspektrum, 0, 2000, 1, 2000, 0, 1, 0)
```

Dazu wird ein horizontaler Schnitt durchgeführt. Das Ergebnis soll in Hz (entsprechend der x-Koordinate der Matrix) skaliert sein. Die Koordinaten z1 und z2 sind egal, müssen nur verschieden sein.

Ein drehzahlabhängiges Spektrum "Drehzahlspektrum" liegt vor. In x-Richtung ist die Frequenz aufgetragen, in z-Richtung die Drehzahl. Der Verlauf der 50Hz Spektrallinie soll abhängig von der Drehzahl bestimmt werden.

```
Spektrallinie = MatrixCut(Drehzahlspektrum, 50, 0, 50, 1, 0, 2, 0)
```

Dazu wird ein vertikaler Schnitt durchgeführt. Das Ergebnis soll in U/min (Umdrehungen pro Minute, entsprechend der z-Koordinate der Matrix) skaliert sein. Die Koordinaten x1 und x2 sind egal, müssen nur verschieden sein.

Ein drehzahlabhängiges Spektrum "Drehzahlspektrum" liegt vor. In x-Richtung ist die Frequenz aufgetragen, in z-Richtung die Drehzahl. Der Verlauf der 3. Ordnung soll abhängig von der Drehzahl bestimmt werden. Die 3. Ordnung ist die Linie, bei der gilt: 3 * Drehzahl [U/min]= 60 * Frequenz [Hz]. Das Ergebnis soll alle 50 U/min einen Wert erhalten.

```
Ordnung3 = MatrixCut(Drehzahlspektrum, 0, 0, 3, 60, 0, 2, 50)
```

Dazu wird ein schräger Schnitt durchgeführt. Der erste Punkt ist der Nullpunkt. Der zweite legt die Proportionalität zwischen Frequenz und Drehzahl, also die Steigung der Geraden fest. Das Ergebnis soll in U/min (Umdrehungen pro Minute, entsprechend der z-Koordinate der Matrix) skaliert sein. Ein festes Inkrement von 50 U/min (z-Koordinate) wird vorgegeben.

Siehe auch:

MatrixTranspose, MatrixSumLines

MatrixEigen

Verfügbar ab: Professional Edition

Eigenwerte und Eigenvektoren einer Matrix

Deklaration:

```
MatrixEigen ( Matrix [, Option] ) -> Ergebnis
```

Parameter:

Matrix	Matrix
Option	Sollen Eigenwerte und/oder Eigenvektoren berechnet werden? (optional , Standardwert: "val")
	"val" : Eigenwerte
	"vect" : Eigenvektoren
	"val+vect" : Eigenwerte und Eigenvektoren
Ergebnis	Ergebnis

Beschreibung:

Die Funktion berechnet die Liste der Eigenwerte.

Die Funktion berechnet bei einer quadratische Matrix so viele Eigenwerte wie Zeilen. Gibt es einen Eigenwert mehrfach, taucht er auch mehrfach im Ergebnis auf.

Die Eigenwerte und Eigenvektoren werden als komplexe Zahlen in der Form Realteil/Imaginärteil ermittelt. Bei reellem Wert ist der Imaginärteil 0.

Eigenvektoren werden normiert ermittelt, sodass ihr größtes Element 1 ist. Die Eigenvektoren sind genauso sortiert wie die zugehörigen Eigenwerte.

Nur quadratische Matrizen werden unterstützt.

Die maximal unterstützte Dimension ist 20*20.

Die Matrix muss reell (nicht komplex) sein.

Die rechten Eigenvektoren werden bestimmt: A*x = Eig*x, wobei A die Matrix, x ein Eigenvektor und Eig der zugehörige Eigenwert ist.

Wenn die Funktion Eigenwerte zurückgibt, dann werden diese als ein Spaltenvektor zurückgegeben.

Wenn die Funktion Eigenvektoren zurückgibt, dann liegen diese in einer Matrix vor. Jeder Eigenvektor ist ein Spaltenvektor der Matrix.

Wenn die Funktion Eigenwerte und Eigenvektoren zurückgibt, dann ist das Ergebnis eine Matrix, deren erste Spalte die Eigenwerte darstellen. Die weiteren Spaltevektoren sind die Eigenvektoren.

Falls ein Eigenwert mit Vielfachheit > 1 vorliegt, kann es zu ihm gleiche wie auch unterschiedliche Eigenvektoren geben.

Vorsicht bei der Interpretation ist geboten, wenn aufgrund der Numerik ein Eigenwert mit größerer Vielfachheit zu mehreren nah benachbarten, aber unterschiedlichen Eigenwerten führt.

Vorsicht ist auch geboten, wenn es um die Frage geht, ob ein Eigenwert reell oder komplex ist. Denn aufgrund der Numerik kann ein reeller Eigenwert auch schon mal einen (kleinen) komplexen Anteil erhalten.

Die Funktion arbeitet mit einem QR-Algorithmus.

Beispiele:

Eigenwerte

```
A = leng(0,9) ; test data
setsegLen( A,3)
A[1,1] = 5
A[2,2] = 3
A[3,3] = 4
A[1,3] = 6 ; row 3, column 1
A[1,2] = 1 ; row 2, column 1
A[2,3] = 2 ; row 2, column 1
; A:
; 5 0 0
; 1 3 0
; 6 2 4
Eig = MatrixEigen ( A )
; Eig = [3, 4, 5]
Eig_r = Eig.r
```

Eigenwerte und Vektoren

```
A = leng(0,9) ; test data
```

```
setsegLen( A,3)
A[1,1] = 5
A[2,2] = 3
A[3,3] = 4
A[1,3] = 6 ; row 3, column 1
A[1,2] = 1 ; row 2, column 1
A[2,3] = 2 ; row 2, column 1
; A:
; 5 0 0
; 1 3 0
; 6 2 4
evv = MatrixEigen ( A, "val+vect" )
Eig = evv[1]
; Eig = [3, 4, 5]
ev1 = evv[2]
; ev1 = [ 0, -0.5, 1]
ev2 = evv[3]
; ev2 = [ 0, 0, 1]
ev3 = evv[4]
; ev3 = [ 1/7, 1/14, 1]
```

Siehe auch:

MatrixFromLine

Verfügbar ab: Professional Edition

Matrix bilden, indem aus einer Verteilung von Punkten (x,y,z) eine Fläche durch Interpolation gebildet wird.

Deklaration:

```
MatrixFromLine ( MatrixRef, Amplitude, ZeileX, SpalteZ [, Ersatz] ) -> Matrix
```

Parameter:

MatrixRef	Matrix, die als Referenz benutzt wird.
Amplitude	Amplitude, Höhe, Y, angegeben in physikalischen Einheiten
ZeileX	Zeile, Daten in X-Richtung, angegeben in physikalischen Einheiten
SpalteZ	Spalte, Daten in Z-Richtung, angegeben in physikalischen Einheiten
Ersatz	Ersatzwert, auf den der Bereich außerhalb der gebildeten Fläche gesetzt wird. (optional , Standardwert: 0)
Matrix	Matrix

Beschreibung:

Jedes Tripel aus Zeile, Spalte und Amplitude ist ein Stützpunkt einer Fläche y = y (x, z). Die Fläche zwischen den Stützpunkten wird linear interpoliert.

Die Fläche wird wie ein Gebirge gedeutet. Dabei wird jedem Punkt (jedem Element der Matrix) eine eindeutige Höhe zugeordnet.

Liegen mehrere Originalpunkte am selben Element der Matrix, wird das Maximum der Amplituden bevorzugt. Das entspricht einer Sicht auf die Fläche aus der Vogelpersepektive.

Die Fläche selbst ist ein konvexer Bereich.

Die Fläche ergibt sich, indem benachbarte Punkte durch Linien verbunden werden, die Dreiecke aufspannen. Die 3D-Darstellung am Kurvenfenster geht genauso vor.

Bei der Zerlegung in Dreiecke wird eine Auflösung von 1e-5 bezogen auf den Werteumfang von ZeileX bzw. SpalteZ angewendet.

Die resultierende Fläche ist i. Allg. nicht eindeutig definiert. Das ist leicht an 4 Punkten zu erkennen, die die Ecken eines Rechtecks bilden. Diese Fläche kann in 2 Dreiecke zerlegt werden. Da es aber 2 Diagonalen gibt, sind 2 unterschiedliche Zerlegungen möglich.

Die Tripel liegen nicht sortiert vor. Die Funktion interpoliert nie zwischen benachbarten Werten der übergebenen Daten (ZeileX[i], ZeileX[i+1]), sondern nur zwischen Werten, die auf der sich ergebenden Fläche nebeneinander liegen.

Die sich ergebende kontinuierliche Fläche wird genau an den Stützpunkten der Matrix abgetastet, also an x0+i*dx für die Zeile, an z0+k*dz für die Spalte. Zwischenwerte der Fläche werden nicht beachtet. Die Auflösung der Matrix ist fein genug zu wählen, damit keine wesentliche Information verloren geht.

An den Stützpunkten wird minimal gerundet: In einem Bereich von 1e-5 des Stüztpunktabstandes (dx bzw. dz) um einen Stützpunkt herum wird auf- bzw. abgerundet.

Die normalen Datensätze für Zeilen, Spalten und Amplitude haben dieselbe Länge. Sie werden ungeachtet ihrer Abtastzeit Punkt für Punkt ausgewertet.

Die Einheiten des Ergebnisses werden aus den Datensätzen Amplitude, ZeileX und SpalteZ abgelesen.

Das Ergebnis ist eine Matrix derselben Dimension wie die Referenz-Matrix. Die Referenz-Matrix gibt x0, dx, z0, dz und Zeilen- und Spaltenanzahl vor.

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Der Rechenaufwand steigt stark mit der Länge der übergebenen Daten. Eine Länge von 10000 gilt schon als groß.

Beispiele:

Matrix aus x, y, z Daten bilden. Alpha (=Amplitude) über Drehzahl N (in x-Richtung, Bereich 0 bis 6000 U/min) und Drehmoment M (in z-Richtung, Bereich 0 bis 500 Nm) dargestellt. Alpha, N und M sind Zeitdatensätze zur selben Zeitbasis.

```
Matrix = MatrixInit ( 61, 51, "0", "", 0, 100, "", 0, 10, "")
Matrix = MatrixFromLine ( Matrix, Alpha, N, M, 0)
```

Siehe auch:

MatrixSet, Setze, SetzeIndex

MatrixGet

Verfügbar ab: Professional Edition

Linearisierung (Korrektur) eines Datensatzes über ein 2dimensionales Kennlinienfeld. Elemente einer Matrix abfragen

Deklaration:

```
MatrixGet ( Matrix, Zeile, Spalte [, Interpolation] [, Skalierung] ) -> Ergebnis
```

Parameter:

Matrix	Matrix
Zeile	Zeile, adressiert in x-Richtung, also das soundsovielte Sample innerhalb eines Segments
Spalte	Spalte, adressiert in z-Richtung, also das soundsovielte Segment
Interpolation	Wie wird interpoliert? (optional , Standardwert: 4)
	0 : Konstant von links Liegt die gewünschte Position zwischen 2 Elementen, wird das linke (vordere) benutzt. Wirkt wie Abrunden des Index.
	1 : Linear: Zwischen den an die gewünschte Position angrenzenden Werten wird linear interpoliert und von dieser Vebindungsgeraden der Wert an der Position abgelesen. Das wird in Zeilenrichtung ausgeführt, mit diesem Ergebnis anschließend in Spaltenrichtung.
	3 : Konstant von rechts. Liegt die gewünschte Position zwischen 2 Elementen, wird das rechte(hintere) benutzt. Wirkt wie Aufrunden des Index.
	4 : Konstant nächster. Liegt die gewünschte Position zwischen 2 Elementen, wird der jeweils nähere benutzt. Wirkt wie Runden des Index.
Skalierung	Wie sind Zeile und Spalte skaliert? (optional , Standardwert: "index")
	"index" : Bei 1 beginnender Index
	"units" : In physikalischen Einheiten angegeben, also x0, dx bzw. z0, dz beinhaltend
Ergebnis	Ergebnis

Beschreibung:

Das Ergebnis ist ein Datensatz, der zu jedem Paar aus Zeile und Spalte ein Element aus der Matrix enthält.

Die Datensätze für Zeilen und Spalten haben dieselbe Länge und Struktur hinsichtlich Segmenten und Events.

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Liegen die angegebenen Werte für Zeile oder Spalte außerhalb der Matrix, wird der Randwert zurückgegeben.

Die Funktion ist geeignet zur Korrektur eines Datensatzes über ein 2dimensionales Kennlinienfeld. Dabei ist das Kennlinienfeld die Matrix, der Datensatz die Zeilen und die weitere Größe (z.B. Temperatur bei Temperaturabhängigkeit) die Spalte.

Der Ergebnisdatensatz übernimmt die Zeitbasis des Datensatzes Zeile. Der Datensatz Zeile darf äquidistant oder XY sein. Falls XY, übernimmt das Ergebnis seine Zeitspur. Vom Datensatz Spalte werden nur die Y-Werte Sample für Sample beachtet.

Bei den konstanten Interpolationen "konstant von rechts" und "konstant von links" erfolgt ein exakter Vergleich der Position. Vor allem bei Skalierung in physikalischen Einheiten kann ein kleinster Rundungsfehler der rellen Zahlen ein Kippen auf den daneben liegenden Wert bedeuten. Beim Arbeiten mit ganzzahligen Indizes kann man die Probleme vermeiden. Werden die Messpunkte selbst adressiert und nicht Zwischenwerte verlangt, ist "Konstant nächster" die passende Wahl.

In der Syntax der Sequenzen kann ein einziges Element einer Matrix direkt über eckige Klammern ermittelt werden, wobei der Segmentindex (=Spaltenindex) zuerst angegeben wird.

Beispiele:

Element abfragen

```
A = ramp(0,1,12) ; test data
setsegLen( A,4)
; A:
; 0 4 8
; 1 5 9
; 2 6 10
; 3 7 11
B = MatrixGet ( A, 3, 2 )
; B = 6
alternativ:
B = A[2,3]
```

Korrektur eines temperaturabhängigen Druckwertes über ein 2dimensionales Kennlinienfeld

```
Pressure = MatrixGet ( Charakteristic, Pressure_Raw, Temperature, 1, "units" )
```

Siehe auch:

Kenn, MatrixSet, Wert, Wert2, WertIndex, MatrixFromLine

MatrixInfo

Verfügbar ab: Professional Edition

Information zu einer Matrix abfragen

Deklaration:

```
MatrixInfo ( Matrix, Info ) -> Ergebnis
```

Parameter:

Matrix	Matrix
Info	Was soll abgefragt werden?
	"rows" : Zeilenanzahl
	"columns" : Spaltenanzahl
	"diag" : Diagonale (Hauptdiagonale), als Spaltenvektor zurückgegeben
	"trace" : Spur, Summe der Elemente der Hauptdiagonalen einer quadratischen Matrix
Ergebnis	Ergebnis

Beschreibung:

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Als Vektor wird in diesem Zusammenhang eine einreihige Matrix verstanden, also eine Matrix mit nur einer Zeile oder mit nur einer Spalte.

Ein Spaltenvektor ist eine Matrix mit nur einer Spalte. Er ist ein Datensatz ohne Segmente.

Ein Spaltenvektor kann auch durch einen segmentierten Datensatz mit genau 1 Segment dargestellt werden.

Ein Zeilenvektor ist ein segmentierter Datensatz mit Segmentlänge 1.

Beispiele:

Diagonale

```
A=ramp(0,1,9) ; test data
setsegLen( A,3)
; A:
; 0 3 6
; 1 4 7
; 2 5 8
Diag = MatrixInfo ( A, "diag" )
; Diag = [0, 4, 8]
```

Zeilenanzahl, Spaltenanzahl

```
A=ramp(0,1,12) ; test data
setsegLen( A,4)
; A:
; 0 4 8
; 1 5 9
; 2 6 10
; 3 7 11
rows = MatrixInfo ( A, "rows" )
; rows = 4
columns = MatrixInfo ( A, "columns" )
; columns = 3
```

Siehe auch:

MatrixMerge, SegLang?, Lang?

MatrixInit

Verfügbar ab: Professional Edition

Erzeugen einer vorinitialisierten Matrix

Deklaration:

```
MatrixInit ( Zeilen, Spalten [, Option] [, y-Einheit] [, x0] [, dx] [, x-Einheit] [, z0] [, dz] [, z-Einheit] )
-> Matrix
```

Parameter:

Zeilen	Anzahl der Zeilen der neuen Matrix > 0; in x-Richtung; Länge eines Segments
Spalten	Anzahl der Spalten der neuen Matrix > 0; in z-Richtung; Anzahl der Segmente
Option	Wie soll die Initialisierung stattfinden? (optional , Standardwert: "0")
	"0" : Alle Werte 0.0
	"I" : Hauptdiagonale 1.0, sonst 0.0
y-Einheit	y-Einheit, Einheit der Elemente (optional , Standardwert: "")
x0	x0, Offset entlang der Zeile, in x-Richtung (optional , Standardwert: 0)
dx	dx, Inkrement entlang der Zeile, in x-Richtung, > 0 (optional , Standardwert: 1)
x-Einheit	x-Einheit, Einheit entlang der Zeile, in x-Richtung (optional , Standardwert: "")
z0	z0, Offset entlang der Spalte, in z-Richtung (optional , Standardwert: 0)
dz	dz, Inkrement entlang der Spalte, in z-Richtung, > 0 (optional , Standardwert: 1)
z-Einheit	z-Einheit, Einheit entlang der Spalte, in z-Richtung (optional , Standardwert: "")
Matrix	Matrix

Beschreibung:

Das Ergebnis ist eine Matrix der Dimension [Zeilen*Spalten].

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Als Vektor wird in diesem Zusammenhang eine einreihige Matrix verstanden, also eine Matrix mit nur einer Zeile oder mit nur einer Spalte.

Ein Spaltenvektor ist eine Matrix mit nur einer Spalte. Er ist ein Datensatz ohne Segmente.

Ein Spaltenvektor kann auch durch einen segmentierten Datensatz mit genau 1 Segment dargestellt werden.

Ein Zeilenvektor ist ein segmentierter Datensatz mit Segmentlänge 1.

Das Ergebnis der Funktion ist im allgemeinen Fall eine Matrix und damit ein segmentierter Datensatz. Ist das Ergebnis ein Spaltenvektor, ist es ein Datensatz ohne Segmente. Ist das Ergebnis eine einzige Zahl, wird sie als Spaltenvektor gedeutet und hat damit auch keine Segmente.

Beispiele:

Einheitsmatrix

```
A = MatrixInit ( 3, 3, "I" )
; A:
; 1 0 0
; 0 1 0
; 0 0 1
```

Nullmatrix

```
A = MatrixInit ( 3, 4, "0" )
; A:
; 0 0 0 0
; 0 0 0 0
; 0 0 0 0
```

Siehe auch:

SetSegLang

MatrixInverse

Verfügbar ab: Professional Edition

Bestimmung der inversen Matrix

Deklaration:

```
MatrixInverse ( Matrix [, Fehlerbehandlung] ) -> Ergebnis
```

Parameter:

Matrix	Matrix
Fehlerbehandlung	Wie soll im Fehlerfall reagiert werden? (optional , Standardwert: 0)
	0 : Abbruch mit Fehlermeldung
	1 : Leeren Datensatz zurückgeben
Ergebnis	Inverse Matrix

Beschreibung:

Das Ergebnis ist eine Matrix derselben Dimension.

Nur eine eindeutige Lösung wird als Lösung zurückgegeben. Ist die Lösung nicht eindeutig oder existiert keine Lösung, wird die Fehlerbehandlung wirksam.

Bei Aufruf mit falschen Parametern oder nicht ausreichend Speicher wird stets mit der üblichen Fehlermeldung abgebrochen.

Einheiten und Abtastzeiten werden nicht beachtet.

Die Matrix muss quadratisch sein.

Beispiele:

Invertieren einer Matrix A

```
A = leng(0,9) ; test data
setsegLen( A,3)
A[3,1] = 0.25 ; row 1, column 3
A[1,2] = 1
A[2,3] = 2
; A:
; 0 0 0.25
; 1 0 0
; 0 2 0
C = MatrixInverse ( A )
; C:
; 0 1 0
; 0 0 0.5
; 4 0 0
```

Siehe auch:

SolveLinEq

MatrixIpol

Verfügbar ab: Professional Edition

Interpolation einer Matrix entlang Zeilen und Spalten

Deklaration:

```
MatrixIpol ( Matrix, FaktorZeileX, FaktorSpalteZ, Interpolation ) -> Ergebnis
```

Parameter:

Matrix	Matrix
FaktorZeileX	Faktor in Richtung der Zeile (x). Interpolation innerhalb eines Segmentes.
FaktorSpalteZ	Faktor in Richtung der Spalte (z). Interpolation zwischen den Segmenten durch Einfügen von weiteren Segmenten.
Interpolation	Wie wird interpoliert?
	0 : Konstant. Vervielfachung, indem zwischen 2 benachbarten Werten der linke(vordere) vervielfacht wird. Auch Vervielfachung des letzten (Extrapolation).
	1 : Linear. Vervielfachung, indem zwischen 2 benachbarten Werten linear interpolierende Werte ergänzt werden (Verbindungsgerade).
	2 : Kubischer Spline. Vervielfachung, indem durch die Messwertfolge ein kubischer Spine gelegt wird. Aus dem Spline werden interpolierende Werte gebildet.
	3 : Konstant von rechts. Vervielfachung, indem zwischen 2 benachbarten Werten der rechte(hintere) vervielfacht wird. Auch Vervielfachung des letzten (Extrapolation).
	4 : Konstant nächster: Vervielfachung, indem zwischen 2 benachbarten Werten der jeweils nähere Wert zur Vervielfachung benutzt wird. Auch Vervielfachung des letzten (Extrapolation).
	5 : Linear mit letzter Treppe: Vervielfachung, indem zwischen 2 benachbarten Werten linear interpolierende Werte ergänzt werden (Verbindungsgerade). Der letzte Wert wird vervielfacht (Extrapolation).
Ergebnis	Interpolierte Matrix

Beschreibung:

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Die Abtastzeit des erzeugten Datensatzes ist um den übergebenen FaktorZeileX kleiner, dz um den Faktor FaktorSpalteZ.

Die Faktoren vervielfachen im Wesentlichen die Datenmenge durch Interpolation. Bei den nicht extrapolierenden Interpolationen wird der Datensatz in der betreffenden Richtung nicht ganz um den übergebenen Faktor vergrößert. Es fehlen (Faktor -1) Werte in der Richtung.

Hat der Eingangsdatensatz keine Segmente, ist er ein Spaltenvektor und kann nur mittels FaktorZeileX in x-Richtung interpoliert werden.

Die Funktion kann auch auf Daten mit Events, aber ohne Segmente angewendet werden, führt dann aber nur eine Interpolation in x-Richtung aus.

RGB-Daten werden bei konstanter und linearer Interpolation berücksichtigt.

Beispiele:

Verfeinerung eines grob aufgelösten Kennlinienfeldes um den Faktor 10

```
Map = MatrixIpol( Map, 10, 10, 2 )
```

Siehe auch:

Ipol, Ipli

MatrixMerge

Verfügbar ab: Professional Edition

Eine kleinere Matrix in eine größere an bestimmter Position kopieren

Deklaration:

```
MatrixMerge ( MatrixA, MatrixB, Startzeile, Startspalte ) -> Matrix
```

Parameter:

MatrixA	Matrix A
MatrixB	Matrix B
Startzeile	Ab diesem Zeilenindex, > 0; in x-Richtung; Index des Samples innerhalb eines Segments
Startspalte	Ab diesem Spaltenindex, > 0; in z-Richtung; Index des Segmentes
Matrix	Matrix

Beschreibung:

Das Ergebnis ist eine Matrix derselben Dimension wie Matrix A.

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Als Vektor wird in diesem Zusammenhang eine einreihige Matrix verstanden, also eine Matrix mit nur einer Zeile oder mit nur einer Spalte.

Ein Spaltenvektor ist eine Matrix mit nur einer Spalte. Er ist ein Datensatz ohne Segmente.

Ein Spaltenvektor kann auch durch einen segmentierten Datensatz mit genau 1 Segment dargestellt werden.

Ein Zeilenvektor ist ein segmentierter Datensatz mit Segmentlänge 1.

Das Ergebnis der Funktion ist im allgemeinen Fall eine Matrix und damit ein segmentierter Datensatz. Ist das Ergebnis ein Spaltenvektor, ist es ein Datensatz ohne Segmente. Ist das Ergebnis eine einzige Zahl, wird sie als Spaltenvektor gedeutet und hat damit auch keine Segmente.

Sollen Teilmatrizen zu einer größeren ergänzt werden, so kann z.B. mit MatrixInit() zunächst die alles umfassende Matrix mit Nullen erzeugt werden. Anschließend wird mit mehreren Aufrufen von MatrixMerge() hineinkopiert.

Die Diagonale (Hauptdiagonale) einer Matrix A wird gesetzt, wenn Matrix B ein Vektor (Zeilen- oder Spaltenvektor) ist und gleichzeitig Startzeile=-1 und Startspalte =-1.

Das Ergebnis übernimmt die x- und z-Koordinate der Matrix A.

Beispiele:

1 Zeile setzen

```
A=ramp(0,1,12) ; test data
setsegLen( A,4)
; A:
; 0 4 8
; 1 5 9
; 2 6 10
; 3 7 11
B = TransposeMatrix([5, 5, 5])
C = MatrixMerge ( A, B, 3, 1 )
; C:
; 0 4 8
; 1 5 9
; 5 5 5
; 3 7 11
```

Diagonale setzen

```
A=ramp(0,1,9) ; test data
setsegLen( A,3)
; A:
; 0 3 6
; 1 4 7
; 2 5 8
Diag = [2, 2, 1]
C = MatrixMerge ( A, Diag, -1, -1 )
; C:
; 2 3 6
; 1 2 7
; 2 5 1
```

Siehe auch:

MatrixPart, StückIndex, MatrixChangeDim, MatrixInfo

MatrixMult

Verfügbar ab: Professional Edition

Multiplikation zweier Matrizen oder Vektoren, auch ihrer Transponierten

Deklaration:

```
MatrixMult ( Matrix A, Matrix B [, Berechnung] ) -> Produkt
```

Parameter:

Matrix A	Matrix bzw. Vektor A
Matrix B	Matrix bzw. Vektor B
Berechnung	Sollen die Matrizen vor der Multiplikation transponiert werden? (optional , Standardwert: "A*B")
	"A*B" : A * B
	"AT*B" : A transponiert * B
	"A*BT" : A * B transponiert
	"AT*BT" : A transponiert * B transponiert
	"cross" : Kreuzprodukt zweier 3dimensionaler Spaltenvektoren
Produkt	Produkt

Beschreibung:

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Als Vektor wird in diesem Zusammenhang eine einreihige Matrix verstanden, also eine Matrix mit nur einer Zeile oder mit nur einer Spalte.

Ein Spaltenvektor ist eine Matrix mit nur einer Spalte. Er ist ein Datensatz ohne Segmente.

Ein Spaltenvektor kann auch durch einen segmentierten Datensatz mit genau 1 Segment dargestellt werden.

Ein Zeilenvektor ist ein segmentierter Datensatz mit Segmentlänge 1.

Das Ergebnis der Funktion ist im allgemeinen Fall eine Matrix und damit ein segmentierter Datensatz. Ist das Ergebnis ein Spaltenvektor, ist es ein Datensatz ohne Segmente. Ist das Ergebnis eine einzige Zahl, wird sie als Spaltenvektor gedeutet und hat damit auch keine Segmente.

Beim Skalarprodukt wird Zeilenvektor mit Spaltenvektor multipliziert. Liegen z.B. 2 Spaltenvektoren vor, wird "AT*B" benutzt.

Beim dyadischen Produkt wird Spaltenvektor mit Zeilenvektor multipliziert. Liegen z.B. 2 Spaltenvektoren vor, wird "A*BT" benutzt.

Diese Funktion wird nicht benutzt, wenn die Matrix mit einem (skalaren) Faktor multipliziert wird. Das erledigt das Multiplikationszeichen "*". Ein einfacher Faktor vom Typ Einzelwert wird als Skalar gedeutet.

Addieren und Subtrahieren von Matrizen mit den Zeichen "+" und "-".

Addieren und Subtrahieren von Matrizen und ihrere Transponierten auch mit MatrixAdd().

Zum Dividieren durch eine Matrix wird mit der inversen Matrix multipliziert.

Einheiten und Abtastzeiten werden nicht beachtet.

Beispiele:

Matrix mit Spaltenvektor multiplizieren

```
A = leng(0,9)  ; test data
setsegLen( A,3)
A[3,1] = 4 ; row 1, column 3
A[1,2] = 1
A[2,3] = 2
; A:
; 0 0 4
; 1 0 0
; 0 2 0
b = [3, 2, -3]
; b:
; 3
; 2
; -2
c = MatrixMult( A, b, "A*B" )
; c:
; -12
; 3
; 4
```

Siehe auch:

MatrixAdd

MatrixPart

Verfügbar ab: Professional Edition

Teil einer Matrix bilden

Deklaration:

```
MatrixPart ( Matrix, Startzeile, Zeilen, Startspalte, Spalten ) -> Teilmatrix
```

Parameter:

Matrix	Matrix
Startzeile	Ab diesem Zeilenindex, > 0; in x-Richtung; Index des Samples innerhalb eines Segments
Zeilen	Anzahl der zu übernehmenden Zeilen
Startspalte	Ab diesem Spaltenindex, > 0; in z-Richtung; Index des Segmentes
Spalten	Anzahl der zu übernehmenden Spalten
Teilmatrix	Teilmatrix

Beschreibung:

Das Ergebnis ist eine Matrix der Dimension [Zeilen*Spalten].

Die Funktion liefert eine Teilmatrix. Der auszuschneidende Bereich von Zeilen und Spalten muss sich komplett innerhalb der Matrix befinden. Hat z.B. die Matrix 4 Zeilen, so dürfen ab der 1. Zeile 4 Zeilen ausgeschnitten werden, aber der 4. Zeile nur noch eine.

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Als Vektor wird in diesem Zusammenhang eine einreihige Matrix verstanden, also eine Matrix mit nur einer Zeile oder mit nur einer Spalte.

Ein Spaltenvektor ist eine Matrix mit nur einer Spalte. Er ist ein Datensatz ohne Segmente.

Ein Spaltenvektor kann auch durch einen segmentierten Datensatz mit genau 1 Segment dargestellt werden.

Ein Zeilenvektor ist ein segmentierter Datensatz mit Segmentlänge 1.

Das Ergebnis der Funktion ist im allgemeinen Fall eine Matrix und damit ein segmentierter Datensatz. Ist das Ergebnis ein Spaltenvektor, ist es ein Datensatz ohne Segmente. Ist das Ergebnis eine einzige Zahl, wird sie als Spaltenvektor gedeutet und hat damit auch keine Segmente.

Das Ergebnis übernimmt die x- und z-Koordinate, liefert aber entsprechend verschobene x- und z-Offsets.

Beispiele:

1 Zeile, nämlich die dritte, ermitteln

```
A=ramp(0,1,12) ; test data
setsegLen( A,4)
; A:
; 0 4 8
; 1 5 9
; 2 6 10
; 3 7 11
B = MatrixPart ( A, 3, 1, 1, 4 )
; B = [2, 6, 10]
```

Teilmatrix

```
A=ramp(-4,1,16) ; test data
setsegLen( A,4)
; A:
; -4 0 4 8
; -3 1 5 9
; -2 2 6 10
; -1 3 7 11
B = MatrixPart ( A, 2, 2, 3, 2 )
; B:
; 5 9
; 6 10
```

Siehe auch:

MatrixMerge, GrenIndex, CutDt

MatrixSet

Verfügbar ab: Professional Edition

Elemente einer Matrix setzen

Deklaration:

```
MatrixSet ( Matrix, Zeile, Spalte, Neu [, Skalierung] ) -> Matrix
```

Parameter:

Matrix	Matrix
Zeile	Zeile, adressiert in x-Richtung, also das soundsovielte Sample innerhalb eines Segments
Spalte	Spalte, adressiert in z-Richtung, also das soundsovielte Segment
Neu	Neue Werte
Skalierung	Wie sind Zeile und Spalte skaliert? (optional , Standardwert: "index")
	"index" : Bei 1 beginnender Index
	"units" : In physikalischen Einheiten angegeben, also x0, dx bzw. z0, dz beinhaltend
Matrix	Matrix

Beschreibung:

Jedes Wertepaar von Zeile und Spalte addressiert ein Element in der Matrix. Dieses Element wird auf den neuen Wert gesetzt.

Die Zahlenwerte aus Zeile und Spalte werden auf das nächste Element gerundet. Liegt ein Index nach Rundung außerhalb der Matrix, wird kein Element der Matrix verändert.

Die Datensätze für Zeilen, Spalten und neue Werte haben dieselbe Länge und Struktur hinsichtlich Segmenten und Events.

Das Ergebnis ist eine Matrix derselben Dimension wie die übergebene Matrix.

Die Datensätze für Zeile und Spalte können Events und Segmente haben.

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

In der Syntax der Sequenzen kann auch das Element einer Matrix direkt über eckige Klammern gesetzt werden, wobei der Segmentindex (=Spaltenindex) zuerst angegeben wird.

Beispiele:

Element setzen

```
A = MatrixInit ( 3, 3, "I" )
; A:
; 1 0 0
; 0 1 0
; 0 0 1
B1 = MatrixSet ( A, 3, 2, 5 )
; B:
; 1 0 0
; 0 1 0
; 0 5 1
alternativ:
B2 = A
B2[2,3] = 5
```

Siehe auch:

MatrixGet, MatrixFromLine, Setze, SetzeIndex

MatrixSumLines

Die Funktion bestimmt den Vektor der Zeilen- oder Spaltensummen einer Matrix (segmentierter Datensatz).

Deklaration:

```
MatrixSumLines ( Matrix, EwOption ) -> Summe
```

Parameter:

Matrix	Matrix (segmentierter Datensatz)
EwOption	Optionsparameter
	0 : Summe jeweils über den Inhalt einer Zeile. Die erste Zeile besteht aus dem 1. Element eines jeden Segmentes.
	1 : Summe jeweils über den Inhalt einer Spalte, also eines Segmentes.
Summe	Vektor mit Spalten- oder Zeilensummen

Beschreibung:

Der übergebene segmentierte Datensatz wird als Matrix interpretiert (1 Segment entspricht einer Spalte) und in Abhängigkeit vom Optionsparameter die Zeilen- bzw. Spaltensummen bestimmt.

Beispiele:

Eine Matrix (segmentierter Datensatz) habe die folgende Form:

```
; 2 1 0
; 4 1 1
; 5 2 0
; mit 2,4,5 = Segment 1
; Bestimmung der Spaltensumme:
Matrix = [2,4,5,1,1,2,0,1,0]
SetSegLen(Matrix, 3)
columnSum = MatrixSumLines(Matrix, 1)
; Das Ergebnis ist ein Datensatz mit den Werten (11,4,1), jedes Element ist die Summe einer Matrixspalte.
rowSume = MatrixSumLines(Matrix, 0)
; Das liefert ( 3, 6, 7 ).
```

Siehe auch:

MatrixTranspose, MatrixCut

MatrixTranspose

Eine Matrix (segmentierter Datensatz) wird transponiert (Zeilen und Spalten werden vertauscht).

Deklaration:

```
MatrixTranspose ( Matrix ) -> TransponierteMatrix
```

Parameter:

Matrix	Matrix (segmentierter Datensatz)
TransponierteMatrix	Resultierende transponierte Matrix

Beschreibung:

Der als Parameter übergebene segmentierte Datensatz wird als Matrix interpretiert, 1 Segment entspricht einer Spalte der Matrix. Die Funktion transponiert diese Matrix, d.h. Zeilen und Spalten werden vertauscht.

Beispiele:

Die Funktion AmpSpectrumRMS() aus dem imc Spektralpaket berechnet eine Serie von Spektren, die als Segmente im Ergebnis abgelegt werden. Diese Ergebnis-Matrix besteht aus Amplitudenwerden, aufgetragen über Zeit und Frequenz. Anschließend wird die Zeit- und Frequenzachse vertauscht.

```
spectra = AmpSpectrumRMS(signal, 1024, 0, 50, 1, 0)
tm = MatrixTranspose(spectra)
```

Siehe auch:

MatrixSumLines, MatrixCut, MatrixMult

Max

Das Maximum eines Datensatzes wird ermittelt.

Deklaration:

```
Max ( Daten ) -> EwMaximum
```

Parameter:

Daten	Datensatz, von dem das Maximum ermittelt werden soll. [ND],[XY]
EwMaximum	Maximum des Datensatzes

Beschreibung:

Die Funktion Max() liefert den größten Zahlenwert des Datensatzes (y-Wert). Die Lage des Maximums (x-Koordinate) kann mit der Funktion Pos() berechnet werden.

Beispiele:

Das Maximum eines sinusförmigen Datensatzes ist seine Amplitude, auch Spitzenwert genannt:

```
Amplitude = Max(sineWave)
```

Im Histogramm eines ASCII-Textes ist zu ermitteln, wie oft der häufigste Großbuchstabe auftritt. Großbuchstaben haben die Werte 65 ... 90:

```
thePeak = Max(Cut(HistoText, 65, 90))
```

..und der häufigste Großbuchstabe selbst:

```
letter = Pos(HistoText, thePeak)
```

Siehe auch:

Min, Mean, Pos, PosiEx, SearchLevel, MvMax, Stat

MDIR

Verzeichnis zum Laden und Speichern von Sequenzen und Anwenderdefinierten Dialogen setzen

Deklaration:

```
MDIR Verzeichnis
```

Parameter:

Verzeichnis	Vollständiger Pfadname des gewünschten Verzeichnisses.

Beschreibung:

Statt dem Kommando MDIR sollte in neu zu erstellenden Sequenzen die Funktion SetOption() verwendet werden.

Das Verzeichnis zum Laden und Sichern von Sequenzen/Dialogen mit wird neu gesetzt.

Nach Ausführung dieses Befehls wird dieses Verzeichnis zum Laden und Sichern von Sequenzen und Anwenderdefinierten Dialogen benutzt.

Dies betrifft den Befehl SEQUENCE, die Funktion Dialog() sowie die Dialoge zum manuellen Laden und Speichern von Sequenzen und Dialogen.

Der Verzeichnisname darf auch in Anführungszeichen angegeben werden. Dies ist dann zwingend notwendig, wenn Leerzeichen im Namen enthalten sind.

Dieses Kommandos kann auch ohne Parameter (also ohne Verzeichnisangabe) aufgerufen werden. Dann wird das unter "Optionen/Verzeichnisse" eingestellte Verzeichnis wieder als aktueller Standard verwendet.

Das hier gewählte Verzeichnis bleibt gültig bis:

- der Befehl MDIR erneut aufgerufen wird
- die Funktion SetOption("Dir.Sequences",...) aufgerufen wird
- eine Sequenz- oder Dialog-Datei manuell aus einem anderen Verzeichnis geladen oder in ein anderes Verzeichnis gesichert wird
- mit dem Dialog "Optionen"/ "Verzeichnisse" ein neues Verzeichnis für Sequenzen/Dialoge definiert wird

Multithreading: Der Befehl wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
SEQUENCE SEQ1
MDIR C:\MAKRO
SEQUENCE SEQ2
```

Aus dem aktuell eingestellten Verzeichnis wird die Sequenz SEQ1 aufgerufen, anschließend die Sequenz C:\MAKRO\SEQ2.SEQ

```
MDIR "c:\Meine Versuche vom 12.1.98"
```

Der Pfadname enthält Leerzeichen und muss deshalb in Anführungszeichen geschrieben werden.

Siehe auch:

SetOption, SEQUENCE, Dialog, LDIR

Mean

Arithmetischer Mittelwert eines Datensatzes
Alternativer Name: **Mitte**

Deklaration:

```
Mean ( Daten ) -> EwMittelwert
```

Parameter:

Daten	Daten, von denen der Mittelwert berechnet werden soll [ND].
EwMittelwert	Arithmetischer Mittelwert des Datensatzes

Beschreibung:

Der arithmetische Mittelwert der Zahlenwerte (y-Werte) eines Datensatzes wird ermittelt. Dieser statistische Kennwert ist definiert als die Summe aller Zahlenwerte dividiert durch deren Anzahl.

Beispiele:

Ein pH-Wert ist zu ermitteln. Da in der Nähe der Messeinrichtung ein stark störender Stromrichter arbeitet, wurde die Messgröße 100 mal gemessen. Der Mittelwert über alle Messungen kann als wesentlich verbesserter Messwert angesehen werden:

```
PH_mean = Mean(PH_allValues)
```

Aus einem Beschleunigungssignal wird durch Integration die Geschwindigkeit berechnet. Dazu muss das Signal zunächst mittelwertfrei gemacht werden:

```
v = Int(a - Mean(a))
```

Siehe auch:

Max, Min, StDev, RMS, MvMean, Stat

MeasChanNames?

Liefert die Namen aller Kanäle, die einer gegebenen Messung zugeordnet sind.

Deklaration:

```
MeasChanNames? ( TxMessungsname, TxAuswahlfilter [, SvOption] ) -> TxKanalNamen
```

Parameter:

TxMessungsname	Name der anzufragenden Messung.
TxAuswahlfilter	Leerer Text, um alle Kanäle zu erhalten. Ansonsten können die Jokerzeichen '?' (ein beliebiges Zeichen) und '*' (beliebige Anzahl beliebiger Zeichen) verwendet werden, um ein Auswahlfilter zu definieren.
SvOption	Option (optional)
	0 : Gruppenkanäle werden in der üblichen Schreibweise 'GruppenName:KanalName' geliefert. Der Gruppenname selbst ist im Ergebnis nicht enthalten.(Standard)
	1 : Für jede enthaltene Gruppe wird nur der Gruppenname geliefert.
TxKanalNamen	Textfeld mit Kanalnamen

Beschreibung:

Die Funktion liefert die Namen aller Kanäle, die einer gegebenen Messung zugeordnet sind. Dies entspricht dem Inhalt des entsprechenden Messungs-Knotens in der Variablenliste/Messungsansicht.

Wenn der jeweilige Name nicht den gültigen Regeln für Variablennamen in Sequenzen gehorcht (z.B. Sonderzeichen oder Leerzeichen enthält), wird der Name zusätzlich in **{...}** eingeklammert.

Das Konzept der Messungszugehörigkeit einer Variablen findet bisher hauptsächlich im Datenquellen-Browser Anwendung. Dort wird beim Laden einer Messwert-Datei jeder erzeugten Variablen automatisch ein Messungsname zugeordnet, um gleichnamige Variablen, die aus verschiedenen Datenquellen stammen, bequem unterscheiden zu können.

Die zurückgegebenen Namen sind alphabetisch sortiert.

Beispiele:

Ein Panel enthält ein Kurvenfenster, welches zur Anzeige des 1. Kanals der 1. selektierten Messung konfiguriert ist ('Kanal #1 @ Messung #1') und einen Knopf 'Zeigen'. Der Anwender lädt zunächst manuell alle gewünschten Messungen. Auf Knopfdruck werden dann für jede Messung alle Kanäle, die mit 'U_' anfangen, für jeweils für 2 Sekunden angezeigt.

Ereignis-Sequenz 'Knopf gedrückt (Zeigen)':

```
measurements = MeasNames?("*")
FOREACH ELEMENT name in measurements
   SelMeasListSetName(1, name, 1)
   channels = MeasChanNames?(name, "U_*")
   FOREACH ELEMENT chan in channels
      SelChanListSetName(1, chan, "", 1)
      SelListControl(0) ; jetzt Kurvenfenster aktualisieren
      Sleep(2)
   END
END
```

Siehe auch:

MeasNames?, SelMeasListSetName, SelChanListSetName, SetMeasurementName

MeasNames?

Liefert die Namen aller aktuell geladenen Messungen.

Deklaration:

```
MeasNames? ( TxAuswahlfilter ) -> TxMessungsNamen
```

Parameter:

TxAuswahlfilter	Leerer Text, um alle Messungen zu erhalten. Ansonsten können die Jokerzeichen '?' (ein beliebiges Zeichen) und '*' (beliebige Anzahl beliebiger Zeichen) verwendet werden, um ein Auswahlfilter zu definieren.
TxMessungsNamen	Textfeld mit Messungsnamen

Beschreibung:

Die Funktion liefert die Namen aller aktuell geladenen Messungen. Dies entspricht dem (ungefilterten) Inhalt der Variablenliste/Messungsansicht.

Das Konzept der Messungszugehörigkeit einer Variablen findet bisher hauptsächlich im Datenquellen-Browser Anwendung. Dort wird beim Laden einer Messwert-Datei jeder erzeugten Variablen automatisch ein Messungsname zugeordnet, um gleichnamige Variablen, die aus verschiedenen Datenquellen stammen, bequem unterscheiden zu können.

Die zurückgegebenen Namen sind alphabetisch sortiert.

Beispiele:

Ein Panel enthält ein Kurvenfenster, welches zur Anzeige des 1. Kanals der 1. selektierten Messung konfiguriert ist ('Kanal #1 @ Messung #1') und einen Knopf 'Zeigen'. Der Anwender lädt manuell alle gewünschten Messungen und selektiert einen Kanalnamen (bekommt die Nummer #1) in der Variablenliste/Messungsansicht. Auf Knopfdruck werden alle aktuell vorhandenen Messungen aufgezählt und das entsprechende Kurvenbild für den gewählten Kanal jeweils für 2 Sekunden angezeigt.

Ereignis-Sequenz 'Knopf gedrückt (Zeigen)':

```
measurements = MeasNames?("")
FOREACH ELEMENT name IN measurements
   SelMeasListSetName(1, name)
   Sleep(2)
END
```

Siehe auch:

MeasChanNames?, SelMeasListSetName, SetMeasurementName

MeasurementName?

Ermittlung des Namens der Messung, der die Variable zugeordnet ist.

Deklaration:

```
MeasurementName? ( Variable ) -> TxMessungsName
```

Parameter:

Variable	Abzufragende Variable.
TxMessungsName	Name der zugeordneten Messung.

Beschreibung:

Wenn der Variablen keine Messung zugeordnet ist, wird ein leerer Text zurückgeliefert.

Das Konzept der Messungszugehörigkeit einer Variablen findet bisher hauptsächlich im Datenquellen-Browser Anwendung. Dort wird beim Laden einer Messwert-Datei jeder erzeugten Variablen automatisch ein Messungsname zugeordnet, um gleichnamige Variablen, die aus verschiedenen Datenquellen stammen, bequem unterscheiden zu können.

Beispiele:

Die aktuell bearbeiteten Messungen enthalten die Kanäle 'Voltage' und 'Current'. Zur aktuell selektierten Messung #1 (Variablenliste/Messungsansicht) wird die Leistung berechnet und mit dem entsprechenden Messungsnamen versehen. Damit wird die berechnete Variable automatisch in der Kanalliste (Variablenliste/Messungsansicht) eingeblendet.

```
IF SelUseMeasurement(1) = 0
   EXITSEQUENCE
END
TxMeasName = MeasurementName?(Voltage)
Power = Voltage * Current
SetMeasurementName(Power, TxMeasName)
```

Siehe auch:

SetMeasurementName

Median

Signalglättung mittels Median-Filterung über eine vorgebbare Punktezahl.

Deklaration:

```
Median ( Daten, EwFilterbreite ) -> Filtrat
```

Parameter:

Daten	Zu filternder Datensatz.
EwFilterbreite	Filterbreite in Punkten
Filtrat	Geglätteter Datensatz

Beschreibung:

Die Medianfilterung verwendet folgenden Algorithmus (Beispiel für eine Filterbreite von 5): Für jeden Eingangswert an der Stelle [n] wird ein Ergebniswert gebildet. Dazu werden die Eingangswerte an den Positionen [n-2], [n-1], [n], [n+1] und [n+2] (also insgesamt 5 Werte) entsprechend ihrer Amplitude sortiert. Der Ergebniswert ist jeweils das mittlere Element der sortierten Liste.

Die Filterbreite muß im Bereich von 3 bis 99 liegen und ungerade sein, sinnvolle Werte sind z.B. 3 oder 5. Wenn eine gerade Filterbreite übergeben wird, wird auf den nächsten ungeraden Wert abgerundet.

Die Median-Filterung kann angewendet werden, um kurze, deutliche Störspitzen im Eingangs-Signal zu unterdrücken. Die Filterbreite muss dazu mehr als doppelt so breit wie die erwartete Breite der Störspitzen gewählt werden.

Auf der anderen Seite kann das Signal durch diese Funktion stark verfälscht werden, insbesondere dann, wenn das Signal starkt schwingt. Wird diese Funktion dagegen auf im wesentlichen monotone Signale bzw. Signalabschnitte angewendet, ist die Verfälschung nur gering.

Der Eingangsdatensatz wird am Anfang und Ende als konstant fortgesetzt angenommen. Am Anfang und Ende des Datensatzes findet somit innerhalb der halben Filterbreite ein Ein- bzw. Ausschwingen statt, Störungen können hier u.U. nicht unterdrückt werden.

Das Median-Filter ist ein nichtlineares Filter, Eingangs- und Ausgangsdatensatz sind nicht phasenverschoben.

Beispiele:

Medianfilterung über 5 Punkte. Anschließend werden die beiden ersten und die beiden letzten Werte des Ergebnisses abgeschnitten.

```
filtered = Median(signal, 5)
filtered = CutIndex(filtered, 3, Leng?(filtered)-2)
```

Siehe auch:

Smo, MvMean, MvMin, MvMax, Sort

Min

Das Minimum eines Datensatzes wird ermittelt.

Deklaration:

```
Min ( Daten ) -> EwMinimum
```

Parameter:

Daten	Datensatz, von dem das Minimum ermittelt werden soll. [ND],[XY]
EwMinimum	Minimum des Datensatzes

Beschreibung:

Die Funktion Min() liefert den kleinsten Zahlenwert des Datensatzes (y-Wert). Die Lage des Minimums (x-Koordinate) kann mit der Funktion Pos() berechnet werden.

Beispiele:

Ein Datensatz enthält die Messfehler einer Apparatur zu verschiedenen Zeiten. Der betragsmäßig kleinste Fehler ist zu ermitteln:

```
mini = Min(Abs(error))
```

Die Versorgungsspannung eines Gerätes wurde gemessen. Es ist der minimal aufgetretene Wert zu ermitteln und als Abweichung in Prozent vom Sollwert 12V auszudrücken:

```
deviation  = (Min(U_supply) -12'V') * 100'%' / 12'V'
```

Siehe auch:

Max, Mean, Pos, PosiEx, SearchLevel, MvMin, Stat

MInt

Gleitendes Integral über vorgebbare Integrationsintervallbreite
Alternativer Name: **GInt**

Deklaration:

```
MInt ( Daten, EwBreite ) -> Ergebnis
```

Parameter:

Daten	Zu integrierender Datensatz. Erlaubte Datentypen: [ND]
EwBreite	Breite des Integrationsintervalls
Ergebnis	Ergebnis des gleitenden Integrals.

Beschreibung:

Beim gleitenden Integral hat das Integrationsintervall eine fest vorgebbare Breite. Dieses konstante Integrationsintervall wird über den gesamten Datensatz geschoben. Die Werte dieses Integrals bilden aneinandergereiht den erzeugten Datensatz. Die Integration wird durchgeführt, indem alle Werte des übergebenen Datensatzes innerhalb des Integrationsintervalls summiert und mit der Abtastzeit multipliziert werden.

Folgende Gleichung zeigt das gleitende Integral:

$$GInt(t) = \int_{t-T}^{t} f(x)\,dx \qquad , t \geq T + x_0$$

Dabei ist T das Integrationsintervall, f(x) die zu integrierende Funktion, MInt(t) das gleitende Integral, x0 der x-Offset der zu integrierenden Funktion. Der erste Wert des gleitenden Integrals liegt für t = T + x0 vor, d. h. der erzeugte Datensatz beginnt etwas später als der übergebene. Dies äußert sich in einem um T erhöhten x-Offset.

Abgesehen von der enthaltenen Multiplikation mit der Abtastzeit wirkt das gleitende Integral wie ein Filter. Es glättet die Funktion, ist also ein Tiefpassfilter. Die Gewichtsfunktion ist rechteckig. Die oben genannte Erhöhung des x-Offset hat zur Folge, dass der Einschwingvorgang unterdrückt wird, indem er nicht berechnet wird.

- Die y-Einheit des erzeugten Datensatzes ist das Produkt von x- und y-Einheit des übergebenen Datensatzes.
- Die Breite des Integrationsintervalls muss kleiner sein als die Dauer des zu integrierenden Datensatzes, so dass der erzeugte Datensatz mindestens die Länge 2 hat. Ansonsten kann die Funktion Int() benutzt werden.

Beispiele:

```
NDp_real = MInt(NDp, 0.020 's') / 0.020 's'
```

Ein Datensatz, der die Momentanleistung p einer Last am Netz darstellt, wird benutzt, um die Wirkleistung zu berechnen. Die Periodendauer am Netz beträgt 20ms. Um einen Zahlenwert für die Wirkleistung zu erhalten, kann die Funktion "Wert" benutzt werden.

Siehe auch:

Int, Sum, MvSum, Smo, Hyst

Mirror

Spiegelung eines Datensatzes
Alternativer Name: **Spieg**

Deklaration:

```
Mirror ( Daten ) -> Gespiegelt
```

Parameter:

Daten	Zu spiegelnder Datensatz. Erlaubte Typen: [ND],[XY].
Gespiegelt	Gespiegelter Datensatz

Beschreibung:

Der übergebene Datensatz wird gespiegelt. Dabei befindet sich die Spiegelachse in der Mitte des Datensatzes. Der gespiegelte Datensatz wird wie folgt erzeugt: Der erste Wert des Datensatzes wird mit dem letzten Wert des Datensatzes vertauscht. Der zweite Wert des Datensatzes wird mit dem vorletzten Wert des Datensatzes vertauscht usw.

Bei XY-Daten werden sowohl die X- als auch die Y-Werte gespiegelt, es ändert sich damit nur die Reihenfolge der XY-Paare im Datensatz, die grafische Darstellung bleibt unverändert.

Für eine 'grafische' Spiegelung eines XY-Datensatzes können Sie folgende Formelzeile verwenden:

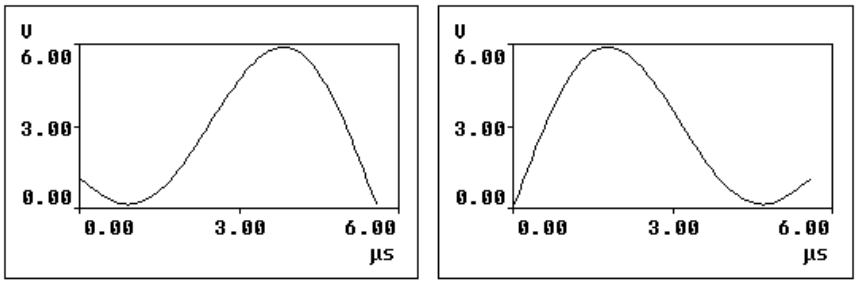
```
Quelle_Gespiegelt = XYof(min(Quelle.X)+ max(Quelle.X)-Quelle.X, Quelle.Y)
```

Wird die Funktion Mirror() wieder auf den gespiegelten Datensatz angewendet, wird wieder der Ausgangsdatensatz erzeugt.

Beispiele:

```
NDmirror = Mirror(NDdaten)
```

Der in der linken Abbildung dargestellte Datensatz NDdaten soll gespiegelt werden. Als Ergebnis der Spiegelung wird in imc FAMOS der in der rechten Abbildung dargestellte Datensatz NDspieg ausgegeben.



Siehe auch:

Sort

Mod

Modulo (Divisionsrest)

Deklaration:

```
Mod ( Zähler, Nenner ) -> Ergebnis
```

Parameter:

Zähler	Zähler. Erlaubte Typen: [ND],[XY].
Nenner	Nenner. Erlaubte Typen: [ND].
Ergebnis	Divisionsrest (modulo) aus Zähler und Nenner.

Beschreibung:

Die Funktion Mod (modulo) bestimmt den Divisionsrest. Der Nenner braucht nicht ganzzahlig zu sein. Die Funktion arbeitet folgendermaßen:

Die Zahlen Zähler und Nenner werden positiv gemacht. Dann wird der Nenner vom Zähler so oft abgezogen (subtrahiert), wie das Ergebnis nicht Null unterschreitet. Das Vorzeichen des Ergebnisses wird dann auf das Vorzeichen des Zählers gesetzt.

Bei Datensätzen wird die modulo-Funktion auf jeden Punkt angewendet. Sind beide Parameter der Funktion Mod() normale Datensätze, werden die Elemente punktweise verknüpft, ungeachtet der zugehörigen x-Koordinaten. Sind beide Parameter von verschiedenem Typ, wird der Einzelwert auf jeden Punkt des normalen Datensatzes angewendet.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Anmerkungen

- Die Einheit der erzeugten Daten ist der Quotient der Einheiten des 1. und 2. Parameters.
- Wie bei einer Division darf der 2. Parameter nicht Null sein. Gegebenfalls wird eine entsprechende Warnung erzeugt.
- Haben die übergebenen Datensätze unterschiedliche Längen, entspricht die Länge des Ergebnisses der des kürzeren Datensatzes.

```
EWrest = Mod(EWzähler, EWnenner)
```

Zähler	Nenner	Mod
17	2	1
18	2	0
17	-2	1
-17	2	-1
-17	-2	-1
8.5	3.5	1.5
7.5	3.5	0.5
7.0	3.5	0

Beispiele:

Die Phase eines linearen Filters 10.Ordnung hat die Werte -900° ... 0°. Da Phasen zwischen -360° und 0° anschaulicher sind, eine Phase aber nicht verändert wird, wenn Vielfache von 360° addiert werden, ist eine Anwendung der Funktion Mod() angebracht:

```
NDphase360 = Mod(NDphase, 360)
```

Siehe auch:

/(Division), PhaseMod, Floor

Monoflop

Verfügbar ab: Professional Edition

Monoflop

Deklaration:

```
Monoflop ( Daten, Impulsdauer [, Typ] [, Start] [, Richtung] [, Logik] ) -> Ergebnis
```

Parameter:

Daten	Eingangsdaten
Impulsdauer	Impulsdauer in x-Einheiten, wird auf ganze Vielfache der Abtastzeit gerundet
Typ	Retriggerbar? (optional , Standardwert: "no retrig")
	"no retrig" : Nicht retriggerbar: Steigende Flanken werden erst nach dem Ende eines Impulses wieder ausgewertet.
	"retrig" : Retriggerbar. Steigende Flanken während des erzeugten Impulses verlängern ihn.
	"1 retrig" : retriggerbar durch Wert 1: Wenn einmal ein Puls erzeugt wird, verlängert jede 1 im Eingangssignal den erzeugten Impuls.
Start	Deutung Startwert=1? (optional , Standardwert: "const")
	"const" : keine Flanke. Wirkt, als wären die Werte vor dem 1. Messwert alle auf 1.
	"0-1" : Übergang von 0 auf 1: Wirkt, als wären die Werte vor dem 1. Messwert alle auf null.
Richtung	Richtung, vorwärts oder rückwärts (optional , Standardwert: "")
	"" : Regulär: Das Signal wird in regulärer Richtung vorwärts, also von vorn nach hinten durchlaufen
	"reverse" : Rückwärts: Das Signal wird in umgekehrter Richtung, also von hinten nach vorn durchlaufen.
Logik	Logik, regulär oder invertiert (optional , Standardwert: "")
	"" : Regulär: Pulse bestehen aus 1, Zwischenräume aus null.
	"invert" : Invertiert: Pulse bestehen aus null, Zwischenräume aus 1. Der Eingang wird invertiert gedeutet, das Ergebnis wird invertiert.
Ergebnis	Ergebnis

Beschreibung:

Das Monoflop gibt bei einer steigenden Flanke (einem Übergang von null auf nicht null) einen Impuls von definierter Dauer ab (Wert 1 während Impuls, sonst null).

Sind die Eingangsdaten null, wird das als logisch null (false, low) gedeutet. Sonst als logisch 1 (true, high).

Das Monoflop realisiert eine Haltezeit.

In einfacher und typischer Anwendung wird die Funktion mit 2 Parametern aufgerufen.

Die Eingangsdaten können Events und Segmente haben.

Bei Durchlauf in umgekehrter Richtung mit Richtung = reverse wird das Signalende zum Start. Pulse werden nicht nach hinten verlängert, sondern nach vorn. Das Verhalten ist nicht mehr kausal. Z.B. für Anwendungen, in denen Pulse bestimmte Ereignisse andeuten und um diese Ereignisse herum ein Bereich abzustecken ist, der nicht immer nur dahinter, sondern auch mal davor liegt.

Beispiele:

Standardanwendung: Nicht retriggerbares Monoflop, Impulsdauer = 3s, Abtastzeit = 1s

```
MF = Monoflop ( data, 3 )
; in = 0 1 0 0 0
;out = 0 1 1 1 0

; in = 0 1 0 1 0
;out = 0 1 1 1 0

; in = 1 1 0 0 0
;out = 0 0 0 0 0
```

Retriggerbares Monoflop, Impulsdauer = 3s, Abtastzeit = 1s

```
MF = Monoflop ( data, 3, "retrig" )
; in = 0 1 0 1 0 0 0 0
;out = 0 1 1 1 1 1 0 0
```

```
; in = 0 1 1 0 0
;out = 0 1 1 1 0

; in = 1 1 0 0 0
;out = 0 0 0 0 0
```

Durch Zustand 1 retriggerbares Monoflop, Impulsdauer = 3s, Abtastzeit = 1s

```
MF = Monoflop ( data, 3, "1 retrig"  )
; in = 0 1 1 0 0 0 0 0
;out = 0 1 1 1 1 0 0 0

; in = 0 1 1 0 1 1 0 0 0
;out = 0 1 1 1 1 1 1 1 0

; in = 1 1 0 0 0
;out = 0 0 0 0 0
```

Nicht retriggerbares Monoflop, Impulsdauer = 3s, Abtastzeit = 1s. Eine 1 zu Beginn soll als Flanke gedeutet werden.

```
MF = Monoflop ( data, 3, "no retrig", "0-1" )
; in = 0 1 0 0 0
;out = 0 1 1 1 0

; in = 0 1 0 1 0
;out = 0 1 1 1 0

; in = 1 1 0 0 0
;out = 1 1 1 0 0
```

Nicht retriggerbares Monoflop, Impulsdauer = 3s, Abtastzeit = 1s. Inverse Logik

```
MF = Monoflop ( data, 3, "no retrig", "const", "", "invert" )
; in = 1 0 1 1 1
;out = 1 0 0 0 1

; in = 1 0 1 0 1
;out = 1 0 0 0 1
```

Nicht retriggerbares Monoflop, Impulsdauer = 3s, Abtastzeit = 1s. Rückwärts

```
MF = Monoflop ( data, 3, "no retrig", "const", "reverse" )
; in = 0 0 1 1 0 0 0 0 1 0 1 0 1 0
;out = 0 1 1 1 0 0 1 1 1 0 1 1 1 0
```

Siehe auch:

Flipflop

MSave

Speichern von mehreren Datensätzen als mehrspaltige Ascii-Datei.
Alternativer Name: **MSichern**

Die Funktion ist veraltet. Zum Export von mehrspaltigen ASCII-Dateien ist die Definition einer ASCII-Exportvorlage und die Verwendung der Funktionen FileSave() oder FileOpenASCII2() meist die bequemere und flexiblere Alternative.

Deklaration:

```
MSave ( EwAuftrag, EwOption, Daten, TxFormatOderDatei ) -> EwFehler
```

Parameter:

EwAuftrag	Auftrag
	1 : Neue Ausgabe vorbereiten. Muss zu Beginn jeder Speicherung einmal ausgeführt werden. Alle anderen Parameter müssen zwar vom richtigen Typ sein, werden aber ignoriert.
	2 : Einen Datensatz zur Datei hinzufügen. [EwOption] muss vom richtigen Typ sein, wird aber ignoriert. [Daten] ist der Datensatz, der hinzugefügt wird. [TxFormatOderDatei] beschreibt das Ausgabeformat, siehe Beschreibung. Geben Sie mit diesem Kommando nacheinander alle Datensätze an, die in der Datei spaltenweise gespeichert werden sollen.
	3 : Die Datei wird abschließend gespeichert. [EwOption] wählt verschiedene Modi. [Daten] muss vom richtigen Typ sein, wird aber ignoriert. [TxFormatOderDatei] gibt den Dateinamen an.
EwOption	Modus-Auswahl für [EwAuftrag] = 3 (Datei speichern). Ansonsten ignoriert und z.B. auf 0 zu setzen. Siehe Beschreibung.
Daten	Zu speichernde Variable. Datensatz oder Einzelwert.
TxFormatOderDatei	Für [EwAuftrag] = 2 Vorgabe des Zahlenformats. Für [EwAuftrag] = 3 der zu verwendende Dateiname. Ansonsten ignoriert und z.B. auf "" zu setzen.
EwFehler	Fehlercode
	0 : Die Datei wurde erfolgreich gespeichert.
	1 : Nicht genügend Arbeitsspeicher vorhanden.
	2 : Fehler beim Anlegen der Datei. Bitte prüfen Sie den angegebenen Dateinamen.
	3 : Es sind maximal 512 Spalten erlaubt.
	4 : Die Datei konnte nicht geschrieben werden. Möglicherweise reicht der Platz auf dem Datenträger nicht aus.
	5 : Falsche Angabe in [TxFormatOderDatei] bei [EwAuftrag] = 2.

Beschreibung:

Bedeutung des Parameters [TxFormatOderDatei] für [EwAuftrag] = 1:

"fV.N"	Fließkomma: V=Vorkommastellen ohne Vorzeichen, N=Nachkommast. (Default = 6), zusammen <= 15
"f0V.N"	wie "fV.N", aber mit führenden Nullen vor dem Komma
"eN"	Exponential: N=Nachkommastellen <= 15 (Default = 6)
"e0N"	wie "eN", aber mit führenden Nullen im 3-stelligen Exponent
"xS"	Hexadezimal: S=Stellenzahl <= 8
"x0S"	wie "xS", aber mit führenden Nullen
"aN"	automatisch Fließkomma/Exponential: N=Nachkommastellen (Default = 6)
"", " "	wirkt wie das frühere Festformat "f4.4"

Bedeutung des Parameters [EwOption] für [EwAuftrag] = 2:

Addieren Sie:	
0:	Keine Spalte mit Zeit erzeugen
1:	Erste Spalte ist Zeit des ersten Datensatzes in Sekunden.
2:	wie 1, aber in Milli-Sekunden
3:	wie 1, aber in Micro-Sekunden
4:	wie 1, aber in Nano-Sekunden
5:	Die Zeit wird in Std:Min:Sek ausgegeben, Uhrzeit absolut. Bei Einzelwerten ist die Startzeit die Zeit der Dateierzeugung.

6:	Die Zeit wird in Std:Min:Sek ausgegeben, Start mit 00:00:00
7:	Zeit und Datum im eingestellten Länderformat. Bei Einzelwerten ist die Startzeit die Zeit der Dateierzeugung.
8:	wie 5, Startzeit jedoch Zeit der Dateierzeugung
9:	wie 7, Startzeit und -datum jedoch Zeit der Dateierzeugung
Addieren Sie:	
0:	Die Datei wird neu erzeugt
10:	Die Daten werden an eine bestehende Datei angehängt. Falls die Datei nicht existiert, wird sie erzeugt.
Addieren Sie:	
0:	Die Spalten werden durch Tabulator-Zeichen getrennt.
100:	Die Spalten werden durch Leerzeichen getrennt.
900:	Die Spaltentrennung ist durch die Systemsteuerung vorgegeben.
Addieren Sie:	
0:	Als Dezimalzeichen wird ein Punkt verwendet.
1000:	Als Dezimalzeichen wird ein Komma verwendet. (Excel)
9000:	Das Dezimalzeichen ist durch die Systemsteuerung vorgegeben.
Addieren Sie:	
0:	Die Datensätze sind äquidistant.
10000:	Der erste Datensatz enthält X-Koordinaten für die übrigen Spalten
Addieren Sie:	(nur bei formatierter Zeitausgabe, Option + 5,6 oder 7):
0:	Sekunden ohne Nachkommastellen
100000:	Anzahl der Sekunden-Nachkommastellen = 1
..	
900000:	Anzahl der Sekunden-Nachkommastellen = 9

Falls kein voller Dateiname angegeben ist, wird das aktuelle Standardverzeichnis verwendet. Das aktuelle Standardverzeichnis wird beim Starten von FAMOS auf die Voreinstellung (Dialog 'Optionen'/ 'Verzeichnisse') gesetzt. Es kann mit der Funktion SetOption() umgesetzt werden.

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Beispiele:

Sichert die Datensätze a, b, c und d mit Zeit in Milli-Sekunden. Die Spalten werden durch Tabulator getrennt. In der ersten Zeile der Datei wird eine kurze Beschreibung ausgegeben. Es wird ein Dezimalpunkt verwendet.

Der Name der Datei ist "C:\Test.txt".

```
fh = FileOpenASCII("C:\Test.txt", 1)
IF fh > 0
   err = FileLineWrite(fh, "1.Zeit/ms, 2.U/mV, 3.I/mA", 0)
   err = FileClose(fh )
END
MSave(1, 0, 0, "")
MSave(2, 0, a,"f3.2")   ; z.B. " 123.12" oder "12345.12" oder "  1.12"
MSave(2, 0, b,"E3")      ; z.B. " 1.123E+001"
MSave(2, 0, c,"e5")      ; z.B. "-1.12345e+001"
MSave(2, 0, d,"A3")      ; z.B. "1.123E+005" oder "-1123.456"
MSave(3, 12, 0, "C:\Test.txt") ;Zeit in ms, An Datei anhängen
```

Siehe auch:

FileOpenASCII2, FileSave, FileLineWrite

Mult

Zeit- bzw. x-richtige Multiplikation

Deklaration:

```
Mult ( Faktor1, Faktor2, EwOption ) -> Produkt
```

Parameter:

Faktor1	Erster Parameter, Faktor. Erlaubte Typen: [ND],[XY].
Faktor2	Zweiter Parameter, Faktor. Erlaubte Typen: [ND],[XY].
EwOption	Option
	0 : Die Triggerzeit der beiden Summanden wird ignoriert
	1 : Zeitrichtige Überlagerung mit Berücksichtigung der Triggerzeit
Produkt	Produkt, Ergebnis der Multiplikation. [XY]

Beschreibung:

Zwei Datensätze werden zeit- bzw. x-richtig multipliziert, d. h. es werden immer 2 y-Werte multipliziert, die die gleiche Zeit bzw. den gleichen x-Wert besitzen.

Das Ergebnis ist nur für den x-Bereich definiert, den die beiden Parameter-Datensätze gemeinsam haben. In diesem Bereich wird überall dort ein Ergebnis-Wert berechnet, wo mindestens einer der beiden Datensätze eine Stützstelle besitzt. Wenn an dieser Stelle der andere Datensatz keine Stützstelle besitzt, wird dieser dort linear interpoliert.

Die x-Spur beider Parameter-Datensätze muss monoton steigend sein, d.h. die x-Koordinaten müssen kontinuierlich wachsen.

Der Operator * (Multiplikation) führt dagegen eine punktweise Multiplikation von Datensätzen aus.

Beispiele:

Zwei Kanäle werden zwischen 11 und 13 Uhr bzw. 12 und 14 Uhr aufgenommen.

```
power12_13h = Mult(voltage11_13h, current12_14h, 1)
```

Eine zeitrichtige Multiplikation der beiden Datensätze mit Berücksichtigung der Triggerzeit wird ausgeführt. Das Ergebnis ist zwischen 12 und 13 Uhr definiert.

Siehe auch:

*(Multiplikation), Add, Sub, Div, Append

MvMax

Gleitendes Maximum mit Nachabtastung

Deklaration:

```
MvMax ( Daten, EwFenster, EwReduktion [, EwKompatibilität] ) -> GleitMax
```

Parameter:

Daten	Zu mittelnder Datensatz. [ND]
EwFenster	Breite des Mittelungsintervalls (in x-Einheiten).
EwReduktion	Breite des Reduktionsintervalls (in x-Einheiten).
EwKompatibilität	(optional , Standardwert: 0)
	0 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 10 begrenzt. Die Berechnung ist damit kompatibel zu FAMOS-Versionen <=7.4.
	1 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 1000 begrenzt.
GleitMax	Gleitendes Maximum

Beschreibung:

Jeder berechnete Wert ist das absolute Maximum über die Breite [EwFenster].

[EwReduktion] gibt die Breite des Reduktionsintervalls an. Für jedes solches Intervall wird genau ein Ergebniswert erzeugt. Die x-Koordinate dieses Ergebniswertes ist der Intervallbeginn. Somit ist eine Darstellung des Ergebnisses im Kurvenfenster als 'Treppenstufen' sinnvoll.

Mittelungsintervall und Reduktionsintervall sollten Vielfache der Abtastzeit sein. Andernfalls wird gerundet, da immer über eine ganze Anzahl von Messwerten gerechnet werden muss.

Außerdem muss zwischen der Zahl der Messwerte in [EwFenster] und [EwReduktion] ein ganzzahliges Verhältnis von 1:1, 2:1 .. **10**:1 (für EwKompatibilität=0) bzw. **1000**:1 (für EwKompatibilität=1) bestehen. Wenn dies nicht der Fall ist, werden beide Intervalle entsprechend angepasst.

Wenn [EwFenster] und [EwReduktion] ungleich sind, stehen am Anfang nicht genug Werte zur Verfügung. Es werden dann die jeweils zur Verfügung stehenden Eingangswerte verrechnet.

Beispiele:

Es wird alle 10 s das Maximum der letzten 10 s berechnet:

```
resultMax = MvMax(timeData, 10, 10)
```

Es wird alle 10 s das Maximum der letzten 20 s berechnet:

```
resultMax = MvMax(timeData, 20, 10)
```

Es wird für jeden 5. Punkt das Maximum der letzten 10 Punkte berechnet:

```
_xdelta = xdel?(timeData)
resultMax = MvMax(timeData, 10*_xdelta, 5*_xdelta)
```

Siehe auch:

Max, MvMin, MvMean, MvRMS

MvMean

Gleitendes Mittelwert mit Nachabtastung

Deklaration:

```
MvMean ( Daten, EwFenster, EwReduktion [, EwKompatibilität] ) -> GleitMittel
```

Parameter:

Daten	Zu mittelnder Datensatz. [ND]
EwFenster	Breite des Mittelungsintervalls (in x-Einheiten).
EwReduktion	Breite des Reduktionsintervalls (in x-Einheiten).
EwKompatibilität	(optional , Standardwert: 0)
	0 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 10 begrenzt. Die Berechnung ist damit kompatibel zu FAMOS-Versionen <=7.4.
	1 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 1000 begrenzt.
GleitMittel	Gleitender Mittelwert

Beschreibung:

Jeder berechnete Wert ist das arithmetische Mittel über die Breite [EwFenster].

[EwReduktion] gibt die Breite des Reduktionsintervalls an. Für jedes solches Intervall wird genau ein Ergebniswert erzeugt. Die x-Koordinate dieses Ergebniswertes ist der Intervallbeginn. Somit ist eine Darstellung des Ergebnisses im Kurvenfenster als 'Treppenstufen' sinnvoll.

Mittelungsintervall und Reduktionsintervall sollten Vielfache der Abtastzeit sein. Andernfalls wird gerundet, da immer über eine ganze Anzahl von Messwerten gerechnet werden muss.

Außerdem muss zwischen der Zahl der Messwerte in [EwFenster] und [EwReduktion] ein ganzzahliges Verhältnis von 1:1, 2:1 .. **10**:1 (für EwKompatibilität=0) bzw. **1000**:1 (für EwKompatibilität=1) bestehen. Wenn dies nicht der Fall ist, werden beide Intervalle entsprechend angepasst.

Wenn [EwFenster] und [EwReduktion] ungleich sind, stehen am Anfang nicht genug Werte zur Verfügung. Es werden dann die jeweils zur Verfügung stehenden Eingangswerte verrechnet.

Beispiele:

Es wird alle 10 s der Mittelwert der letzten 10 s berechnet:

```
resultMean = MvMean(timeData, 10, 10)
```

Es wird alle 10 s der Mittelwert der letzten 20 s berechnet:

```
resultMean = MvMean(timeData, 20, 10)
```

Es wird für jeden 5. Punkt der Mittelwert der letzten 10 Punkte berechnet:

```
_xdelta = xdel?(timeData)
resultMean = MvMean(timeData, 10*_xdelta, 5*_xdelta)
```

Siehe auch:

Mean, MvMin, MvMax, MvRMS, Median

MvMin

Gleitendes Minimum mit Nachabtastung

Deklaration:

```
MvMin ( Daten, EwFenster, EwReduktion [, EwKompatibilität] ) -> GleitMin
```

Parameter:

Daten	Zu mittelnder Datensatz. [ND]
EwFenster	Breite des Mittelungsintervalls (in x-Einheiten).
EwReduktion	Breite des Reduktionsintervalls (in x-Einheiten).
EwKompatibilität	(optional , Standardwert: 0)
	0 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 10 begrenzt. Die Berechnung ist damit kompatibel zu FAMOS-Versionen <=7.4.
	1 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 1000 begrenzt.
GleitMin	Gleitendes Minimum

Beschreibung:

Jeder berechnete Wert ist das absolute Minimum über die Breite [EwFenster].

[EwReduktion] gibt die Breite des Reduktionsintervalls an. Für jedes solches Intervall wird genau ein Ergebniswert erzeugt. Die x-Koordinate dieses Ergebniswertes ist der Intervallbeginn. Somit ist eine Darstellung des Ergebnisses im Kurvenfenster als 'Treppenstufen' sinnvoll.

Mittelungsintervall und Reduktionsintervall sollten Vielfache der Abtastzeit sein. Andernfalls wird gerundet, da immer über eine ganze Anzahl von Messwerten gerechnet werden muss.

Außerdem muss zwischen der Zahl der Messwerte in [EwFenster] und [EwReduktion] ein ganzzahliges Verhältnis von 1:1, 2:1 .. **10**:1 (für EwKompatibilität=0) bzw. **1000**:1 (für EwKompatibilität=1) bestehen. Wenn dies nicht der Fall ist, werden beide Intervalle entsprechend angepasst.

Wenn [EwFenster] und [EwReduktion] ungleich sind, stehen am Anfang nicht genug Werte zur Verfügung. Es werden dann die jeweils zur Verfügung stehenden Eingangswerte verrechnet.

Beispiele:

Es wird alle 10 s das Minimum der letzten 10 s berechnet:

```
resultMin = MvMin(timeData, 10, 10)
```

Es wird alle 10 s das Minimum der letzten 20 s berechnet:

```
resultMin = MvMin(timeData, 20, 10)
```

Es wird für jeden 5. Punkt das Minimum der letzten 10 Punkte berechnet:

```
_xdelta = xdel?(timeData)
resultMin = MvMin(timeData, 10*_xdelta, 5*_xdelta)
```

Siehe auch:

Min, MvMax, MvMean, MvRMS

MvRMS

Gleitender Effektivwert mit gleichgewichteter Mittelung

Deklaration:

```
MvRMS ( Daten, EwFenster, EwReduktion [, EwKompatibilität] ) -> GleitEff
```

Parameter:

Daten	Zu mittelnder Datensatz. [ND]
EwFenster	Breite des Mittelungsintervalls (in x-Einheiten).
EwReduktion	Breite des Reduktionsintervalls (in x-Einheiten).
EwKompatibilität	(optional , Standardwert: 0)
	0 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 10 begrenzt. Die Berechnung ist damit kompatibel zu FAMOS-Versionen <=7.4.
	1 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 1000 begrenzt.
GleitEff	Gleitender Effektivwert

Beschreibung:

Es wird der gleitende Effektivwert mit gleichmäßig gewichteter Mittelung und anschließender Nachabtastung berechnet.

Jeder berechnete Wert ist der Effektivwert über die Breite [EwFenster].

[EwReduktion] gibt die Breite des Reduktionsintervalls an. Für jedes solches Intervall wird genau ein Ergebniswert erzeugt. Die x-Koordinate dieses Ergebniswertes ist der Intervallbeginn. Somit ist eine Darstellung des Ergebnisses im Kurvenfenster als 'Treppenstufen' sinnvoll.

Mittelungsintervall und Reduktionsintervall sollten Vielfache der Abtastzeit sein. Andernfalls wird gerundet, da immer über eine ganze Anzahl von Messwerten gerechnet werden muss.

Außerdem muss zwischen der Zahl der Messwerte in [EwFenster] und [EwReduktion] ein ganzzahliges Verhältnis von 1:1, 2:1 .. **10**:1 (für EwKompatibilität=0) bzw. **1000**:1 (für EwKompatibilität=1) bestehen. Wenn dies nicht der Fall ist, werden beide Intervalle entsprechend angepasst.

Wenn [EwFenster] und [EwReduktion] ungleich sind, stehen am Anfang nicht genug Werte zur Verfügung. Es werden dann die jeweils zur Verfügung stehenden Eingangswerte verrechnet.

Beispiele:

Es wird alle 10 s der Effektivwert der letzten 10 s berechnet:

```
resultRMS = MvRMS(timeData, 10, 10)
```

Es wird alle 10 s der Effektivwert der letzten 20 s berechnet:

```
resultRMS = MvRMS(timeData, 20, 10)
```

Es wird für jeden 5. Punkt der Effektivwert der letzten 10 Punkte berechnet:

```
_xdelta = xdel?(timeData)
resultRMS = MvRMS(timeData, 10*_xdelta, 5*_xdelta)
```

Siehe auch:

RMS, ExpoRMS, MvMin, MvMax, MvStDev

MvStDev

Gleitende Standardabweichung mit Nachabtastung

Deklaration:

```
MvStDev ( Daten, EwFenster, EwReduktion [, EwKompatibilität] ) -> GleitStdAbw
```

Parameter:

Daten	Zu mittelnder Datensatz. [ND]
EwFenster	Breite des Berechnungsintervalls (in x-Einheiten).
EwReduktion	Breite des Reduktionsintervalls (in x-Einheiten).
EwKompatibilität	(optional , Standardwert: 0)
	0 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 10 begrenzt. Die Berechnung ist damit kompatibel zu FAMOS-Versionen <=7.4.
	1 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 1000 begrenzt.
GleitStdAbw	Gleitende Standardabweichung

Beschreibung:

Jeder berechnete Wert ist die Standardabweichung über die Breite [EwFenster].

[EwReduktion] gibt die Breite des Reduktionsintervalls an. Für jedes solches Intervall wird genau ein Ergebniswert erzeugt. Die x-Koordinate dieses Ergebniswertes ist der Intervallbeginn. Somit ist eine Darstellung des Ergebnisses im Kurvenfenster als 'Treppenstufen' sinnvoll.

Berechnungsintervall und Reduktionsintervall sollten Vielfache der Abtastzeit sein. Andernfalls wird gerundet, da immer über eine ganze Anzahl von Messwerten gerechnet werden muss.

Außerdem muss zwischen der Zahl der Messwerte in [EwFenster] und [EwReduktion] ein ganzzahliges Verhältnis von 1:1, 2:1 .. **10**:1 (für EwKompatibilität=0) bzw. **1000**:1 (für EwKompatibilität=1) bestehen. Wenn dies nicht der Fall ist, werden beide Intervalle entsprechend angepasst.

Wenn [EwFenster] und [EwReduktion] ungleich sind, stehen am Anfang nicht genug Werte zur Verfügung. Es werden dann die jeweils zur Verfügung stehenden Eingangswerte verrechnet.

Beispiele:

Es wird alle 10 s die Standardabweichung der letzten 10 s berechnet:

```
resultStDev = MvStDev(timeData, 10, 10)
```

Es wird alle 10 s die Standardabweichung der letzten 20 s berechnet:

```
resultStDev = MvStDev(timeData, 20, 10)
```

Es wird für jeden 5. Punkt die Standardabweichung der letzten 10 Punkte berechnet:

```
_xdelta = xdel?(timeData)
resultStDev = MvStDev(timeData, 10*_xdelta, 5*_xdelta)
```

Siehe auch:

StDev, MvMean, MvRMS, Stat

MvSum

Gleitende Summenbildung mit Nachabtastung

Deklaration:

```
MvSum ( Daten, EwFenster, EwReduktion [, EwKompatibilität] ) -> GleitSumme
```

Parameter:

Daten	Zu summierender Datensatz. [ND]
EwFenster	Breite des Berechnungsintervalls (in x-Einheiten).
EwReduktion	Breite des Reduktionsintervalls (in x-Einheiten).
EwKompatibilität	(optional , Standardwert: 0)
	0 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 10 begrenzt. Die Berechnung ist damit kompatibel zu FAMOS-Versionen <=7.4.
	1 : Das maximale Verhältnis von Fensterbreite zu Reduktionsintervall wird auf 1000 begrenzt.
GleitSumme	Gleitende Summe

Beschreibung:

Jeder berechnete Wert ist die Summe über die Breite [EwFenster].

[EwReduktion] gibt die Breite des Reduktionsintervalls an. Für jedes solches Intervall wird genau ein Ergebniswert erzeugt. Die x-Koordinate dieses Ergebniswertes ist der Intervallbeginn. Somit ist eine Darstellung des Ergebnisses im Kurvenfenster als 'Treppenstufen' sinnvoll.

Berechnungsintervall und Reduktionsintervall sollten Vielfache der Abtastzeit sein. Andernfalls wird gerundet, da immer über eine ganze Anzahl von Messwerten gerechnet werden muss.

Außerdem muss zwischen der Zahl der Messwerte in [EwFenster] und [EwReduktion] ein ganzzahliges Verhältnis von 1:1, 2:1 .. **10**:1 (für EwKompatibilität=0) bzw. **1000**:1 (für EwKompatibilität=1) bestehen. Wenn dies nicht der Fall ist, werden beide Intervalle entsprechend angepasst.

Wenn [EwFenster] und [EwReduktion] ungleich sind, stehen am Anfang nicht genug Werte zur Verfügung. Es werden dann die jeweils zur Verfügung stehenden Eingangswerte verrechnet.

Beispiele:

Es wird alle 10 s die Summe über die letzten 10 s berechnet:

```
resultSum = MvSum(timeData, 10, 10)
```

Es wird alle 10 s die Summe über die letzten 20 s berechnet:

```
resultSum = MvSum(timeData, 20, 10)
```

Es wird für jeden 5. Punkt die Summe über die letzten 10 Punkte berechnet:

```
_xdelta = xdel?(timeData)
resultSum = MvSum(timeData, 10*_xdelta, 5*_xdelta)
```

Siehe auch:

Sum, MvRMS, Int, GInt

Name?

Abfrage des Namens einer Variablen

Deklaration:

```
Name? ( Variable [, Aufbau] [, Format] ) -> TxName
```

Parameter:

Variable	Variable, deren Name ermittelt werden soll. Der Datentyp ist beliebig.
Aufbau	Aufbau des ermittelten Namens (optional , Standardwert: 0)
	0 : Kompletter Bezeichner, ggf. mit Gruppenname und Messung
	1 : Bezeichner ggf. mit Gruppenname, ohne Messung
	2 : Name ohne Gruppenname, ohne Messung
Format	Formatierung des ermittelten Namens. (optional , Standardwert: 0)
	0 : Standard
	1 : Erweiterte Syntax: Damit ein Variablenname in Formeln angegeben werden kann, muss er bestimmten Regeln gehorchen (z.B. keine Leerzeichen, erstes Zeichen keine Ziffer etc.). Falls dies nicht der Fall ist, muss der Name zusätzlich in geschweifte Klammern {...} eingeschlossen werden. Mit dieser Option wird der zurückgegebene Name um diese geschweiften Klammern ergänzt, falls nötig.
TxName	Name

Beschreibung:

Der Name der übergebenen Variablen wird ermittelt. Typische Anwendungsfälle für diese Funktion sind:

- Sequenzfunktionen mit Parameterübergabe per Referenz: Der lokale Parameter ist hier keine echte Variable, sondern nur ein Alias für die beim Aufruf übergebene Variable. Die Anwendung der Funktion auf einen solchen Referenz-Parameter liefert dann den Namen der referenzierten Variable.
- Klassische Sequenzen: Aus dem Platzhalter-Namen (z.B. PA1) kann der Name der beim Sequenzaufruf übergebenen Variablen bestimmt werden.
- Datengruppen: Ermittlung eines Kanalnamens aus dem Kanalindex. Leistungsfähigere Alternative zur Funktion GrChanName?().

Der erste Parameter muss eine direkt angegebene Variable ohne weitere Indizierung (wie z.B. Komponente, Event-Index) sein. Bei Gruppen ist die Angabe des Kanal-Index erlaubt.

Beispiele:

Die folgende Sequenzfunktion zeigt den übergebenen Datensatz in einem freifliegenden Kurvenfenster an. Der Name des Datensatzes wird in der Kopfzeile angezeigt.

```
!ShowCurve(par)
   ; par: [ND, Referenz]
   CwNewWindow(par, "show")
   CwNewChannel("append new axis", par)
   CwDisplaySet("header.count", 1)
   CwDisplaySet("header.text", Name?(par))
```

In einer Datengruppe [input] werden alle Kanäle aufgezählt. Alle Kanäle, deren Maximum größer als 2 ist, werden in eine neue Datengruppe [output] kopiert.

```
limit = 2
count = GrChanNum?(input)
FOR i = 1 TO count
   IF max(input:[i]) > limit
      name = Name?(input:[i], 2, 1)
      output:<name> = input:[i]
   END
END
```

Siehe auch:

Comm?, MeasurementName?, GrChanName?, VarGetName?

NorthCorrection

Verfügbar ab: Professional Edition

Nordsprungkorrektur innerhalb eines Fenster, damit eine Mittelung über Windrichtungen oder Winkel bzw. Phasen sinnvoll erfolgen kann.

Deklaration:

```
NorthCorrection ( Daten, Fenster ) -> Ergebnis
```

Parameter:

Daten	Eingangsdaten
Fenster	Breite des Mittelungsintervalls in x-Einheiten, wird auf ganze Vielfache der Abtastzeit gerundet
Ergebnis	Ergebnis

Beschreibung:

Die Funktion NorthCorrection() führt die Nordsprungkorrektur nach der Addiermethode durch.

Sie verhindert einen Nordsprung, d.h. einen Sprung von 360° auf 0°, im Mittelungsintervall. Bei Werten, die um 360° herum schwanken und damit auch wieder um 0° herum liegen können, wird so ein falscher Mittelwert von 180° vermieden.

Das Ergebnis der Mittelung kann jenseits des Windrosenbereichs von 0°...360° liegen, z.B. 365°. Die Funktion PhaseMod() führt das Ergebnis wieder in den Windrosenbereich (0°...360°) zurück, z.B. von 365° auf 5°.

Bei dem Verfahren wird vorausgesetzt, dass die Windrichtung innerhalb des Mittelungsintervalls nicht um mehr als 180 Grad dreht.

Sinnvoll angewendet werden kann NorthCorrection() nur in Verbindung mit Mittelungen o.ä. und anschließender PhaseMod().

Die Eingangsdaten haben die Einheit Grad und liegen im typisch im Wertebereich -360 bis 360 Grad oder auch etwas darüber hinaus.

Die Funktion addiert bzw. subtrahiert 360 Grad an den nötigen Stellen.

Die Eingangsdaten sind äquidistant.

Beispiele:

Berechnung der gleitend über je 10s gemittelten Windrichtung

```
NC = NorthCorrection( Channel, 10 )
NC_Mean = MvMean( NC, 10, 10 )
Wind = PhaseMod( NC_Mean )
```

Siehe auch:

PhaseMod, PhaseContinuous

NOT

Logische Invertierung

Deklaration:

```
NOT ( Operand ) -> NullOderEins
```

Parameter:

Operand	Zu invertierender Einzelwert bzw. Datensatz.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

Logische Invertierung einer Zahl. Das Ergebnis ist 1, wenn der Operand 0 ist. Andernfalls ist das Ergebnis 0.

Die Funktion kann auf Einzelwerte und auf Datensätze angewendet werden. Bei Datensätzen erfolgt eine punktweise Invertierung.

Beispiele:

Die beiden folgenden Ausdrücke sind äquivalent:

```
NOT(A) AND NOT(B)
NOT(A OR B)
```

Invertierung eines digitalen Datensatzes.

```
Result = NOT(DigChannel1)
```

Siehe auch:

AND, OR, XOR

OctA

Terz-Analyse
Alternativer Name: **TerzA**

Deklaration:

```
OctA ( Daten, EwFrequenzUnten, EwFrequenzOben ) -> TerzDaten
```

Parameter:

Daten	Der Datensatz, der einer Terz-Analyse zu unterziehen ist.
EwFrequenzUnten	Das untere Ende des Frequenz-Bereichs
EwFrequenzOben	Das obere Ende des Frequenz-Bereichs
TerzDaten	Ergebnis der Filterung

Beschreibung:

Diese Funktion kann die folgenden gängigen Frequenz-Band-Analysen durchführen: Terzen, Oktaven, 1/12 Oktaven, 1/24 Oktaven. Die Funktion führt gleichzeitig eine Zeit- und Frequenz-Bewertung aus. Alle Filterungen werden über digitale Filter durchgeführt. Die FFT wird nicht benutzt.

Bei der Analyse wird folgendermaßen vorgegangen:

Aus den Vorgaben für die Abtastzeit, die Bandbreite und die gewünschten Frequenz-Bänder werden digitale Filter entworfen.

Der Datensatz wird mit diesen digitalen Filtern gefiltert.

Die gefilterten Daten werden einer Zeitbewertung unterzogen. Bei der Zeitbewertung wird der gleitende Effektivwert mit zugrunde liegender exponentieller Mittelung gebildet. Die Zeitkonstante ist vorgebbar. Der gleitende Effektivwert wird auf folgende Weise berechnet: Das Signal wird quadriert, dann exponentiell gemittelt, dann wird wieder die Wurzel gezogen. Quadrierung, Mittelung und anschließende Radizierung sind die Merkmale einer Effektivwert-Bildung. Beim gewöhnlichen Effektivwert wird ein gleichmäßig gewichtetes Mittel aller Quadrate gebildet, während hier beim gleitenden Effektivwert eine zeitliche Bewertung durchgeführt wird. Die benutzte exponentielle Mittelung bewirkt ein "Vergessen". Sie kann als Filterung mit einem Tiefpass 1.Ordnung verstanden werden.

Der zeitbewertete Datensatz wird nun optional noch einmal nachabgetastet. Nur jeder n-te Wert wird weiterverarbeitet. Wenn eine starke Mittelung bei der Zeit-Bewertung benutzt wird, ist es i. a. nicht nötig, jeden Wert zu benutzen, da die benachbarten Werte sich kaum voneinander unterscheiden.

Das Ergebnis kann noch normiert werden, indem mit einem Bezugswert multipliziert wird. Der Bezugswert ist ein Wert, der z. B. 0 dB entspricht.

Nun erfolgt die Frequenz-Bewertung. Die Bewertungen linear, A, B, C und D sind möglich.

Die Funktion kann drei verschiedene Ergebnis-Varianten liefern:

Der zeitliche Verlauf eines Frequenz-Bandes wird bestimmt. Sie erhalten einen Datensatz über der Zeit.

Mehrere Frequenz-Bänder werden bestimmt, wobei über den gesamten Datensatz gemittelt wird. Sie erhalten einen Datensatz über der Frequenz.

Mehrere Frequenz-Bänder werden gleichzeitig über der Zeit ermittelt. Wenn z. B. die 3 Frequenzen f1, f2 und f3 über der Zeit ermittelt werden, dann nimmt der Ergebnis-Datensatz folgende Gestalt an: Zuerst stehen die Werte f1, f2 und f3 zur ersten Zeit, dann die Werte f1, f2, f3 zur zweiten Zeit usw.. Der Datensatz ist für das erste Set von Frequenzen (f1..f3) skaliert. Die Zeit zwischen den Abtastungen der Frequenz und auch die Anzahl der Frequenzen ist nicht im Datensatz enthalten und muss selbst verwaltet werden. Dieser Datentyp ist komplizierter zu handhaben und daher nur Spezial-Anwendungen vorbehalten! Siehe auch Kurvenfenster-Beschreibung des Perioden-Vergleichs. Dieser Datensatz kann nur über die Darstellung Perioden-Vergleich mit den Kurvenfenstern sinnvoll dargestellt werden. Eine einzelne Periode kann mit der mathematischen Funktion Perio() extrahiert werden. Der Datensatz kann als Matrix verstanden werden, wobei eine Zeile eine Periode ist und alle Zeilen direkt aneinandergereiht sind.

Um Beschleunigungen in Geschwindigkeiten oder Wege umzurechnen oder umgekehrt, kann das Signal noch integriert oder differenziert werden. Das ist nach der bereits durchgeführten Bandbegrenzung eine besonders günstige Stelle, da hochfrequentes Rauschen bereits entfernt ist (also Ableitung gut bestimmbar) und auch niederfrequente Driften und Offsets eliminiert sind (also Integration gut durchführbar). Das ist nicht in der DIN beschrieben, aber von hoher praktischer Bedeutung. Integrationen und Differentiationen werden durchgeführt, indem das digitale Filter für den Bandpass um Pol- bzw. Nullstellen bei der Frequenz Null erweitert wird.

Die zeitliche Veränderung der Amplitude eines Frequenz-Bandes ist i. a. ein stark schwankendes Signal. Um eine Aussage zu erhalten, wird dieses Signal gleichrichtet und gemittelt. Das angewendete Verfahren dazu ist die gleitende Effektivwert-Bildung. Folgende Zeitbewertungen sind wählbar:

=0	Keine Zeitbewertung
> 0	Sie geben selbst eine Zeitkonstante an. Die Zeitkonstante darf kleiner oder größer als die Abtastzeit sein.
-1 FAST	Die Zeitkonstante beträgt 0.125s.
-2 SLOW	Die Zeitkonstante beträgt 1s.

-3 Impuls	Bei ansteigender Amplitude beträgt die Zeitkonstante 35ms, bei abfallender Amplitude 1.5s. Damit werden impulsförmige Signale schnell erfasst, die Anzeige klingt langsam ab.
-4 Spitze	Extreme Anzeige für ganz kurze Impulse, wobei garantiert der Spitzenwert gezeigt wird. Bei ansteigender Amplitude Zeitkonstante Null (ist im Computer exakt machbar, in Analogschaltung nur näherungsweise) . Bei abfallender Amplitude 3s.

Da das menschliche Ohr für unterschiedliche Frequenzen unterschiedlich empfindlich ist, gibt es Frequenz-Bewertungen. Im Folgenden sind die wählbaren Frequenz-Bewertungen aufgelistet. Bei linearer Bewertung wird nicht bewertet. Für Frequenzen zwischen den angegebenen Stützpunkten wird linear interpoliert, d. h. der dB-Wert wird linear interpoliert. über die Enden der Tabelle hinaus wird die letzte Bewertung konstant gehalten.

Bewertungen

Frequenz [Hz]	A [dB]	B [dB]	C [dB]	D [dB]
10	-70.4	-38.2	-14.3	-26.5
12.5	-63.4	-33.2	-11.2	-24.5
16	-56.7	-28.5	-8.5	-22.5
20	-50.5	-24.2	-6.2	-20.5
25	-44.7	-20.4	-4.4	-18.5
31.5	-39.4	-17.1	-3.0	-16.5
40	-34.6	-14.2	-2.0	-14.5
50	-30.2	-11.6	-1.3	-12.5
63	-26.2	-9.3	-0.8	-11.0
80	-22.5	-7.4	-0.5	-9.0
100	-19.1	-5.6	-0.3	-7.5
125	-16.1	-4.2	-0.2	-6.0
160	-13.4	-3.0	-0.1	-4.5
200	-10.9	-2.0	0.0	-3.0
250	-8.6	-1.3	0.0	-2.0
315	-6.6	-0.8	0.0	-1.0
400	-4.8	-0.5	0.0	-0.5
500	-3.2	-0.3	0.0	0.0
630	-1.9	-0.1	0.0	0.0
800	-0.8	0.0	0.0	0.0
1000	0.0	0.0	0.0	0.0
1250	0.6	0.0	0.0	2.0
1600	1.0	0.0	-0.1	5.5
2000	1.2	-0.1	-0.2	8.0
2500	1.3	-0.2	-0.3	10.0
3150	1.2	-0.4	-0.5	11.0
4000	1.0	-0.7	-0.8	11.0
5000	0.5	-1.2	-1.3	10.0
6300	-0.1	-1.9	-2.0	8.5
8000	-1.1	-2.9	-3.0	6.0
10000	-2.5	-4.3	-4.4	3.0
12500	-4.3	-6.1	-6.2	0.0
16000	-6.6	-8.4	-8.5	-4.0
20000	-9.3	-11.1	-11.2	-7.5

Folgende Nenn-Durchlass-Bereiche gelten für Oktav-Filter, wobei sich die Zahlenwerte der Frequenzen alle 3 Dekaden wiederholen. Damit kann die Liste nach hinten und vorn beliebig verlängert werden.

Oktaven

Mittenfrequenz [Hz]	Untere Grenze [Hz]	Obere Grenze [Hz]
16	11.2	22.4
31.5	22.4	45
63	45	90
125	90	180
250	180	355
500	355	710
1000	710	1400
2000	1400	2800
4000	2800	5600
8000	5600	11200
16000	11200	22400

Folgende Frequenzen gelten für Terzen, wobei die Zahlenwerte sich für jede Dekade wiederholen, so dass die Tabelle nach vorn und hinten verlängert werden kann.

Terzen

Mittenfrequenz [Hz]	Untere Grenze [Hz]	Obere Grenze [Hz
1000	900	1120
1250	1120	1400
1600	1400	1800
2000	1800	2240
2500	2240	2800
3150	2800	3550
4000	3550	4500
5000	4500	5600
6300	5600	7100
8000	7100	9000
10000	9000	11200

Die Mitten-Frequenzen der 1/12- und 1/24-Oktave-Bänder liegen auf den Mittenfrequenzen der Terzen und auf Zwischenwerten, die aus logarithmisch gleichen Abständen gebildet werden. Dabei werden als weitere Frequenz-Stützpunkte die Randbereiche der Terzen benutzt.

Beispiel für 1/12-Okaven:

1/12 Oktaven

Mittenfrequenz [Hz]	Untere Grenze [Hz]	Obere Grenze [Hz]
1000	974	1029
1058	1029	1089
1120	1089	1151
1183	1151	1216
1250	1216	1286
1323	1286	1361
1400	1361	1448
1497	1448	1547
..	..	..

Beispiel für 1/24-Okaven:

1/24 Oktaven

Mittenfrequenz [Hz]	Untere Grenze [Hz]	Obere Grenze [Hz]
1000	987	1014
1029	1014	1043

1058	1043	1073
1089	1073	1104
1120	1104	1135
1151	1135	1167
1183	1167	1200
1216	1200	1233
1250	1233	1268
1286	1268	1304
1323	1304	1342
..	..	..

Wenn Datensätze im Frequenz-Bereich erzeugt werden, wird die x-Achse auf eine spezielle Weise skaliert. Die Frequenz-Bänder haben einen logarithmischen Abstand, so dass es zweckmäßig ist, die x-Achse mit dem Logarithmus der Frequenz zu kennzeichnen. Wenn Sie solche Datensätze dann als Kurvenfenster darstellen, können Sie die Darstellungsart "Terz-Beschriftung" für die x-Achse wählen. Der Logarithmus wird dann wieder expandiert und die Frequenzen laut DIN an die Achse geschrieben. Die folgende Tabelle zeigt an einigen Beispielen den Zusammenhang zwischen der x-Skalierung der Daten und den Frequenz-Bändern, wobei folgende Regel zugrunde liegt: Bilden Sie den zehnfachen Zehner-Logarithmus der Mittenfrequenz und runden Sie den Wert.

x-Skalierung

x-Skalierung	**Mitten-Frequenz [Hz]**
..	..
-3	0.5
-2	0.63
-1	0.8
0	1
1	1.25
2	1.6
3	2
4	2.5
5	3.15
6	4
7	5
8	6.3
9	8
10	10
11	12.5
..	..
20	100
30	1000
40	10000
41	12500
42	16000
43	20000
..	..

Die Terzen finden Sie an den x-Positionen 0, 1, 2..., die Oktaven an den Positionen 0, 3, 6, 9, 12..., 1/12-Oktaven an den Positionen 0, 0.25, 0.5, 0.75, 1, 1.25..., und 1/24-Oktaven finden Sie an allen Vielfachen von 1/8.

D. h. die unterschiedlichen Bandbreiten sind als Vielfache einer Terz ausgedrückt. Dementsprechend wird das Delta-X des Ergebnis-Datensatzes bei Frequenz-Skalierung auf folgende Werte gesetzt:

Bandbreite	**Delta-X**
Oktave	3

Terz	1
1/12 Oktave	0.25
1/24 Oktave	0.125

Die Funktion OctA() hat selbst nur 3 Argumente. Alle anderen Einstellungen werden über die Funktion OctI(). Die Argumente der Funktion OctI() sind hier ebenfalls erläutert.

Parameter	Bedeutung
FrequenzUnten	Das eine Ende des interessierenden Frequenz-Bereiches. Wenn der Wert Null angegeben wird, wird der Wert auf [FrequenzOben] gesetzt. Sie geben stets die Mitte eines Frequenz-Bandes an. Die Mitten-Frequenz muss nicht exakt getroffen werden. Wichtig ist aber, dass die Frequenz innerhalb des Durchlassbereichs des gewünschten Bandes liegt. Es wird dann automatisch die exakte Frequenz benutzt. Mindestens eine der beiden angegebenen Frequenzen muss größer als Null sein. Angaben kleiner Null sind nicht erlaubt.
FrequenzOben	Wie [FrequenzUnten], aber das andere Ende des Frequenz-Bereichs
...Integrationen	Anzahl der auszuführenden Integrationen 0 Keine Integration 1 Eine Integration 2 Zweifache Integration -1 Erste Ableitung -2 Zweite Ableitung Haben Sie z. B. mit einem Beschleunigungs-Sensor gemessen und möchten aber den Weg analysieren, wählen Sie eine zweifache Integration.
...ZeitBewert	Mittelungsdauer, Zeitbewertung > 0 Ihre Mittelungsdauer 0 Keine Mittelung -1 FAST -2 SLOW -3 Impuls -4 Spitze
...Reduktion	Reduktionsfaktor. Um diesen Faktor wird die Anzahl der Ergebnis-Punkte gegenüber den Quell-Punkten reduziert. Ein Faktor 2 bedeutet, dass jeder 2. Wert genommen wird. Ein Faktor 1 bedeutet, dass jeder Wert ohne Reduktion benutzt wird. Ein Reduktionsfaktor von 0 deutet an, dass der Faktor gleich der Länge des Quell-Datensatzes sein soll, womit genau ein Ergebnis-Punkt erzeugt wird. Bei Reduktionsfaktor = 0 wird ein Effektivwert für den gesamten Datensatz erzeugt. Dieser Effektivwert ist der echte (gleichgewichtete) Effektivwert, wie ihn auch die mathematische Funktion RMS() berechnet. Die für die Zeitbewertung angegebene Konstante wird nicht beachtet.
...Breite	Breite der Frequenz-Bänder -1 Oktave -2 Terz -3 1/12 Oktave -4 1/24 Oktave
...FreqBewert	Frequenz-Bewertung 0 linear 1 A 2 B 3 C 4 D
...Bezug	Bezugswert zur Normierung, mit diesem Wert wird multipliziert. Bei einem Bezugswert von Null wird keine Normierung durchgeführt.

Die Algorithmen entsprechen:

- DIN 45651, Oktavfilter für elektroakustische Messungen
- DIN 45652, Terzfilter für elektroakustische Messungen
- DIN IEC 651, Schallpegelmesser
- Die Filterungen werden mit Bandpässen 8. Ordnung durchgeführt. Die Bandpässe haben Butterworth-Charakteristik und genügen den Anforderungen nach DIN EN 61260 und IEC 1260, Klasse 0.

Achtung, Zeitkonstante!

Wenn Sie die gleitende Effektiv-Wert-Bildung benutzen, beachten Sie, dass die Wirkung anders ist als die eines einfachen Tiefpasses. Die Zeitkonstante, die Sie angeben, ist nicht direkt ablesbar, denn das gefilterte Signal wird noch anschließend radiziert. Sie können aber die Zeitkonstante besser ablesen, wenn Sie das Ergebnis quadrieren!

Abtastzeiten

++++
Wenn die Funktion eine Filterung durchführt, wird ein digitales Filter benutzt. Die Grenzfrequenzen dieses digitalen Filters können nur kleiner als die halbe Abtastfrequenz sein. Höhere Grenzfrequenzen sind nicht möglich. Maßgeblich sind die Frequenzen der Ränder der Bänder, nicht die Mitten-Frequenzen.

Genauigkeit der Filterung

>
Der Entwurf der digitalen Filter, die für den Bandpass benutzt werden, ist nur für Frequenzen deutlich unterhalb der halben Abtastfrequenz gut. Um überhaupt den Bandpass ausnutzen zu können, sollte die Abtastfrequenz deutlich höher als das Doppelte der obersten auszuwertenden Frequenz sein.

Gleitender Effektivwert

Die Mittelungszeit darf größer oder kleiner als die Abtastzeit sein. Gerechnet wird mit einer Approximation nullter Ordnung (treppen-förmiges Signal). Bei Reduktionsfaktor Null wird der gleichgewichtete Effektivwert berechnet, siehe Funktion RMS(). Das ist für das Spektrum des gesamten Datensatz die beste Aussage.

Einschwingen

Beachten Sie beim Anwenden von Bandpässen das Einschwingen, dessen Dauer mit 1/Frequenz-Differenz abgeschätzt werden kann. Ignorieren Sie ggf. die ersten Werte des Ergebnis-Datensatzes. Das Einschwingen ist kein spezielles Problem von FAMOS oder von digitalen Filtern, sondern von Filtern ganz allgemein.

Zahlenbereich

Beachten Sie den zulässigen Zahlenbereich, wenn Sie einen Bezugswert angeben. Bei überschreitungen des Zahlenbereichs wird der entsprechende Wert des Ergebnis-Datensatzes auf Null gesetzt.

Einheit

Die x-Einheit des Quell-Datensatzes wird in Sekunden erwartet. Die der Funktion übergebenen Zeit-Angaben werden in derselben Einheit erwartet. Die Frequenz-Angaben werden in Hz erwartet. Das Ergebnis trägt stets die y-Einheit der Quell-Daten, auch bei Integrationen. Die x-Einheit des Quell-Datensatzes wird bei einer Zeitdarstellung übernommen, bei einer Darstellung in Frequenzen wird "Tz.Nr." angegeben. Diese Einheit deutet an, dass eine Nummer zu sehen ist, anhand derer die Terzen gezählt werden können (Terz-Nummer). Eine Darstellung mit richtiger Beschriftung kann am Kurvenfenster gewählt werden.

Y-Skalierung

Die berechneten Werte werden stets linear angegeben. Wenn Sie eine Darstellung in dB wünschen, dann können Sie die mathematische Funktion dB() oder an den Kurvenfenstern eine entsprechende Darstellung wählen.

Reduktion

Beachten Sie, dass bei einer Datenreduktion der zeitbewertete Datensatz lediglich abgetastet wird. Wenn Sie z. B. einen Reduktionsfaktor von 3 wählen, dann werden der erste und zweite Punkt verworfen, der dritte, sechste, neunte usw. werden in das Ergebnis übertragen. Sie sollten also nicht wesentlich langsamer abtasten als die Zeitkonstante ist. Empfohlen sind 3 Abtastungen pro Zeitkonstante, mehr scheint nicht nötig, weil die Zwischenwerte ohnehin etwa auf den Verbindungsgeraden liegen. Bei Reduktionsfaktor Null wird der gleichgewichtete (ganz gewöhnliche) Effektivwert über alle Daten berechnet. Die Zeitkonstante für die Zeitbewertung wird nicht beachtet.

Beispiele:

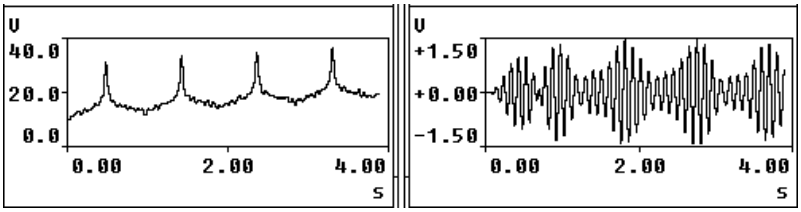
Ein Testdatensatz wird aus dem Beispiel-Datensatz ANSTIEG gewonnen. Die Terz mit der Mittenfrequenz 10Hz wird dabei in Abhängigkeit von der Zeit analysiert. Es soll also festgestellt werden, wie sich die Frequenz-Anteile im Bereich von 10Hz im Lauf der Zeit verändern.

Zuerst wird der Datensatz b durch Interpolation und Verändern der Abtastzeit erzeugt. Damit hat er eine Dauer von knapp 4s bei einer Abtastzeit von 1ms.

```
b = ipol((xdel(anstieg, 0.01), 10)
```

Die Frequenz-Band-Analyse wird nun so initialisiert, dass keine Integrationen durchgeführt werden, keine Zeitbewertung und keine Datenreduktion eingestellt sind. Es wird eine Terz-Filterung ohne Frequenz-Bewertung durchgeführt. Es wird auch nicht normiert. Das ist nun die einfachste Einstellung für die Funktion OctI(). Anschließend wird mit der Funktion OctA() der Datensatz t_10 erzeugt, der das Bandpass-gefilterte Signal bei der Terz 10Hz darstellt.

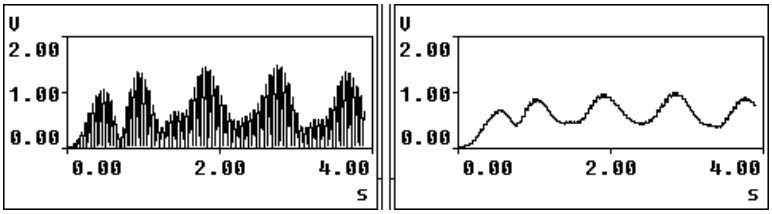
```
OctI(0, 0, 1, -2, 0, 0, 0)
t_10 = OctA(b, 10.0, 0)
```



Um nun die Amplitude dieses Signals über der Zeit betrachten zu können, ist eine Gleichrichtung erforderlich. Das wird mittels der Zeitbewertung eingestellt. Die Zeitbewertung wird zunächst einmal mit einer ganz kleinen Zeit (1e-20s) durchgeführt, danach mit der Standard-Variante FAST. Bei der ganz kleinen Zeitkonstante entspricht das Ergebnis einer gewöhnlichen Gleichrichtung.

```
OctI(0, 1e-20, 1, -2, 0, 0, 0)
tgl_10 = OctA(b, 10.0, 0)
OctI(0, -1, 1, -2, 0, 0, 0)
tf_10 = OctA(b, 10.0, 0)
```

Wenn eine gewisse Mittelungsdauer eingestellt wird, ist der Trend der Amplitude wesentlich besser zu erkennen:

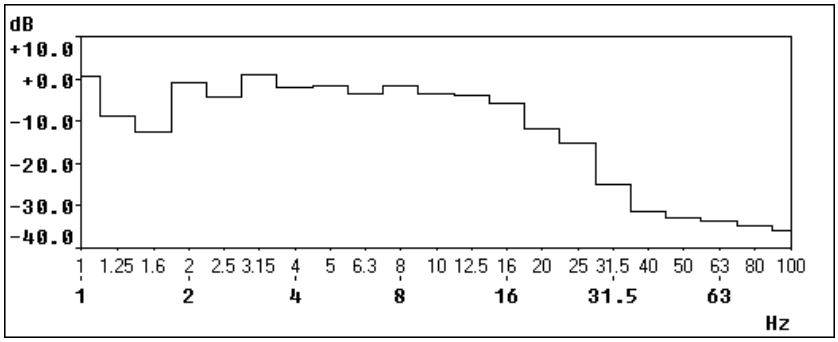


Nun werden mehrere Terzen gleichzeitig bestimmt, aber nur einmal für den gesamten Datensatz. Im Unterschied zu vorher wird die Reduktion auf 0 gesetzt, was die Zeitbewertung ausschaltet. Das Terz-Spektrum wird von 1Hz bis 100Hz bestimmt. Das sind dann die Amplituden für die entsprechende Terz, bezogen auf bzw. gemittelt über den ganzen Datensatz. Denn bei Reduktionsfaktor 0 wird ein Wert pro Datensatzlänge erzeugt.

```
OctI(0, 0, 0, -2, 0, 0, 0)
tz = OctA(b, 1, 100)
```

In dieser grafischen Darstellung wird die x-Achse in Terzen und Oktaven beschriftet. Diese Darstellung ist wählbar über das Darstellungs-Menü der Kurvenfenster und eine entsprechende Einstellung an der x-Achse.

Außerdem wurde die y-Achse in dB skaliert, Treppen wurden eingestellt.

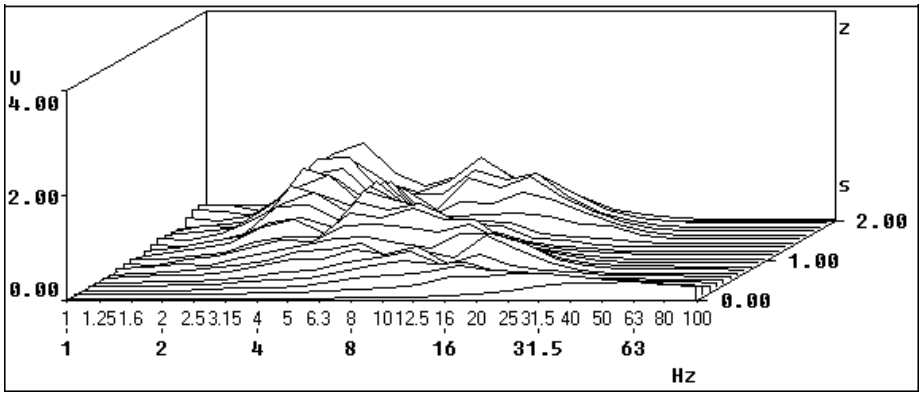


Nun wird die kompliziertesteste Form angewendet. Alle Terzen von 1Hz bis 100Hz werden über der Zeit ermittelt und in einem Wasserfall-Diagramm dargestellt. Es wird ein Reduktionsfaktor von 100 und die FAST-Zeitbewertung gewählt, alle restlichen Einstellungen sind wie im vorherigen Beispiel. Mit der Funktion OctA() wird ein Datensatz erzeugt, der auf ganz bestimmte Weise weiterverarbeitet und dargestellt werden muss.

```
OctI(0, -1, 100, -2, 0, 0, 0)
tz_21 = OctA(b, 1, 100)
```

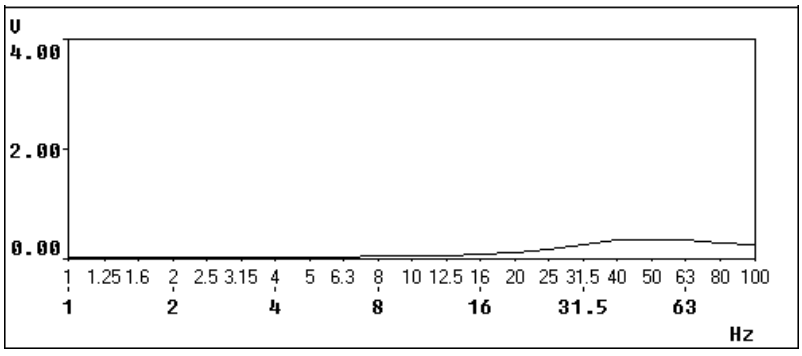
Zur Darstellung des Datensatzes in einem Wasserfall-Diagramm wird zunächst der Datensatz als Kurvenfenster dargestellt, dann wird die Wasserfall-Darstellung gewählt, der Perioden-Vergleich und die Terz-Beschriftung der x-Achse.

Über die Auswahl der weiteren Kurven wird die Periode von 21 angegeben. 21 ist die Anzahl der Terzen, die zwischen 1Hz und 100Hz liegen. Das Ergebnis der Funktion OctA ist also als Multi-Perioden-Datensatz zu verstehen. Sie können weiterhin die Anzahl der zu vergleichenden Perioden angeben, im Beispiel sind es 21 (Diese 21 hat nichts mit der Anzahl der Terzen zu tun, sondern ist (2Sek. / 0.1s + 1)). Mit der Abtastzeit des Quell-Datensatzes von 1ms und dem Reduktionsfaktor von 100 wird alle 0.1s ein komplettes Terzspektrum erzeugt. über die Optionen zur 3D-Darstellung kann die z-Achse entsprechend mit dz=0.1s skaliert werden. Damit wird die z-Achse von 0s bis 2s beschriftet.



Nun werden aus dem Datensatz, der als Matrix verstanden werden kann, einzelne Spektren herausgeschnitten, und zwar die ersten drei. Dazu wird die Funktion Perio() benutzt. Auch hier ist wieder die Länge eines Spektrums (21 Terzen) anzugeben.

```
t0 = Perio(tz_21, 21, 0)
t1 = Perio(tz_21, 21, 1)
t2 = Perio(tz_21, 21, 2)
```



Siehe auch:

OctI, SpecThirds, SpecThirds_1, OtrRpmThirds, ExpoRMS, ABCRating

OctI

Initialisierung der Terz-Analyse-Funktion OctA().
Alternativer Name: **TerzI**

Deklaration:

```
OctI ( EwIntegrationen, EwZeitbewertung, EwReduktion, EwBreite, EwFrequenzBewertung, EwBezug, Null )
```

Parameter:

EwIntegrationen	Anzahl der auszuführenden Integrationen
	0 : Keine Integration (Standard)
	1 : Eine Integration
	2 : Zweifache Integration
	-1 : Erste Ableitung
	-2 : Zweite Ableitung
EwZeitbewertung	Mittelungsdauer, Zeitbewertung
	>0 : Freie Mittelungsdauer
	0 : Keine Mittelung
	-1 : FAST (Standard)
	-2 : SLOW
	-3 : Impuls
	-4 : Spitze
EwReduktion	Reduktionsfaktor (Standard = 0)
EwBreite	Breite der Frequenz-Bänder
	-1 : Oktave
	-2 : Terz (Standard)
	-3 : 1/12 Oktave
	-4 : 1/24 Oktave
EwFrequenzBewertung	Frequenz-Bewertung
	0 : Linear (Standard)
	1 : A
	2 : B
	3 : C
	4 : D
EwBezug	Bezugswert zur Normierung, mit diesem Wert wird multipliziert. Standard ist 0 - keine Normierung.
Null	Reservierter Parameter, auf 0 zu setzen.

Beschreibung:

Mit dieser Funktion wird eine Terz-Analyse initialisiert. Hier werden viele Parameter einmal eingestellt, die dann bei einer wiederholten Ausführung der Terz-Analyse nicht jedes Mal wieder angegeben werden müssen.

Die Parameter sind ausführlich in ihrer Wirkung bei der Funktion Terz-Analyse-Funktion OctA() beschrieben.

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Siehe auch:

OctA, SpecThirds, SpecThirds_1, OtrRpmThirds

OdsGetLastErrorCode

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Abfrage des Fehlercodes zum zuletzt aufgetretenen Fehler einer ODS-Funktion.

Deklaration:

```
OdsGetLastErrorCode ( ) -> EwError
```

Parameter:

EwError	Fehlercode

Beschreibung:

Der Aufruf dieser Funktion löscht den internen Fehlerspeicher.

Siehe auch:

OdsGetLastErrorTxt

OdsGetLastErrorTxt

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Abfrage der Fehlerbeschreibung zum zuletzt aufgetretenen Fehler einer ODS-Funktion.

Deklaration:

```
OdsGetLastErrorTxt ( ) -> TxError
```

Parameter:

TxError	Fehlertext

Beschreibung:

Der Aufruf dieser Funktion löscht den internen Fehlerspeicher.

Siehe auch:

OdsGetLastErrorCode

OdsIEAddAttribute

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Numerisches Instanz-Attribut zufügen.

Deklaration:

```
OdsIEAddAttribute ( InstanzID, TxName, Daten, Null ) -> Status
```

Parameter:

InstanzID	ID des Instanz-Elementes
TxName	Name des neuen Attributs.
Daten	Datensatz oder Einzelwert mit dem Inhalt des neuen Attributs.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
Status	Status der Funktion
	0 : Fehler
	1 : Funktion erfolgreich ausgeführt.

Beschreibung:

Dem angegebenen Instanz-Element wird ein neues Attribut zugefügt.

Ein solches lokal definiertes Attribut (auch als Instanz-Attribut bezeichnet) ist privat zum Instanz-Element und nicht durch das Applikationsmodell definiert.

Der ODS-Datentyp des neuen Attributs wird automatisch aus dem Datentyp des übergebenen Datensatzes bestimmt.

Im Fehlerfall wird eine 0 zurückgeliefert, bei erfolgreicher Ausführung der Funktion eine 1. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Allen Instanz-Elementen, die momentan in der Baumansicht des ODS-Browsers selektiert sind, werden 2 private Attribute zugefügt.

```
SessionID = OdsPluginSessionConnect( 0 )
WENN SessionID <> 0
   IDList = OdsPluginListSelItems(0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      OdsIEAddAttribute( IDList[i], "Status", 1, 0)
      OdsIEAddAttributeTxt( IDList[i], "Prüfer", "Mustermann", 0)
      i = i+1
   ENDE
ENDE
```

Siehe auch:

OdsIEGetAttribute, OdsIESetAttributeTxt, OdsIEAddAttributeTxt

OdsIEAddAttributeTxt

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Instanz-Attribut (Text) zufügen.

Deklaration:

```
OdsIEAddAttributeTxt ( InstanzID, TxName, TxInhalt, Null ) -> Status
```

Parameter:

InstanzID	ID des Instanz-Elementes
TxName	Name des neuen Attributs.
TxInhalt	Inhalt des neuen Attributs.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
Status	Erfolg der Funktion
	1 : Funktion erfolgreich durchgeführt
	0 : Fehler.

Beschreibung:

Dem angegebenen Instanz-Element wird ein neues Text-Attribut zugefügt.

Ein solches lokal definiertes Attribut (auch als Instanz-Attribut bezeichnet) ist privat zum Instanz-Element und nicht durch das Applikationsmodell definiert.

Der ODS-Datentyp des neuen Attributs ist [DT_STRING].

Im Fehlerfall wird eine 0 zurückgeliefert, bei erfolgreicher Ausführung der Funktion eine 1. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Allen Instanz-Elementen, die momentan in der Baumansicht des ODS-Browsers selektiert sind, werden 2 private Attribute zugefügt.

```
SessionID = OdsPluginSessionConnect( 0 )
WENN SessionID <> 0
   IDList = OdsPluginListSelItems(0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      OdsIEAddAttribute( IDList[i], "Status", 1, 0)
      OdsIEAddAttributeTxt( IDList[i], "Prüfer", "Mustermann", 0)
      i = i+1
   ENDE
ENDE
```

Siehe auch:

OdsIEGetAttribute, OdsIESetAttributeTxt, OdsIEAddAttributeTxt

OdsIEBuildVarName

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Aus dem ASAM-Pfad eines Instanz-Elementes wird ein gültiger FAMOS-Variablenname abgeleitet.

Deklaration:

```
OdsIEBuildVarName ( InstanzID, Option ) -> TxVarName
```

Parameter:

InstanzID	ID des Instanz-Elementes
Option	Option für Namensbildung.
	0 : Nur Name des Instanz-Elementes
	1 : Name des Eltern-Elementes wird vorangestellt.
	2 : Name des Eltern-Elementes wird angehängt.
TxVarName	Gültiger FAMOS-Variablenname.

Beschreibung:

Zu einem gegebenen Instanz-Element wird ein gültiger FAMOS-Variablenname abgeleitet. Dazu werden der Name des Instanz-Elementes (Attribut "name" im ODS-Basismodell) sowie je nach Option (2. Parameter) weitere Elemente des ASAM-Pfades herangezogen.

Unzulässige Zeichen im resultierenden Namen werden durch einen Unterstrich '_' ersetzt.

Im Fehlerfall wird ein leerer Text zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Eine Verbindung zu einem ODS-Server mit dem Namen "ODSTest" wird hergestellt und alle Messgrößen nach FAMOS importiert, deren Name mit "Channel1_" beginnt. Der FAMOS-Variablenname wird aus dem Namen der übergeordneten Messung und dem Kanalnamen gebildet.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurementQuantity" ,"Channel1_*", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxVarName = OdsIEBuildVarName( IDList[i], 1)
       <TxVarName> = OdsIEGetChannel( IDList[i], "", 0)
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEGetChannel, OdsIEGetMeasurement

OdsIECreateElement

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Anlegen eines neuen Instanz-Elementes

Deklaration:

```
OdsIECreateElement ( Typ, InstanzID, Option, Null ) -> IDNeueInstanz
```

Parameter:

Typ	ODS-Typbezeichner für die anzulegende Instanz.
InstanzID	ID des verwandten Instanz-Elementes. Bedeutung abhängig vom nächsten Parameter.
Option	Relation der neuen Instanz bzgl. [InstanzID].
	0 : Das neue Instanz-Element wird als Kind vom [InstanzID] angelegt.
	1 : Das neue Instanz-Element wird als Kopie von [InstanzID] angelegt. Beide Elemente haben anschließend den gleichen Vater.
	2 : Das neue Instanz-Element ist ein TopLevel-Element, besitzt also keinen Vater. [InstanzID] muß 0 sein.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
IDNeueInstanz	ID der neu angelegten Instanz bei Erfolg, 0 sonst.

Beschreibung:

Die Attribute der neuen Instanz werden mit Standardwerten initialisiert.

Die Funktion kann **nicht** benutzt werden, um Instanzen vom Typ "AoMeasurementQuantity", "AoLocalColumn" oder "AoSubMatrix" zu erzeugen. Zu diesem Zweck benutzen Sie die Funktion OdsIEImportData().

Beispiele:

Eine neue TopLevel-Instanz vom Typ "AoTest" wird erzeugt und diverse Attribute gesetzt.

```
SessionID = OdsSessionCreate( "TEST", "","", 0 )
WENN SessionID > 0
   id = OdsIECreateElement("#AoTest", 0, 2, 0)
   WENN id > 0
      OdsIESetAttributeTxt( id, "#name", "Test_123", 0)
      OdsIESetAttributeTxt( id, "#description", "Test only", 0)
      OdsIESetAttribute( id, "Charge", 1205, 0)
      OdsIESetAttribute( id, "#version_date", ZeitSystem?(), 0)
      OdsSessionClose(0)
   ENDE
ENDE
```

Siehe auch:

OdsIEDeleteElement, OdsIEImportData

OdsIEDeleteElement

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Löschen eines Instanz-Elementes

Deklaration:

```
OdsIEDeleteElement ( InstanzID, Null ) -> Status
```

Parameter:

InstanzID	ID des zu löschenden Instanz-Elementes.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
Status	Erfolg der Funktion
	0 : Fehler
	1 : Funktion erfolgreich ausgeführt.

Beschreibung:

Das Löschen von Instanzen kann nicht rückgängig gemacht werden, Sie sollten diesen Befehl also mit entsprechender Vorsicht anwenden.

Mit dem Löschen eines Instanz-Elementes werden auch alle Kinder des Elementes gelöscht, also alle Elemente im entsprechenden Zweig der Baumdarstellung.

Es wird hier nicht geprüft, ob durch das Löschen von Instanzen die Datenablage eventuell inkonsistent wird. Es kann z.B. sein, das in anderen Instanzen noch Relationen existieren, die auf hier gelöschte Instanzen verweisen. Diese werden dann natürlich ungültig.

Beispiele:

Es werden alle Instanzen von Typ "AoTest" aufgezählt, deren Name mit "T_" beginnt.

Jede gefunden Instanz wird (nach Bestätigung durch den Anwender) gelöscht.

```
SessionID = OdsSessionCreate( "Test", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoTest", "T_*", 0)
   i = 1
   Count = Lang?( IDList)
   solange i >= Count
      TxName = OdsIEGetAttributeTxt( IDList[i], "#name", 0)
      WENN BoxNachricht( "Instanz löschen?", TxName, "?4") = 1
         OdsIEDeleteElement( IDList[i], 0)
      ENDE
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIECreateElement

OdsIEGetAttribute

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Abfragen des Wertes eines numerischen Attributs eines Instanz-Elementes.

Deklaration:

```
OdsIEGetAttribute ( InstanzID, TxAttributName, Null ) -> Wert
```

Parameter:

InstanzID	ID des Instanz-Elementes
TxAttributName	Name des Attributs. Entweder der entprechende Name laut Applikationsmodell der Datenbasis oder, sofern existent, Name laut ODS-Basismodell.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
Wert	Wert(e) des spezifizierten Attributs.

Beschreibung:

Als Attribut-Bezeichner (2. Parameter) kann entweder der entsprechende Name laut Applikationsmodell oder alternativ der Name aus dem Basismodell angegeben werden. Für Basis-Attributnamen muß dem eigentlichen Bezeichner ein '#' vorangestellt werden.

Der Inhalt des Attributs muß entweder in einen numerischen Wert oder in einen Vektor von numerischen Werten konvertierbar sein.

Falls es sich bei dem Attribut um eine Datums-/Zeitangabe (Datentyp DT_DATE) handelt, wird diese in das FAMOS-interne Zeitformat (double) konvertiert.

Im Fehlerfall wird ein leerer Datensatz (Länge = 0) zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Jedes ODS-Element muß als als Pflichtattribut eine eindeutige ID besitzen, der Name dieses Attributs im Basismodell ist "id". Eine Verbindung zu einem ODS-Server mit dem Namen "ODSTest" wird geöffnet und die ID einer durch ihren ASAM-Pfad spezifizierten Messung ausgelesen.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   InstanzID = OdsIEListByAsamPath("/[Test]T_2;/[SubTest]SV_5;/[Measurement]N1_14;", 0)
   WENN InstanzID > 0
      ID = OdsIEGetAttribute( InstanzID, "#id", 0)
      ; Angenommen, der korrespondierende Attributname im
      ; Applikationsmodell ist "MessungsID", dann ist der Aufruf
      ; ID = OdsIEGetAttribute( InstanzID, "MessungsID", 0)
      ; äquivalent
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEGetAttributes, OdsIEGetAttributeTxt, OdsIEGetPropertyTxt

OdsIEGetAttributes

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Abfragen aller Attribute eines Instanz-Elementes.

Deklaration:

```
OdsIEGetAttributes ( InstanzID, Null ) -> GrAttribute
```

Parameter:

InstanzID	ID des Instanz-Elementes
Null	Reservierter Parameter. Immer auf 0 zu setzen.
GrAttribute	Gruppe mit allen Attributen des Instanzlementes.

Beschreibung:

Die Funktion liest alle vorhandenen Attribute eines gegebenen Instanz-Elementes aus und liefert diese in Form einer Datengruppe zurück. Die Kanäle der Datengruppe erhalten den Names des jeweiligen Attributs, der jeweilige resultierende Datentyp (Datensatz oder Text) wird entsprechend dem ODS-Datentyp des Attributs automstisch bestimmt.

Im Fehlerfall wird eine leere Datengruppe zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Eine Verbindung zum ODS-Server "ODSTest" wird geöffnet und alle Messgrößen nach FAMOS importiert, deren Name mit "Channel1_" beginnt. Dabei wird pro Messgröße eine eigene Gruppe angelegt, deren Namen aus dem Namen der übergeordneten Messung sowie dem Namen der Messgröße gebildet wird. Jede Gruppe enthält die Attribute der Messung sowie die eigentlichen Messdaten.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurementQuantity" ,"Channel1_*", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxVarName = OdsIEBuildVarName( IDList[i], 1)
       <TxVarName> = OdsIEGetAttributes( IDList[i], 0)
       <TxVarName>:Daten = OdsIEGetChannel( IDList[i], "", 0)
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEGetAttribute, OdsIEGetAttributeTxt, OdsIEGetPropertyTxt, OdsIEGetChannel

OdsIEGetAttributeTxt

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Abfragen des Inhalts (als Text) eines Attributs eines Instanz-Elementes.

Deklaration:

```
OdsIEGetAttributeTxt ( InstanzID, TxAttributName, Null ) -> TxInhalt
```

Parameter:

InstanzID	ID des Instanz-Elementes
TxAttributName	Name des gesuchten Attributs
Null	Reservierter Parameter. Immer auf 0 zu setzen.
TxInhalt	Inhalt des spezifizierten Attributs

Beschreibung:

Als Attribut-Bezeichner (2. Parameter) kann entweder der entsprechende Name laut Applikationsmodell oder alternativ der Name aus dem Basismodell angegeben werden. Für Basis-Attribute muß dem eigentlichen Bezeichner ein '#' vorangestellt werden.

Der Inhalt des Attributs muß in einen Text konvertierbar sein, darf also z.B. keinen Vektor von numerischen Werten enthalten.

Im Fehlerfall wird ein leerer Text zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Eine Verbindung zu einem ODS-Server mit dem Namen "ODSTest" wird geöffnet und die Namen sowie das Startdatum aller enthaltenen Messungen [AoMeasurement] aufgelistet.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurement" ,"", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxName = OdsIEGetAttributeTxt( IDList[i], "#name", 0)
      TxDate = OdsIEGetAttributeTxt( IDList[i], "#measurement_begin", 0)
      BoxAusgabe( TxName + " " + TxDate, LEER, "", 1)
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEGetAttribute, OdsIEGetAttributes, OdsIEGetPropertyTxt

OdsIEGetChannel

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Zu einer gegebenen Messgröße werden die Messdaten ausgelesen und in einen FAMOS-Datensatz konvertiert.

Deklaration:

```
OdsIEGetChannel ( InstanzID, TxKanalName, Null ) -> Messdaten
```

Parameter:

InstanzID	ID des Instanz-Elementes, dessen Messdaten ausgelesen werden sollen. Die Instanz muß entweder von Typ Messung [AoMeasurement] oder Messgröße [AoMeasurementQuantity] sein.
TxKanalName	Wenn der erste Parameter auf eine Messung verweist, ist hier der Name der gewünschten Messgröße anzugeben. Wenn der erste Parameter direkt auf eine Messgröße verweist, ist ein leerer Text zu übergeben.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
Messdaten	Datensatz mit den Messdaten der gewählten Messgröße.

Beschreibung:

Die Funktion liest die Messdaten zu einer Messgröße [AoMeasurementQuantity] aus.

Die im ersten Parameter spezifizierte Instanz muss dabei einen der folgenden Typen aufweisen:

Messung [AoMeasurement]: Der erste Parameter bestimmt die Messung, der daraus zu wählende Kanal wird dann durch den im 2. Parameter angegebenen Namen festgelegt.

Messgröße [AoMeasurementQuantity]: Der erste Parameter bestimmt die gewünschte Messgröße direkt. Für den 2. Parameter ist dann ein leerer Text zu übergeben.

Im Fehlerfall wird ein leerer Datensatz (Länge = 0) zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Eine Verbindung zum ODS-Server "ODSTest" wird hergestellt. Es werden dann alle Messungen gesucht, deren Name mit "TEST2_" beginnt. Zu diesen Messungen wird dann jeweils der Kanal mit dem Namen "T2" ausgelesen.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurement" ,"TEST2_*", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      chan = OdsIEGetChannel( IDList[i],"T2", 1, 0)
      TxName = OdsIEBuildVarName( IDList[i],0)
      TxName = TxName + "_T2"
      BENENNE chan <TxName>
   ENDE
   OdsSessionClose(0)
ENDE
```

Wenn statt dessen alle Messgrößen mit dem Namen "T2" unabhängig von der übergeordneten Messung ausgelesen werden sollen:

```
...
 IDList = OdsIEListByType("#AoMeasurementQuantity" ,"T2", 0)
 Count = Lang?( IDList)
 i = 1
 SOLANGE i <= Count
      chan = OdsIEGetChannel( IDList[i],"", 1, 0)
      TxName = OdsIEBuildVarName( IDList[i],1)
      BENENNE chan <TxName>
 ENDE
...
```

Siehe auch:

OdsIEGetMeasurement

OdsIEGetMeasurement

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Die in einer Messung enthaltenen Kanäle werden ausgelesen und in eine FAMOS-Datengruppe konvertiert.

Deklaration:

```
OdsIEGetMeasurement ( InstanzID, TxNameMuster, AttributOption, Null ) -> GrMessung
```

Parameter:

InstanzID	InstanzID der Messung. Typ: AoMeasurement
TxNameMuster	Muster fur Kanalnamen
AttributOption	Attribute der Messung mit übernehmen?
	0 : Keine Attribute
	1 : Alle Attribute
Null	Reservierter Parameter. Immer auf 0 zu setzen.
GrMessung	Gruppe mit allen spezifizierten Kanälen und ggf. den Attributen der Messung.

Beschreibung:

Die Funktion liest die Messkanäle [AoMeasurementQuantity] der gewählten Messung aus und legt diese in Form einer Datengruppe ab. Die übergebene InstanzID muß auf ein Instanz-Element vom Typ Messung [AoMeasurement] verweisen. Optional werden auch die Attribute der Messung mit ausgelesen.

Mit dem 2. Parameter (TxNameMuster) können die zu lesenden Kanäle durch Angabe einer Suchmaske für den Kanalnamen eingeschränkt werden. Die üblichen Jokerzeichen '*' und '?' sind erlaubt. Wenn alle Kanäle gelesen werden sollen, geben Sie hier entweder einen leeren Text oder "*" an.

Im Fehlerfall wird eine leere Gruppe zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Eine Verbindung zum ODS-Server "ODSTest" wird hergestellt. Es werden dann alle Messungen gesucht, deren Name mit "TEST2_" beginnt. Zu diesen Messsungen werden dann alle Kanäle ausgelesen, deren Name mit "T2" beginnt.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurement" ,"TEST2_*", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      GrMeas = OdsIEGetMeasurement( IDList[i],"T2*", 1, 0)
      TxName = OdsIEBuildVarName( IDList[i],0)
      BENENNE GrMeas <TxName>
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEGetChannel

OdsIEGetPropertyTxt

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Abfrage von vordefinierten Eigenschaften eines Instanz-Elementes.

Deklaration:

```
OdsIEGetPropertyTxt ( InstanzID, TxName, Null ) -> TxEigenschaftsWert
```

Parameter:

InstanzID	ID des Instanz-Elementes
TxName	Auswahl der gesuchten Eigenschaft.
	"Path" : Kompletter Asam-Pfad des Elementes.
	"AppName" : Name des Elementes im Applikationsmodell.
	"BaseName" : Name des Elementes im ODS-Basismodell.
	"ParentName" : Name der Eltern-Instanz.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
TxEigenschaftsWert	Abgefragte Eigenschaft

Beschreibung:

Neben den durch Basis- und Applikationsmodell definierten Attributen eines Instanz-Elementes besitzt dieses auch einige feste, nicht änderbare Eigenschaften, die sich im Wesentlichen aus dem Typ eines Elementes und seiner Position in der ODS-Baumstruktur der Datenquelle ableiten lassen.

Dazu zählen z.B. der eindeutige ASAM-Pfad des Instanz-Elementes, sowie der Name des zugehörigen Elementes in Applikations- und Basismodell.

Im Fehlerfall wird ein leerer Text zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Eine Verbindung zum ODS-Server "ODSTest" wird hergestellt und die ASAM-Pfade aller enthaltenen Messungen [AoMeasurement], deren Name mit "Test2_" beginnt, aufgelistet.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurement" ,"TEST2_*", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxPath = OdsIEGetPropertyTxt( IDList[i], "Path", 0)
      BoxAusgabe( TxPath, LEER, "", 1)
      ; Der folgende Aufruf
      ; TxBaseName = OdsIEGetPropertyTxt( IDList[i], "BaseName", 0)
      ; liefert stets "AoMeasurement" zurück.
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEGetAttribute, OdsIEGetAttributes, OdsIEGetAttributeTxt

OdsIEImportData

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Einfügen von Messdaten in die ODS-Datenablage

Deklaration:

```
OdsIEImportData ( InstanzID, Daten, Null ) -> IDNeueInstanz
```

Parameter:

InstanzID	ID des Instanz-Elementes, zu dem die Daten eingefügt werden soll. Je nach Datentyp (Datensatz oder Datengruppe) muß die Instanz entweder vom Typ Messung [AoMeasurement] oder Test [AoTest/AoSubTest] sein.
Daten	Datensatz oder Datengruppe mit den Messdaten.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
IDNeueInstanz	ID der neu angelegten Instanz bei Erfolg, 0 sonst.

Beschreibung:

Die als Parameter übergebenen Messdaten werden als neue Instanz in die Datenablage übernommen.

Es kann entweder ein einzelner Datensatz oder eine Datengruppe importiert werden.

[Data] ist vom Typ Datensatz: Die Daten werden einer vorhandenen Messung als neuer Kanal zugefügt. [InstanzID] muß auf ein Element vom Typ Messung [AoMeasurement] verweisen, die neu angelegte Instanz ist vom Typ [AoMeasurementQuantity].

[Data] ist vom Typ Datengruppe: Die Daten werden einem vorhandenen Test als neue Messung zugefügt. [InstanzID] muß auf ein Element vom Typ Test [AoTest oder AoSubTest] verweisen, die neu angelegte Instanz ist vom Typ [AoMeasurement].

Beispiele:

In der Datenenlage wird eine neue Messung mit dem Namen "M_212" als Kind des Subtests "SV7" angelegt. Die beiden FAMOS-Datensätze "SO21" und "SO22" werden als Messgrößen zu dieser Messung importiert:

Variante 1:

```
SessionID = OdsSessionCreate( "Test", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoSubTest", "SV_7", 0)
   WENN lang?(IDList) = 1
      idTest = IDList[1]
      idmea = OdsIECreateElement("#AoMeasurement", idTest, 0, 0)
      ok = OdsIESetAttributeTxt( idmea, "#name", "M_212", 0)
      idmeq1 = OdsIEImportData( idmea, SO21, 0)
      idmeq2 = OdsIEImportData( idmea, SO22, 0)
   ENDE
   OdsSessionClose(0)
ENDE
```

Variante 2:

```
SessionID = OdsSessionCreate( "Test", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoSubTest", "SV_7", 0)
   WENN lang?(IDList) = 1
      M_212:SO21 = SO21
      M_212:SO22 = SO22
      idmea  = OdsIEImportData( idTest, M_212, 0)
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEGetMeasurement, OdsIEGetChannel

OdsIEListByAsamPath

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Ermittlung des Instanz-Elementes zu einem gegebenen ASAM-Pfad.

Deklaration:

```
OdsIEListByAsamPath ( TxAsamPfad, Null ) -> InstanzID
```

Parameter:

TxAsamPfad	Kompletter ASAM-Pfad für das gewünschte Instanz-Element.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
InstanzID	ID des Instanz-Elementes bei Erfolg. Diese ID muß bei allen Instanz-bezogenen Kit-Funktionen als Parameter angegeben werden.
	> 0 : ID des Instanz-Elementes
	0 : Fehler.

Beschreibung:

Der ASAM-Pfad beschreibt die Position der Instanz innerhalb der hierarchisch strukturierten Datenablage und ist für jede Instanz eindeutig. Er ist von Funktion und Aufbau in etwa vergleichbar mit dem Pfad einer Datei in einem Dateisystem.

Jedes Teil-Element des ASAM-Pfades wird durch den Name des beschreibenden Applikations-Elementes (in eckigen Klammern), den Namen der Instanz und, falls vorhanden, die Version der Instanz definiert, z.B.:

```
"/[Test]T_2;/[SubTest]SV_5;/[Measurement]N1_14;"
```

Im Fehlerfall wird eine 0 zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Eine Verbindung zu dem ODS-Server mit dem Namen "ODSTest" wird geöffnet und die Messdaten zu einer durch Ihren ASAM-Pfad spezifizierten Messgröße in FAMOS eingelesen.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   InstanzID = OdsIEListByAsamPath("/[Test]T_2;/[SubTest]SV_5;/[Measurement]N1_14;", 0)
   WENN InstanzID > 0
      TxVarName = OdsIEBuildVarName( InstanzID, 0)
       <TxVarName> = OdsIEGetChannel( InstanzID, "", 0)
      ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEListByType, OdsIEListChildren, OdsPluginListSelItems

OdsIEListByType

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Ermittlung aller Instanz-Elemente zu einem gegebenen Typ. Die Suche kann durch Vorgabe eines Namensmusters (mit Jokerzeichen) eingeschränkt werden.

Deklaration:

```
OdsIEListByType ( TxOdsTyp, TxNamensMuster, Null ) -> InstanzElemente
```

Parameter:

TxOdsTyp	Bezeichnung eines ODS-Element-Typs. Der Typname kann entweder aus dem Applikationsmodell der Datenquelle oder dem ODS-Basismodell stammen.
TxNamensMuster	Es werden nur solche Instanzen zurückgeliefert, deren Name dem hier angegebenen Muster (mit Jokerzeichen) entspricht
Null	Reservierter Parameter. Immer auf 0 zu setzen. Ausnahme für Muster = "#", siehe unten.
InstanzElemente	ID's aller Instanz-Elemente, die den Vorgaben entsprechen.

Beschreibung:

Für den Typ-Bezeichner (1. Parameter) kann entweder der entsprechende Name laut Applikationsmodell oder alternativ der Name aus dem ODS-Basismodell angegeben werden. Für Basismodell-Namen muß dem eigentlichen Bezeichner ein "#" vorangestellt werden.

Im Namensmuster (2. Parameter) können die Jokerzeichen "?" und "*" in üblicher Weise ("?" steht für genau ein beliebiges Zeichen, "*" für beliebig viele Zeichen) verwendet werden.

Um alle Instanzen des gewünschten Typs unabhängig vom Namen zu finden, übergeben Sie hier einen leeren Text oder "*".

Sonderfall: Wenn der 2. Parameter den Inhalt "#" hat, wird der 3. Parameter als ODS-Instanz-ID interpretiert. Diese entspricht dem Wert des Basisattributes "id" des Instanzelements. Das Ergebnis hat dann die Länge 0 (wenn Element nicht existiert) oder 1.

Der zurückgegebene Datensatz enthält die ID's aller Instanz-Elemente, die den Vorgaben entsprechen. Zur Bestimmung der Anzahl der gefundenen Elemente kann die Lang?() benutzt werden, zum Zugriff auf ein bestimmtes Element der '[ ]' Operator (siehe Beispiel).

Im Fehlerfall wird ein leerer Datensatz (Länge = 0) zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Eine Verbindung zum ODS-Server mit dem Namen "ODSTest" wird geöffnet und alle Messgrößen nach FAMOS importiert, deren Name mit "Channel1_" beginnt.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurementQuantity" ,"Channel1_*", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxVarName = OdsIEBuildVarName( IDList[i], 1)
       <TxVarName> = OdsIEGetChannel( IDList[i], "", 0)
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsIEListByAsamPath, OdsIEListChildren, OdsPluginListSelItems

OdsIEListChildren

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Ermittlung der direkten Kind-Instanzen (Nachfahren) eines gegebenen Instanz-Elementes.

Deklaration:

```
OdsIEListChildren ( InstanzID, TxOdsTyp, TxNamensMuster, Null ) -> InstanzElemente
```

Parameter:

InstanzID	ID des Instanz-Elementes, dessen Nachfahren bestimmt werden sollen.
TxOdsTyp	Bezeichnung eines ODS-Element-Typs. Es werden nur Instanzen des gegebenen Typs zurückgeliefert. Der Typname kann entweder aus dem Applikationsmodell der Datenquelle oder dem ODS-Basismodell stammen. Leerer Text, falls Typ egal.
TxNamensMuster	Es werden nur solche Instanzen zurückgeliefert, deren Name dem hier angegebenen Muster (mit Jokerzeichen) entspricht. Leerer Text oder "*", falls Name egal.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
InstanzElemente	ID's aller Instanz-Elemente, die den Vorgaben entsprechen.

Beschreibung:

Für den Typ-Bezeichner (2. Parameter) kann entweder der entsprechende Name laut Applikationsmodell oder alternativ der Name aus dem ODS-Basismodell angegeben werden. Für Basismodell-Namen muß dem eigentlichen Bezeichner ein '#' vorangestellt werden.

Wenn alle Kinder unabhängig vom Typ gelistet werden sollen, kann hier ein leerer Text übergeben werden.

Im Namensmuster (3. Parameter) können die Jokerzeichen '?' und '*' in üblicher Weise ('?' steht für genau ein beliebiges Zeichen, '*' für beliebig viele Zeichen) verwendet werden.

Um alle Instanzen unabhängig vom Namen zu finden, übergeben Sie hier einen leeren Text oder "*".

Der zurückgegebene Datensatz enthält die ID's aller Instanz-Elemente, die den Vorgaben entsprechen. Zur Bestimmung der Anzahl der gefundenen Elemente kann die Lang?() benutzt werden, zum Zugriff auf ein bestimmtes Element der '[ ]' Operator (siehe Beispiel).

Im Fehlerfall wird ein leerer Datensatz (Länge = 0) zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Ein ODS-Server mit dem Namen "ODSTest" wird geöffnet und nach allen Messungen gesucht, deren Name mit "TestA_" beginnt. Zu jeder gefundenen Messung werden dann die Namen der enthaltenen Messgrößen aufgelistet.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType( "#AoMeasurement, "TestA_*", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i<= Count
      TxName = OdsIEGetAttributeTxt( IDList[i], "#name", 0)
      BoxAusgabe( TxName, LEER, "", 1)
      IDChildList = OdsIEListChildren( IDList[i], "#AoMeasurementQuantity", "", 0)
      ChildCount = Lang?( IDChildList)
      j = 1
      SOLANGE j <= ChildCount
         TxName = ".." + OdsIEGetAttributeTxt( IDChildList[j], "#name", 0)
         BoxAusgabe( TxName, LEER, "", 1)
         j = j+1
      ENDE
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Hinweis: Gemäß ODS-Basismodell ist das Basis-Element [AoMeasurementQuantity] ein direkter Nachfahre von [AoMeasurement].

Siehe auch:

OdsIEListByAsamPath, OdsIEListByType, OdsPluginListSelItems

OdsIERemoveAttribute

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Instanz-Attribut löschen

Deklaration:

```
OdsIERemoveAttribute ( InstanzID, TxName, Null ) -> Status
```

Parameter:

InstanzID	ID des Instanz-Elementes
TxName	Name des zu löschenden Attributs.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
Status	Erfolg der Funktion
	1 : Funktion erfolgreich durchgeführt
	0 : Fehler.

Beschreibung:

Das angebenene Attribut wird gelöscht.

Es muß sich hierbei um ein lokales Instanz-Attribut handeln. Attribute, die durch das Applikationsmodell definiert sind, können nicht gelöscht werden.

Im Fehlerfall wird eine 0 zurückgeliefert, bei erfolgreicher Ausführung der Funktion eine 1. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Siehe auch:

OdsIEAddAttribute, OdsIEAddAttributeTxt

OdsIESetAttribute

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Ändern eines numerischen Attributs eines Instanz-Elementes.

Deklaration:

```
OdsIESetAttribute ( InstanzID, TxAttributName, Daten, Null ) -> FehlerCode
```

Parameter:

InstanzID	ID des Instanz-Elementes
TxAttributName	Name des Attributes. Entweder der entprechende Name laut Applikationsmodell der Datenbasis oder, sofern definiert, der Name laut ODS-Basismodell.
Daten	Datensatz oder Einzelwert mit dem neuen Inhalt des Attributs.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
FehlerCode	Erfolg der Funktion
	1 : Funktion erfolgreich durchgeführt
	0 : Fehler.

Beschreibung:

Als Attribut-Bezeichner (2. Parameter) kann entweder der entsprechende Name laut Applikationsmodell oder alternativ der Name aus dem Basismodell angegeben werden. Für Basis-Attributnamen muß dem eigentlichen Bezeichner ein '#' vorangestellt werden.

Der Datentyp des addressierten Attributs muß kompatibel zum angegebenen Datenparameter sein. Für numerische Attribute wird ein Einzelwert erwartet, für Attribute mit einem Verktor-Datentyp kann ein Datensatz passenden Typs (z.B. einfach reell oder komplex) angegeben werden.

Im Fehlerfall wird eine 0 zurückgeliefert, bei erfolgreicher Ausführung der Funktion eine 1. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Ein neue TopLevel-Instanz vom Typ "AoTest" wird erzeugt und diverse Attribute gesetzt.

```
SessionID = OdsSessionCreate( "TEST", "","", 0 )
WENN SessionID > 0
   id = OdsIECreateElement("#AoTest", 0, 2, 0)
   WENN id > 0
      OdsIESetAttributeTxt( id, "#name", "Test_123", 0)
      OdsIESetAttributeTxt( id, "#description", "Test only", 0)
      OdsIESetAttribute( id, "Charge", 1205, 0)
      OdsIESetAttribute( id, "#version_date", ZeitSystem?(), 0)
      OdsSessionClose(0)
   ENDE
ENDE
```

Siehe auch:

OdsIEGetAttribute, OdsIESetAttributeTxt

OdsIESetAttributeTxt

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Ändern des Inhalts (Text) eines Attributs eines Instanz-Elementes.

Deklaration:

```
OdsIESetAttributeTxt ( InstanzID, TxAttributName, TxInhalt, Null ) -> FehlerCode
```

Parameter:

InstanzID	ID des Instanz-Elementes
TxAttributName	Name des Attributs. Entweder der entprechende Name laut Applikationsmodell der Datenbasis oder, sofern definiert, der Name laut ODS-Basismodell.
TxInhalt	Neuer Inhalt des adressierten Attributs.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
FehlerCode	Erfolg der Funktion
	1 : Funktion erfolgreich durchgeführt
	0 : Fehler.

Beschreibung:

Als Attribut-Bezeichner (2. Parameter) kann entweder der entsprechende Name laut Applikationsmodell oder alternativ der Name aus dem Basismodell angegeben werden. Für Basis-Attributnamen muß dem eigentlichen Bezeichner ein '#' vorangestellt werden.

Der Datentyp des addressierten Attributs muß vom Datentyp <Text> sein.

Im Fehlerfall wird eine 0 zurückgeliefert, bei erfolgreicher Ausführung der Funktion eine 1. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Ein neue TopLevel-Instanz vom Typ "AoTest" wird erzeugt und diverse Attribute gesetzt.

```
SessionID = OdsSessionCreate( "TEST", "","", 0 )
WENN SessionID > 0
   id = OdsIECreateElement("#AoTest", 0, 2, 0)
   WENN id > 0
      OdsIESetAttributeTxt( id, "#name", "Test_123", 0)
      OdsIESetAttributeTxt( id, "#description", "Test only", 0)
      OdsIESetAttribute( id, "Charge", 1205, 0)
      OdsIESetAttribute( id, "#version_date", ZeitSystem?(), 0)
      OdsSessionClose(0)
   ENDE
ENDE
```

Siehe auch:

OdsIEGetAttributeTxt, OdsIESetAttribute

OdsInitialize

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Initialisierung des ODS-Systems. Die notwendigen Parameter für die (Neu-)Initialisierung der CORBA-Laufzeitschicht werden definiert.

Deklaration:

```
OdsInitialize ( TxNSAdresse, TxORBParameter, TxPrivateParameter, Null ) -> Fehlercode
```

Parameter:

TxNSAdresse	Adresse des CORBA-Nameservice in der Form "RechnerAdresse:Portnummer"
TxORBParameter	Weitere Parameter für die Initialisierung des CORBA-ORB. Im Allgemeinen leer.
TxPrivateParameter	Weitere Initialisierungsparameter für den ODS-Browser. Im Allgemeinen leer.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
Fehlercode	Erfolg der Funktion
	1 : Funktion erfolgreich ausgeführt
	0 : Fehler

Beschreibung:

Der vom Browser verwendete CORBA-Kern wird unter Angabe notwendiger Initialisierungsparameter (neu) gestartet.

Die anzugebenden Initialisierungsparameter hängen dabei im Wesentlichen von den Eigenschaften des anzusteuernden ODS-Servers ab. Dazu gehört insbesondere die Spezifikation des vom Server verwendeten "CORBA-Nameservice"-Dienstes. Letzterer ist zwingend notwendig, damit der Browser den gewünschten Server finden kann.

Wenn diese Funktion nicht aufgerufen wird, werden die am ODS-Plugin festgelegten Einstellungen verwendet (Dialog "Voreinstellungen/System").

Sinnvoll ist der Aufruf der Funktion im Allgemeinen nur, wenn diese in einer Sequenz vor jeder anderen ODS-Zugriffsfunktion aufgerufen wird. Damit kann ein genau definiertes Verhalten unabhängig von den aktuellen Plugin-Einstellungen erzwungen werden.

Der Aufruf dieser Funktion schließt alle momentan offenen Verbindungen! Der CORBA-Kern wird dann gestoppt und mit den angebenen Parametern neu gestartet.

Im Fehlerfall wird eine 0 zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Multithreading: Alle Funktionen des ODS-Kits dürfen nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

Eine Verbindung zu einem ODS-Server wird geöffnet und die Namen aller enthaltenen Messungen [AoMeasurement] aufgelistet. Der gewünschte ODS-Server hat sich vorher unter dem Namen "ODSTest" an jenem ODS-Namensdienst angemeldet, der auf dem lokalen Rechner am Port 900 läuft.

```
OdsInitialize("localhost:900","","",0)
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurement" ,"", 0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxName = OdsIEGetAttributeTxt( IDList[i], "#name", 0)
      BoxAusgabe( TxName, LEER, "", 1)
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsSessionCreate, OdsPluginSessionConnect

OdsPluginListSelItems

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Die selektierten Instanz-Elemente der aktiven Session des ODS-Plugin's werden ermittelt.

Deklaration:

```
OdsPluginListSelItems ( Null ) -> InstanzElementList
```

Parameter:

Null	Reservierter Parameter. Immer auf 0 zu setzen.
InstanzElementList	ID's aller Instanz-Elemente, die aktuell selektiert sind.

Beschreibung:

Der zurückgegebene Datensatz enthält die ID's aller selektierten Instanz-Elemente. Zur Bestimmung der Anzahl der gefundenen Elemente kann die Lang?() benutzt werden, zum Zugriff auf ein bestimmtes Element der '[ ]' Operator (siehe Beispiel).

Die aktive Session für das Kit muß vorher mittels OdsPluginSessionConnect() oder OdsSessionSelect() auf eine Plugin-Session gesetzt worden sein.

Im Fehlerfall wird ein leerer Datensatz (Länge=0) zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Alle Instanz-Elemente, die momentan in der Baumansicht des ODS-Browsers selektiert sind, werden ermittelt und mit allen Attributen nach FAMOS exportiert.

```
SessionID = OdsPluginSessionConnect( 0 )
WENN SessionID <> 0
   IDList = OdsPluginListSelItems(0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxVarName = OdsIEBuildVarName( IDList[i], 0)
       <TxVarName> = OdsIEGetAttributes( IDList[i], 0)
      i = i+1
   ENDE
ENDE
```

Siehe auch:

OdsPluginSessionConnect, OdsSessionSelect

OdsPluginSessionConnect

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Die momentan vom ODS-Plugin angezeigte Session wird die neue aktive Session für das Kit.

Deklaration:

```
OdsPluginSessionConnect ( Null ) -> SessionID
```

Parameter:

Null	Reservierter Parameter. Immer auf 0 zu setzen.
SessionID	ID der Session
	< 0 : Session-ID
	0 : Fehler.

Beschreibung:

Die Funktion wird verwendet, um anschließend die Funktionen des ODS-Kits auf die gerade im ODS-Browser-Plugin geöffnete Session anwenden zu können.

Eine mit dieser Funktion aktivierte Session muss **nicht** mittels OdsSessionClose() geschlossen werden.

Im Fehlerfall wird eine 0 zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Beispiele:

Alle Instanz-Elemente, die momentan in der Baumansicht des ODS-Browsers selektiert sind, werden ermittelt und mit allen Attributen nach FAMOS exportiert.

```
SessionID = OdsPluginSessionConnect( 0 )
WENN SessionID <> 0
   IDList = OdsPluginListSelItems(0)
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxVarName = OdsIEBuildVarName( IDList[i], 0)
       <TxVarName> = OdsIEGetAttributes( IDList[i], 0)
      i = i+1
   ENDE
ENDE
```

Siehe auch:

OdsPluginListSelItems, OdsSessionCreate, OdsSessionSelect

OdsSessionClose

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Schließen der Verbindung zu einem ODS-Server.

Deklaration:

```
OdsSessionClose ( Option ) -> FehlerCode
```

Parameter:

Option	Auswahl
	0 : Schließen der aktiven Session
	-1 : Schließen aller Sessions, die durch das Kit geöffnet worden sind.
FehlerCode	Erfolg der Funktion
	1 : Funktion erfolgreich durchgeführt
	0 : Fehler.

Beschreibung:

[Option] = 0: Die aktive Session wird geschlossen. Die Session muß vorher mit der Funktion OdsSessionCreate() geöffnet worden sein. Die Funktion darf **nicht** auf Sessions angewendet werden, die mittels OdsPluginSessionConnect() aktiviert worden sind.

Falls mehrere Sessions parallel geöffnet sind, wird nach erfolgreichem Schließen dieser Session diejenige aktiviert, die von den verbleibenden die älteste ist. Wenn Sie statt dessen mit einer anderen Session weiterarbeiten möchten, verwenden Sie anschließend die Funktion OdsSessionSelect().

[Option] = -1: Es werden alle Sessions geschlossen, die vorher mittels OdsSessionCreate() durch das Kit geöffnet worden sind.

Im Fehlerfall wird eine 0 zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Siehe auch:

OdsSessionCreate, OdsSessionSelect, OdsPluginSessionConnect

OdsSessionCreate

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Herstellen einer Verbindung zu einer ODS-Datenquelle (ODS-Server).

Deklaration:

```
OdsSessionCreate ( TxServerName, TxServerOptionen, TxSessionParameter, Null ) -> SessionID
```

Parameter:

TxServerName	Bezeichnung der ODS-Datenquelle (ODS-Server).
TxServerOptionen	Server-spezifische Einstellungen, die zur Verbindungsaufnahme benötigt werden. Im Allgemeinen leer.
TxSessionParameter	Optionen, die beim Öffnen der Session an den Server übergeben werden, z.B. Anwendername und Passwort.
Null	Reservierter Parameter. Immer auf 0 zu setzen.
SessionID	ID der geöffneten Verbindung bzw. 0, wenn die Verbindung nicht hergestellt werden konnte.
	> 0 : Session-ID
	0 : Fehler.

Beschreibung:

Die Funktion dient dazu, eine Verbindung (Session) mit einem ODS-Server herzustellen. Anschließend können die weiteren Funktionen des Kits dazu verwendet werden, um auf die Datenablage zuzugreifen.

Bei erfolgreicher Verbindungsaufnahme wird die hier geöffnete Session zur neuen aktiven Session für das Kit. Nahezu alle Funktionen des ODS-Kits beziehen sich auf die jeweils aktive Session.

Zur Umschaltung zwischen mehreren gleichzeitig geöffneten Sessions verwenden Sie die Funktion OdsSessionSelect(). Diese erhält als Parameter die hier zurückgelieferte Session-ID.

Der erste Parameter (Bezeichnung des ODS-Services) muß in der gleichen Form angegeben werden, wie dieser in dem verwendeten CORBA-Nameservice vom Server angemeldet wurde wurde. Für den seltenen Fall, dass 2 Server den gleichen Namen haben und sich nur über den Zusatz-Parameter "kind" (siehe Ausgabefeld "Art" im "Server verbinden"-Dialog) unterscheiden, muss der Server durch "name.kind" angegeben werden.

Für Sonderanwendungen in gemischten Umgebungen kann auch eine Datei angegeben werden, die die IOR (Interoperable Object Reference) einer existierenden CORBA-Verbindung zu einem ODS-Server enthält. Dadurch kann sich das Kit auf eine existierende Verbindung "aufschalten". Die Syntax für [TxServerName] ist dann "IOR_FILE=<dateiname>", wobei <dateiname> auf eine Datei verweist, die eine gültige IOR enthält.

Die Optionen zum Öffnen der Session (3.Parameter) werden jeweils in der Form "NAME=WERT" angegeben und durch Komma getrennt.

Die folgenden Parameter, u. a. zur Authentifizierung, sind durch ODS definiert, ob diese tatsächlich verwendet/geprüft werden, hängt vom Server ab.

Parameter	Bedeutung/Inhalt
USER	Anwender-Name
PASSWORD	Kennwort
OPENMODE	z.B.: "read","write"

Zusätzlich zu diesen vordefinierten Parametern können hier weitere serverspezifische Optionen gesetzt werden, z.B.:

```
SessionID = OdsSessionCreate( "ODSTest", "","USER=hans, PASSWORD=fgasr, PROJECT=foo", 0 )
```

Wenn Sie eine Verbindung nicht mehr benötigen, sollten Sie diese unbedingt mit der Funktion OdsSessionClose() explizit schließen.

Wenn Sie mit dem Kit eine Session bearbeiten möchten, die gerade mit dem ODS-Browser-Plugin geöffnet ist, verwenden Sie an Stelle von OdsSessionCreate() die Funktion OdsPluginSessionConnect().

Im Fehlerfall wird eine 0 zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Multithreading: Alle Funktionen des ODS-Kits dürfen nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

Eine Verbindung zu einem ODS-Server mit dem Namen "ODSTest" wird geöffnet und die Namen aller enthaltenen Messungen [AoMeasurement] aufgelistet.

```
SessionID = OdsSessionCreate( "ODSTest", "","", 0 )
WENN SessionID > 0
   IDList = OdsIEListByType("#AoMeasurement" ,"", 0)
```

```
   Count = Lang?( IDList)
   i = 1
   SOLANGE i <= Count
      TxName = OdsIEGetAttributeTxt( IDList[i], "#name", 0)
      BoxAusgabe( TxName, LEER, "", 1)
      i = i+1
   ENDE
   OdsSessionClose(0)
ENDE
```

Siehe auch:

OdsSessionSelect, OdsSessionClose, OdsPluginSessionConnect

OdsSessionSelect

Anwendungsbereich: ASAM-ODS Browser

Verfügbar ab: Enterprise Edition (ODS-Browser-Kit)

Umschalten der aktuellen Session.

Deklaration:

```
OdsSessionSelect ( SessionID ) -> FehlerCode
```

Parameter:

SessionID	ID der zu aktivierenden Session.
FehlerCode	Erfolg der Funktion
	1 : Funktion erfolgreich durchgeführt
	0 : Fehler.

Beschreibung:

Nahezu alle Kit-Funktionen beziehen sich die auf die aktive Session. Wenn mehrere Sessions parallel geöffnet sind, kann mit dieser Funktion die aktive Session gewechselt werden. Nachfolgende Aufrufe von Session-bezogenen Kit-Funktionen beziehen sich dann auf die hier aktivierte Session.

Der erste Parameter [SessionID] muß bei einem vorhergehenden Aufruf von OdsSessionCreate() oder OdsPluginSessionConnect() zurückgegeben worden sein.

Im Fehlerfall wird eine 0 zurückgeliefert. Die Fehlerursache kann mit den Funktionen OdsGetLastErrorTxt() bzw. OdsGetLastErrorCode() ermittelt werden.

Siehe auch:

OdsSessionCreate, OdsPluginSessionConnect

OnError

Die Funktion legt fest, wie FAMOS im Falle eines Fehlers bei der Sequenzausführung reagieren soll.

Deklaration:

```
OnError ( TxModus [, TxFehlermeldung] [, TxFehlerVariableName] [, FehlerWert] )
```

Parameter:

TxModus	Legt fest, ob/wie bei nachfolgend auftretenden Fehlern die Sequenzausführung fortgesetzt werden soll.
	"" : Standardverhalten. Ein Fehler wird mit einer Meldungsbox angezeigt und die Sequenzausführung abgebrochen. Dies entspricht dem Verhalten in älteren FAMOS-Versionen bis einschließlich Version 7.4.
	"Default" : Gleichbedeutend mit "" (Standard)
	"Return" : Keine Fehlerbox, Sprung ans Ende der Sequenz. Ein möglicherweise vorhandener Aufrufer (Sequenz/Dialog/Panel) wird fortgesetzt. Der Fehler kann ggf. in der aufrufenden Sequenz mit GetLastError() abgefragt werden.
	"ReturnFail" : Keine Fehlerbox, Sprung ans Ende der Sequenz. Der Fehler wird an den Aufrufer zurückgeliefert.
	"ResumeNext" : Keine Fehlerbox. Weiter mit nächster Zeile. Der Fehler kann mit GetLastError() abgefragt werden.
	"ResumeEnd" : Keine Fehlerbox, Sprung zum nächsten END-Befehl. Besonders geeignet, um bei Fehlern in FOR- oder FOREACH-Schleifen mit dem nächsten Schleifendurchlauf fortzufahren. Falls kein nachfolgendes END existiert, dann Sprung ans Ende der Sequenz. Der Fehler kann mit GetLastError() abgefragt werden.
TxFehlermeldung	Auszugebende Fehlermeldung. Wenn nicht angegeben (oder wenn leerer Text), wird die originale Fehlermeldung (Syntaxfehler des Formelinterpreters oder Laufzeitfehler bei der Ausführung einer Funktion) verwendet. (optional)
TxFehlerVariableName	Wenn angegeben, dann wird im Fehlerfall die Variable mit diesem Namen mit dem im nächsten Parameter angegebenen Wert beschrieben. (optional)
FehlerWert	Die im vorherigen Parameter spezifizierte Variable wird im Fehlerfall auf den hier angegebenen Wert (erlaubt ist ein Einzelwert oder ein Text) gesetzt. (optional)

Beschreibung:

Bei der Ausführung von Sequenzen können Fehler auftreten, die entweder durch den Formelinterpreter ("Syntaxfehler", z.B. unbekannte Funktionsnamen, falsche Anzahl von Parametern) oder bei der Ausführung von Funktionen ("Laufzeitfehler", z.B. falscher Typ oder nicht erlaubter Inhalt von Funktionsparametern) entstehen können. Standardmäßig werden diese Fehler durch eine Meldungsbox angezeigt und die weitere Ausführung abgebrochen. Dieses Verhalten kann durch OnError() umkonfiguriert werden.

Es wird empfohlen, die hier definierten erweiterten Fehlermodi sparsam, mit Bedacht und gut getestet einzusetzen. Unerwartete, möglicherweise kritische Fehler werden eventuell "verdeckt". Durch den Sprung im Fehlerfall wird der normale Programmablauf außer Kraft gesetzt, so dass z.B. Aufräumarbeiten (Löschen von Variablen, Schließen von Dateien etc.) möglicherweise nicht ausgeführt werden. Es ist im Allgemeinen sinnvoller, mögliche Fehlerursachen explizit vorher abzutesten und im Programmablauf entsprechend zu reagieren. So kann z.B. durch die Funktion VerifyVar() geprüft werden, ob der Datentyp und der Aufbau einer Variablen den Anforderungen der nachfolgenden Auswertung entspricht.

- Beim Erstellen und Testen von Sequenzen und Sequenzfunktionen ist es unter Umständen hilfreich, beim Auftreten eines Fehlers immer zu unterbrechen, auch wenn die aktuelle OnError()-Einstellung ein anderes Verhalten vorsieht. Um dies zu erreichen, können Sie die Option "Bei Fehler immer unterbrechen" im Menü "Sequenz"/"Ausführen" aktivieren.
- Die hier festgelegten Einstellungen gelten nur für die aktuelle Sequenz, sie werden also auch nicht an von dieser Sequenz aufgerufene Untersequenzen oder Sequenzfunktionen "vererbt". Nach Ende der aktuellen Sequenz gelten wieder die vorher wirksamen Einstellungen.
- Sequenzfunktionen mit [TxModus]="ReturnFail": Am Ende der Sequenzfunktion wird ein eventueller Rückgabewert nicht erzeugt, In/Out-Parameter werden nicht aktualisiert.
- Sequenzfunktionen mit [TxModus]="Return": Die Sequenzfunktion verhält sich, als ob kein Fehler aufgetreten wäre - der Rückgabewert wird erzeugt, In/Out-Parameter werden aktualisiert. Bei Sequenzfunktionen mit diesem Modus und Rückgabewert ist daher zu beachten, dass die Return-Variable bereits vor dem Auftreten potentieller Fehler angelegt und z.B. auf einen Standardwert gesetzt wird.
  Dies kann z.B. durch Erzeugung der Return-Variablen gleich am Sequenzbeginn oder durch Verwendung der Return-Variablen als [TxFehlerVariable] in dieser Funktion sichergestellt werden (siehe Beispiel #3).
- Durch den Aufruf der Funktion wird der interne, mit GetLastError() abfragbare, Fehlerspeicher gelöscht.
- In jedem Modus wird die Fehlermeldung auch im FAMOS-Ausgabefenster angezeigt.

Beispiele:

In einer Sequenz werden in einer Schleife alle Dateien aus einem Verzeichnis geladen. Es wird erwartet, dass jede Datei einen Kanal mit dem Namen 'channel1' enthält, dieser wird geglättet und in einem anderen Verzeichnis gespeichert. Falls eine Datei nicht geladen werden kann oder keinen Kanal 'channel1' enthält, wird dies zunächst ignoriert und mit der nächsten Datei fortgesetzt. Am Ende der Sequenz wird ggf. ein Hinweis ausgegeben, dass nicht alle Dateien verarbeitet werden konnten.

```
error = 0
```

```
inPath = "c:\inbox"
outPath = "c:\outbox"
filenames = FsGetFileNames(inPath, "*.dat", 0, 0, 0)
SetOption("Func.ErrorBoxes", "Yes")  ; Dateifunktionen erzeugen expliziten Fehler!
OnError("ResumeEnd", "", "error", 1)
FOR i = 1 TO TxArrayGetSize(filenames)
   FileLoad(fileNames[i], "", 0)
   channel1 = Smo(channel1, 0.5)
   FileSave(outPath + FsSplitPath(fileNames[i], 4),"", 0, channel1)
   DELETE channel1
END
IF error
   BoxMessage("Fehler", "Es konnten nicht alle Dateien verarbeitet werden!", "!1")
END
```

Die in FAMOS eingebaute Funktion zur Wurzel-Berechnung Sqrt() liefert bei negativen Eingangswerten eine 0 und gibt eine entsprechende Warnung aus. Die folgende Sequenzfunktion gibt statt dessen einen Fehler zurück.

```
; Deklaration: !Sqrt_Strict(Par) => Result
OnError("ReturnFail")
IF min(Par) < 0
   ThrowError("Der Parameter enthält negative Werte!")
END
Result = Sqrt(par)
```

Die nachfolgende Sequenzfunktion prüft, ob sich ein Datensatz in dem durch andere 2 Datensätze definierten Toleranzband befindet. Wenn die Funktion wegen unpassender Parameter nicht ausgeführt werden kann, wird dies durch einen speziellen Rückgabewert signalisiert.

```
; Deklaration: !ToleranceBand(TestData, Lower, Upper) => Result
; Result = 0:  OK
; Result = 1:  Verletzung des Bandes nach oben
; Result = 2:  Verletzung des Bandes nach unten
; Result = 3:  Verletzung des Bandes nach oben und unten
; Result = -1: Fehler: Eingangsdaten passen nicht zusammen (bzgl. ihrer x-Achse) oder haben Events/Segmente

OnError("Return", "", "Result", -1)
Verify(Leng?(TestData)=Leng?(Lower) AND Leng?(TestData)=Leng?(Upper))
Verify(xdel?(TestData)=xdel?(Lower) AND xdel?(TestData)=xdel?(Upper))
Verify(xoff?(TestData)=xoff?(Lower) AND xoff?(TestData)=xoff?(Upper))

TooLarge = Max(TestData- Upper) > 0   ; liefert z.B. Fehler bei segmentierten Daten
TooSmall = (Max(Lower - TestData) > 0) * 2
Result = TooLarge + TooSmall
```

Eine Sequenz ruft nacheinander 3 Sequenzfunktionen auf, die voneinander unabhängige und länger dauernde Auswertungen ausführen. Falls eine Auswertung auf einen Fehler läuft, sollen die nachfolgenden Auswertungen trotzdem ausgeführt werden. Am Ende wird eine Meldungsbox angezeigt, wenn nicht alle Auswertungen erfolgreich waren. Im Ausgabefenster kann der Anwender anhand der spezifischen Fehlermeldungen erkennen, welche Auswertungen fehlgeschlagen sind. (Die Sequenzfunktionen sind durch entsprechenden Einsatz von OnError("ReturnFail",...) so geschrieben, dass sie kritische Fehler an den Aufrufer zurückliefern, aber nicht die Ausführung abbrechen.)

```
error =  0
OnError("ResumeNext", "Fehler in #1", "error", 1)
!Auswertung_1()

OnError("ResumeNext", "Fehler in #2", "error", 1)
!Auswertung_2()

OnError("ResumeNext", "Fehler in #3", "error", 1)
!Auswertung_3()

IF error
   BoxMessage("Fehler", "Mindestens 1 Auswertung konnte nicht ausgeführt werden!", "!1")
END
```

Siehe auch:

Verify, VerifyVar, ThrowError, GetLastError

OR

Logischer "ODER"-Operator

Deklaration:

```
Operand1 OR Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Operand2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

"ODER"-Verknüpfung zweier Zahlen. Das Ergebnis ist 1, wenn mindestens einer der Operanden ungleich 0 ist. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte und auf Datensätze angewendet werden. Bei Datensätzen erfolgt eine punktweise Verknüpfung.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Das Maximum eines Datensatzes wird bestimmt und auf Verlassen eines Toleranzbandes getestet.

```
Channel1 = ...
Maximum = max(Channel1)
IF Maximum < 21 OR Maximum > 34
   BoxMessage("Achtung", "Wert außerhalb Toleranz", "!1")
END
```

Zwei digitale Datensätze werden verknüpft. Der Ergebnisdatensatz ist überall dort 1, wo mindestens 1 Operandendatensatz den Wert 1 aufweist.

```
Result = (DigChannel1 OR DigChannel2)
```

Siehe auch:

AND, NOT, XOR

OtrEncoderRevs01

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Das abgetastete, rechteckige Signal eines Inkrementalgebers (Encoder) wird zur Bestimmung der Drehzahl benutzt. Das Signal enthält das abgetastete Rechtecksignal, das nur aus Nullen und Einsen besteht.

Deklaration:

```
OtrEncoderRevs01 ( Encodersignal, PulseProUmdrehung, Null ) -> Drehzahlverlauf
```

Parameter:

Encodersignal	Das Rechtecksignal eines Encoders
PulseProUmdrehung	Die Anzahl der Pulse (Striche) pro Umdrehung. >= 1. Die Anzahl muss nicht ganzzahlig sein.
Null	Stets null
Drehzahlverlauf	

Beschreibung:

Bei jedem Übergang des Signals von "Kleiner gleich 0.0" auf "Größer 0.0" wird angenommen, dass der Encoder sich um ein Inkrement weiter gedreht hat. Enthält das Signal die Wertefolge { 0, 0, 1, 1, 1, 0, 0, 1, 1, 0 }, so werden 2 Pulse detektiert. Die Funktion liefert als Ergebnis einen Drehzahlverlauf, skaliert in Umdrehungen pro Minute (U/min) zurück. Liegen die gemessenen Werte von ihren Pegeln her nicht passend, kann z.B. mit dem Schmitt-Trigger stri() die nötige Signalformung vorgenommen werden.

Die Wirkung dieser Funktion wird durch OtrTachoMode () nicht beeinflusst.

Hinweis: Für andere Arten von Tachosignalen sind nur OtrTachoMode() und OtrTachoToSpeed() geeignet.

Beispiele:

Zur Messung der Drehzahl steht nur ein binärer Eingang (ein digital IN, was nur 0 oder 1 liefert) zur Verfügung. Das Rechteck-Signal des benutzten Encoders Rectangle_01 wird mit einer festen Abtastzeit abgetastet. Der Geber hat 100 Impulse pro Umdrehung und wurde mit 0.1 ms sehr schnell abgetastet. Die Drehzahl reicht von 10U/min .. 2000U/min. Die Frequenz der Rechtecke reicht von 17Hz bis 3.3kHz, was mit 10kHz gerade noch abgetastet werden kann.

```
speed = OtrEncoderRevs01 ( Rectangle_01, 100, 0 )
```

Im folgenden Beispiel wird der Encoder mit einem analogen Eingang abgetastet. Die erfasste Spannung VoltageInput liegt bei LOW um 0.8V, bei HIGH und 4.7V. Ein Schmitt-Trigger wird zur Signalformung benutzt. Der Geber hat 128 Striche.

```
Rectangle_01 = stri ( VoltageInput,  3, 2)
speed = OtrEncoderRevs01 ( Rectangle_01, 128, 0 )
```

Siehe auch:

OtrTachoMode, OtrTachoToSpeed, OtrTachoToDist

OtrFrequLine

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Frequenzlinienbestimmung: Zu einem Signal, das eine sinusförmige Schwingung mit fester Periodendauer enthält, wird Betrag oder Phase dieser Schwingung bestimmt.

Deklaration:

```
OtrFrequLine ( Schwingungssignal, Periodenlänge, Periodenanzahl zur Mittelung, Option ) -> Ergebnis
```

Parameter:

Schwingungssignal	Signal, z.B. mit periodischer Schwingung
Periodenlänge	Anzahl der Abtastwerte in einer Periode, >=2
Periodenanzahl zur Mittelung	Mittelung über so viele Perioden, >=1
Option	Was ist zu berechnen?
	0 : Betrag (Effektivwert) ermitteln
	1 : Phase (in Grad) ermitteln
Ergebnis	Ermittelter Betrags- oder Phasenverlauf

Beschreibung:

Die Funktion bestimmt in je einem Intervall der Länge "PeriodenLänge * PeriodenAnzahl" den Wert für Betrag oder Phase der Schwingung. Dabei passt genau eine Anzahl von "PeriodenAnzahl" Schwingungen in dieses Intervall. Die Dauer der Schwingung muss immer fest und konstant sein. Sie muss nicht unbedingt eine ganze Anzahl von Abtastschritten lang sein. Sie ist bestimmt ist durch "PeriodenLänge", die Periodendauer dividiert durch die Abtastzeit.

Die Funktion bestimmt eine Linie des diskreten Fourier-Spektrums (DFT) bei Rechteck-Fensterung.

Falls im Signal noch merkliche andere Frequenzanteile enthalten sind, sollte eine große Periodenanzahl gewählt werden, um deren verfälschenden Einfluss zu verringern. Ggf. kann ein Bandpassfilter vorgeschaltet werden.

Das Produkt aus PeriodenLänge und PeriodenAnzahl darf 2e9 nicht überschreiten.

Die Phase wird im Bereich -180 Grad .. +180 Grad bestimmt. Der Wert der Phase beträgt 0 Grad bei einer cos-Schwingung, -90 Grad bei einer sin-Schwingung.

Falls die Periodenlänge keine ganze Zahl ist, muss jedoch das Produkt aus Periodenlänge und Periodenanzahl eine ganze Zahl von Messwerten sein.

Beispiele:

Ein Schwingungssignal wird über dem Winkel abgetastet (vib_revs), sodass alle Schwingungsanteile bis zur 8. Ordnung enthalten sind. Das Signal enthält also 16 Punkte pro Umdrehung. Die Phase der 1. Ordnung soll bestimmt werden. Alle 5 Umdrehungen ist ein Wert für die Phase gewünscht.

```
Phase = OtrFrequLine ( vib_revs, 16, 5, 1 )
```

Siehe auch:

OtrOrderSpecFromFFT

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Das Ordnungsspektrum, abhängig von der Drehzahl, wird aus dem FFT-Spektrum, abhängig von der Drehzahl, bestimmt. Das FFT-Spektrum ist als Effektivwertspektrum gegeben.

Deklaration:

```
OtrOrderSpecFromFFT ( FFT_Spektrum, Auflösung, OrdnungMax, Berechnung ) -> Ergebnis
```

Parameter:

FFT_Spektrum	FFT-Spektrum, abhängig von der Drehzahl. Angegeben wird ein segmentierter Datensatz. Jedes Segment ist dabei in Spektrum.
Auflösung	Die Auflösung des Ordnungsspektrums. Z.B. 0.1, wenn 0.1 Ordnungen der Abstand der Linien im Ordnungsspektrum ist. Die Auflösung ist i. Allg. ein ganzzahliger Teiler von 1.0, also [1, 1/2, 1/3, 1/4, ..]
OrdnungMax	Die höchste im Ordnungsspektrum angezeigte Ordnung(slinie)
Berechnung	Art der Berechnung. Nach welcher Methode wird aus vielen Spektrallinien des FFT-Spektrums eine Ordnungslinie gebildet.
	0 : Maximum wird benutzt
	1 : Leistung bleibt erhalten
	2 : Abtastung (Interpolation)
Ergebnis	Ordnungsspektrum, abhängig von der Drehzahl

Beschreibung:

Die Funktion bestimmt ein Ordnungsspektrum. Aus dem FFT-Spektrum ist das Ordnungsspektrum (näherungsweise) bestimmbar, wenn die Drehzahl bekannt ist. Dabei entspricht eine Frequenz von (Drehzahl / 60.0) der 1. Ordnung.

Der übergebene Datensatz mit dem FFT-Spektrum ist segmentiert. Ein Segment ist eine FFT. Für jedes Segment gilt eine Drehzahl. Die Drehzahl sollte nicht 0.0 sein. Die Drehzahl muss in U/min (Umdrehungen pro Minute) skaliert sein. Die z-Koordinate des Datensatzes legt die Drehzahl fest.

Die Funktion staucht bzw. streckt das FFT-Spektrum. Es lohnt sich deshalb nicht, die Auflösung beliebig klein zu machen und die maximale Ordnung beliebig groß. Das erhöht nur die Rechenzeit und den Speicherplatzbedarf, ohne den Informationsgehalt zu steigern.

Die erste Linie im FFT-Spektrum (das ist der Offset bzw. Gleich-Anteil bzw. DC) muss bei f = 0.0 Hz liegen. Das resultierende Ordnungsspektrum hat den selben DC-Anteil.

Das vorliegende FFT-Spektrum sollte den Betrag (z.B. Effektivwert) der Frequenzlinien enthalten.

Art der Berechnung

- Bei der Berechnungsart "Maximum" wird das Maximum aller Linien des FFT-Spektrums gebildet, die zu einer Ordnungslinie beitragen. Bei der Berechnungsart "Leistung" wird die resultierende Ordnungslinie so bestimmt, dass die in ihr enthaltene Leistung der Summe der Leistungen aller zu ihr beitragenden Linien des FFT-Spektrums entspricht. Damit bleibt die spektrale Leistungsdichte erhalten. Bei der Berechnungsart "Abtastung" werden alle Spektrallinien des FFT-Spektrums durch gerade Linien verbunden (also linear interpoliert) gedacht. Dieser Polygonzug wird an den entsprechenden Stellen abgetastet.

Hinweis:

Zu beachten ist, dass der Algorithmus einfach und ungenau ist. Die präzise und aufwendige Bestimmung des Ordnungsspektrums erfolgt mit der Funktion OtrOrderSpec(), bei der mit Tracking-Filter und Nachabtastung über dem Winkel gearbeitet wird.

Beispiele:

Zunächst wird das drehzahlabhängige FFT-Spektrum aus der Schwingung Vibration und der Drehzahl speed ermittelt. Anschließend soll zur Rechenzeitersparnis das Ordnungsspektrum aus dem bereits ermittelten FFT-Spektrum gebildet werden. Die größten Spitzen im FFT-Spektrum sollen dabei erhalten bleiben (also mit gleicher Amplitude wieder im Ordnungsspektrum auftreten).

```
FFT_Spectrum = OtrRpmSpectrum ( Vibration, speed, 1000, 6000, 100, 1024, 0, 1, 0)
Resolution = 0.1 ; Auflösung des Spektrums, also 1/10 Ordnungen sichtbar
OrderMax = 6.0 ; bis zu dieser Ordnung sollen Linien im Ordnungsspektrum angezeigt werden.
Calc = 0 ; Standard. Maximum nutzen. Spitzenwerte bleiben erhalten
OrderSpec = OtrOrderSpecFromFFT( FFT_Spectrum, Resolution, OrderMax, Calc)
```

Siehe auch:

OtrRpmSpectrum, OtrOrderSpec

OtrOrderSpectrum

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Das Ordnungsspektrum wird aus den Zeitverläufen von Schwingung und Tachosignal in Abhängigkeit von der Drehzahl bestimmt. Der gewünschte Drehzahlbereich wird in Klassen gleicher Breite aufgeteilt.

Deklaration:

```
OtrOrderSpectrum ( Schwingung, Tachosignal, Upm_Min, Upm_Max, Upm_Klassenbreite, Auflösung, OrdnungMax, Fenstertyp, Mittelungsart ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Tachosignal	Tachosignal, Deutung entsprechend OtrTachoMode(). Standard ist ein Drehzahlverlauf, skaliert in U/min.
Upm_Min	Unteres Ende des gewünschten Drehzahlbereichs. In U/min skaliert.
Upm_Max	Oberes Ende des gewünschten Drehzahlbereichs. In U/min skaliert.
Upm_Klassenbreite	Breite einer Drehzahl-Klasse (Intervall). In U/min skaliert.
Auflösung	Die Auflösung des Ordnungsspektrums. 0.1, wenn 0.1 Ordnungen der Abstand der Linien im Ordnungsspektrum ist. Die Auflösung muss ein ganzzahliger Teiler von 1.0 sein, also [1, 1/2, 1/3, 1/4, ..]
OrdnungMax	Die höchste im Ordnungsspektrum angezeigte Ordnung(slinie).
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Mittelungsart	Wie werden alle Spektren zur selben Drehzahlklasse gemittelt?
	0 : arithmet. Mittel
	1 : arithmet. Mittel, 50% Überlappung
	2 : arithmet. Mittel, 75% Überlappung
	3 : Maximum
	4 : Maximum, 50% Überlappung
	5 : Maximum, 75% Überlappung
	6 : Minimum
	7 : Minimum, 50% Überlappung
	8 : Minimum, 75% Überlappung
	9 : das erste
	10 : arithmet. Mittel, streng
	11 : arithmet. Mittel, streng, 50% Überlappung
	12 : arithmet. Mittel, streng, 75% Überlappung
	13 : Maximum, streng
	14 : Maximum, streng, 50% Überlappung
	15 : Maximum, streng, 75% Überlappung
	16 : Minimum, streng
	17 : Minimum, streng, 50% Überlappung
	18 : Minimum, streng, 75% Überlappung
	19 : das erste, streng
	20 : RMS

	21 : RMS, 50% Überlappung
	22 : RMS, 75% Überlappung
	23 : RMS, 90% Überlappung
	24 : RMS, 95% Überlappung
	25 : RMS, 98% Überlappung
	26 : RMS, 99% Überlappung
Ergebnis	Ordnungsspektrum abhängig von der Drehzahl

Beschreibung:

Für das Abtasten wird der Betrag der Drehzahl benutzt, für das Zuordnen zu einer Drehzahlklasse aber die Originaldrehzahl.

Der Kehrwert der Auflösung gibt an, über wie viele Umdrehungen das Ordnungsspektrum gebildet wird. Z.B. bei Auflösung = 0.1 wird jedes Spektrum aus 10 Umdrehungen bestimmt.

Die Spektrallinien sind als Effektivwerte angegeben.

Da intern die FFT mit einer etwas größeren Anzahl von Daten (einer Zweierpotenz) berechnet wird, werden einige Spektrallinien abgeschnitten. Im sichtbaren Spektrum ist also nicht mehr die volle Leistung enthalten. Das Rechteck-Fenster für die FFT ist sehr empfohlen, wenn ein hohe Frequenz-Auflösung gewünscht ist.

Die Mittelung arbeitet auf dem Betragsspektrum.

Strenge Ausführung der Mittelung:

- Spektren werden nur berücksichtigt, wenn während der ganzen Zeit für ein Spektrum die Drehzahl innerhalb einer Drehzahlklasse ist. Die Drehzahl sollte sich also nur langsam ändern. Wenn sich die Drehzahl schnell ändert und die Spektren sollen berücksichtigt werden, ist ein entsprechend großes Upm_Klassenbreite zu wählen!

Nicht strenge Ausführung der Mittelung:

- Die mittlere Drehzahl während eines Spektrums bestimmt die Drehzahlklasse. Das kann sehr ungenau werden, wenn sich die Drehzahl sehr schnell ändert.

Ein mitlaufender Butterworth-Tiefpass wird als Antialiasing-Filter eingesetzt. Das so gefilterte Signal wird über dem Winkel abgetastet. Dann wird eine FFT berechnet. Aus dieser Vorgehensweise folgt, dass beim Abtasten zwischendurch deutlich höhere Ordnungslinien im Signal vorhanden sind als letztendlich im resultierenden Spektrum. Die 3dB-Ordnung des Antialiasing-Filter liegt deutlich hinter den im resultierenden Spektrum liegenden Ordnungslinien.

Die Funktion arbeitet sinnvoll für:

- OrdnungMax < 10 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl.

- Die Drehzahl sollte sich nur langsam ändern. Die Drehzahl sollte nicht wesentlich unter 1% der maximal möglichen sinken. Ist die Drehzahl oder OrdnungMax wesentlich größer als sinnvoll, steigt die Rechenzeit enorm, ohne den Informationsgehalt zu erhöhen.
- Aus dem Tachosignal muss durch Integration (Aufsummieren) der Drehwinkel ermittelt werden. Ist das Tachosignal selbst ein Drehzahlverlauf (und kein Pulssignal), muss deshalb die Drehzahl sehr genau vorliegen.
- Die Abtastzeit des Tachosignals darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Falls in einer Drehzahlklasse kein Spektrum bestimmt wird, wird dieses Spektrum mit Nullen gefüllt.
- Das Ergebnis ist ein segmentierter Datensatz. Jedes Segment ist ein Ordnungsspektrum.
- Vor Aufruf dieser Funktion sollte wenigstens einmal OtrTachoMode() aufgerufen werden.

Beispiele:

Aus dem zeitlichen Verlauf einer Schwingung vib und der Drehzahl speed soll das Ordnungsspektrum abhängig von der Drehzahl bestimmt werden. Die Zeitsignale sind mit 0.2 ms abgetastet.

```
OtrTachoMode ( 0, 0, 0, 0)
rpm_Min = 1000.0 ; Minimum des Drehzahlbereichs
rpm_Max = 6000.0 ; Maximum des Drehzahlbereichs
rpm_Delta = 100.0 ; Breite der einzelnen Drehzahlklassen
Resolution = 0.1 ; Auflösung des Spektrums, also 1/10 Ordnungen sichtbar, Berechnung über 10 Umdrehungen
OrderMax = 6.0 ; bis zu dieser Ordnung sollen Linien im Spektrum angezeigt werden.
Windowtype = 0 ; 0 Standard (Rechteck)
AvgType = 0 ; 0 (arithmet. Mittel)
OSpectrum = OtrOrderSpectrum ( vib, speed, rpm_Min, rpm_Max, rpm_Delta, Resolution, OrderMax, Windowtype, AvgType )
```

Hier gilt: OrderMax = 6.0 < 10 / (0.0002 * 6000) = 8.3

Mit OtrTachoMode() wird vorher die Art des verwendeten Tachosignals festgelegt.

Siehe auch:

OtrTachoMode, OtrRpmOrder, OtrRpmSpectrum, OtrTimeOrderSpectrum, OtrOrderSpecFromFFT

OtrResample

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Abtasten eines Schwingungssignals über dem Winkel, wobei das Tachosignal gegeben ist.

Deklaration:

```
OtrResample ( Schwingung, Tachosignal, OrdnungMax, Interpolation ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Tachosignal	Tachosignal, Deutung entsprechend OtrTachoMode(). Standard ist ein Drehzahlverlauf, skaliert in U/min.
OrdnungMax	Maximal enthaltene Ordnung(slinie) im Resultat
Interpolation	Wie werden beim Abtasten Zwischenwerte gebildet?
	0 : Konstante Interpolation (Treppenstufen)
	1 : Lineare Interpolation
	2 : Kubische Interpolation
Ergebnis	Über dem Winkel abgetastetes Schwingungssignal

Beschreibung:

Der resultierende Signalverlauf über dem Winkel ist so in x-Richtung skaliert, dass er die Anzahl der zurückgelegten Umdrehungen zählt. Dabei beginnt die x-Koordinate bei 0, ist nach einer halben Umdrehung bei 0.5, nach einer ganzen Umdrehung bei 1.0, nach 2 ganzen Umdrehungen bei 2.0 usw.

- Abtastintervall des Resultats: 0.5 / OrdnungMax
- Der Absolutbetrag der Drehzahl wird benutzt.
- Die Funktion enthält kein Antialiasing-Filter. Ein vorheriger Aufruf von OtrTrackingLowPass() ist nötig.

Die Funktion arbeitet sinnvoll für:

OrdnungMax <= 30 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl.

- Die Drehzahl sollte sich nur langsam ändern. Die Drehzahl sollte nicht wesentlich unter 1% der maximal möglichen sinken. Ist die Drehzahl wesentlich größer als sinnvoll, steigen Rechenzeit und Speicherbedarf enorm, ohne den Informationsgehalt zu erhöhen.
- Aus dem Tachosignal muss durch Integration (Aufsummieren) der Drehwinkel ermittelt werden. Ist das Tachosignal selbst ein Drehzahlverlauf (und kein Pulssignal), muss deshalb die Drehzahl sehr genau vorliegen. Dabei ist zu beachten, dass bereits ein leicht falsches oder ungenaues Drehzahlsignal aufintegriert zu einer merklichen Winkelabweichung führen kann, was zu einer (schleichenden) immer stärkeren Phasenabweichung führt.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Vor Aufruf dieser Funktion sollte wenigstens einmal OtrTachoMode() aufgerufen werden.

Beispiele:

Ein Schwingungssignal vib ist mit 0.5 ms abgetastet. Die Drehzahl speed kann bis 3000 U/min hochgehen. Die Schwingung soll über dem Winkel abgetastet werden. Bis zur 15. Ordnung sollen Anteile enthalten sein.

```
OtrTachoMode ( 0, 0, 0, 0)
tlp = OtrTrackingLowPass ( vib, speed, 8.0, 4 ) ; Antialiasing-Filter
omax = 15.0
res = OtrResample ( tlp, speed, omax, 2 )
```

Es gilt: OrdnungMax = 15.0 <= 30 / (0.0005 * 3000) = 20.0.

Das Resultat erhält eine Auflösung von 0.5 / OrdnungMax = 0.0333 Umdrehungen. Es hat 30 Abtastwerte pro Umdrehung. Das Antialiasing Filter ist so dimensioniert, dass es bei der 8. Ordnung um 3 dB dämpft, bei der 14.4ten Ordnung um 20 dB. Bei der 6.1ten Ordnung beträgt der Amplitudenfehler bereits weniger als 5%. Eine Interpolation höherer Ordnung wird benutzt.

Mit OtrTachoMode() wird vorher die Art des verwendeten Tachosignals festgelegt.

Siehe auch:

OtrResampleAAF, OtrTrackingLowPass, OtrTachoMode

OtrResampleAAF

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Abtasten eines Schwingungssignals über dem Winkel, wobei das Tachosignal gegeben ist und ein mitlaufendes Antialiasing-Filter angewendet wird.

Deklaration:

```
OtrResampleAAF ( Schwingung, Tachosignal, OrdnungMax, Ordnung3dB, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Tachosignal	Tachosignal, Deutung entsprechend OtrTachoMode(). Standard ist ein Drehzahlverlauf, skaliert in U/min.
OrdnungMax	Maximal enthaltene Ordnung(slinie) im Resultat
Ordnung3dB	Ordnung(slinie), bei der der Tiefpass um 3dB dämpft.
FilterOrdnung	Die Filter-Ordnung des Tiefpassfilters (1 ... 10)
Ergebnis	Über dem Winkel abgetastetes Schwingungssignal

Beschreibung:

Der resultierende Signalverlauf über dem Winkel ist so in x-Richtung skaliert, dass er die Anzahl der zurückgelegten Umdrehungen zählt. Dabei beginnt die x-Koordinate bei 0, ist nach einer halben Umdrehung bei 0.5, nach einer ganzen Umdrehung bei 1.0, nach 2 ganzen Umdrehungen bei 2.0 usw.

- Abtastintervall des Resultats: 0.5 / OrdnungMax
- Der Absolutbetrag der Drehzahl wird benutzt.
- Ein Butterworth-Tiefpass wird als Antialiasing-Filter eingesetzt.
- Zur Bestimmung von Zwischenwerten wird kubisch interpoliert.

Die Funktion arbeitet sinnvoll für:

OrdnungMax < 24 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl. Dabei ist zu beachten, dass die höchstfrequenten Anteile im resultierenden Signal bereits sehr stark gedämpft sind.

Ordnung3dB << OrdnungMax

<< soll bedeuten: deutlich kleiner.

- Die Drehzahl sollte sich nur langsam ändern. Die Drehzahl sollte nicht wesentlich unter 1% der maximal möglichen sinken. Ist die Drehzahl wesentlich größer als sinnvoll, steigen Rechenzeit und Speicherbedarf enorm, ohne den Informationsgehalt zu erhöhen.
- Aus dem Tachosignal muss durch Integration (Aufsummieren) der Drehwinkel ermittelt werden. Ist das Tachosignal selbst ein Drehzahlverlauf (und kein Pulssignal), muss deshalb die Drehzahl sehr genau vorliegen. Dabei ist zu beachten, dass bereits ein leicht falsches oder ungenaues Drehzahlsignal aufintegriert zu einer merklichen Winkelabweichung führen kann, was zu einer (schleichenden) immer stärkeren Phasenabweichung führt.
- Die obere Grenzfrequenz des Tiefpassfilters muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa (0.4 * Abtastfrequenz) kann keine Filterung mehr durchgeführt werden.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Die Funktion kann durch Aufrufe von OtrTrackingLowPass() und OtrResample() nachgebildet werden.
- Vor Aufruf dieser Funktion sollte wenigstens einmal OtrTachoMode() aufgerufen werden.

Beispiele:

Ein Schwingungssignal vib ist mit 1 ms abgetastet. Die Drehzahl speed kann bis 4500 U/min hochgehen. Die Schwingung soll über dem Winkel abgetastet werden. Bis zur 5. Ordnung sollen Anteile enthalten sein.

```
OtrTachoMode ( 0, 0, 0, 0)
omax = 5.0
o3dB = 2.7 ; Der Tiefpass dämpft um 3 dB bei dieser Ordnung
fo = 8 ; Ein Tiefpassfilter 8. Ordnung wird benutzt
res = OtrResampleAAF ( vib, speed, omax, o3dB, fo )
```

Es gilt: OrdnungMax = 5.0 <= 24 / (0.001 * 4500) = 5.33

Das Resultat erhält eine Auflösung von 0.5 / OrdnungMax = 0.1 Umdrehungen. Es hat 10 Abtastwerte pro Umdrehung. Das Antialiasing Filter ist so dimensioniert, dass es bei der 2.7ten Ordnung um 3 dB dämpft, bei der 5. Ordnung um 60 dB. Bei der 2.3ten Ordnung beträgt der Amplitudenfehler bereits weniger als 5%.

Mit OtrTachoMode() wird vorher die Art des verwendeten Tachosignals festgelegt.

Siehe auch:

OtrTrackingLowPass, OtrResample, OtrTachoMode

OtrRpmOrder

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Bestimmt den Effektivwert einer Ordnungslinie in Abhängigkeit von der Drehzahl. Der gewünschte Drehzahlbereich wird in Klassen gleicher Breite aufgeteilt. Die Berechnung erfolgt über ein mitlaufendes Bandpassfilter.

Deklaration:

```
OtrRpmOrder ( Schwingung, Drehzahl, Upm_Min, Upm_Max, Upm_Klassenbreite, OrdnungMitte, BreiteProzent, FilterOrdnung, Interpolation ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert
Upm_Min	Unteres Ende des gewünschten Drehzahlbereichs. In U/min skaliert.
Upm_Max	Oberes Ende des gewünschten Drehzahlbereichs. In U/min skaliert.
Upm_Klassenbreite	Breite einer Drehzahl-Klasse (Intervall). In U/min skaliert.
OrdnungMitte	Ordnung(slinie), bei der die Mittenfrequenz des Bandpasses liegt.
BreiteProzent	Gesamtbreite in Prozent. Empfohlen 10%...100%. Wertebereich [0.01 ... 10000.0]. Z.B. bei 30% Breite ist das Verhältnis von oberer zu unterer Grenzfrequenz des Bandpasses 1.30.
FilterOrdnung	Die Filter-Ordnung des Bandpassfilters (2, 4, 6, 8, 10)
Interpolation	Wird das Ergebnis interpoliert?
	0 : Keine Interpolation
	1 : Konstante Interpolation (zentriert um Stützwerte)
	2 : Lineare Interpolation
Ergebnis	Effektivwert einer Ordnungslinie abhängig von der Drehzahl

Beschreibung:

Der Drehzahlbereich beginnt immer bei Upm_Min, die Auflösung beträgt immer Upm_Klassenbreite. Die Angabe von Upm_Max wird lediglich dazu benutzt, die Anzahl der Werte des Ergebnisses zu bestimmen.

Für jede Drehzahlklasse (der Breite Upm_Klassenbreite) sollten ausreichend Messwerte im Schwingungssignal vorhanden sein. Sind gar keine Werte vorhanden, bleibt das Ergebnis in dieser Drehzahlklasse null. Nur wenn eine Interpolation ungleich null gewählt ist, werden die nicht gefüllten Drehzahlklassen gefüllt, indem Werte durch Interpolation benachbarter Werte gebildet werden. Ist eine Interpolation gewählt, werden auch nicht gefüllte Klassen am Rand durch konstante Fortsetzung gefüllt.

Die intern gewählte Mittenfrequenz des Filters liegt bei

Mittenfrequenz = OrdnungMitte * (Aktuelle_Drehzahl / 60)

Die obere Grenzfrequenz liegt oberhalb der Mittenfrequenz und ergibt sich aus der Breite des Filters. Die obere Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa 1 / (0.48 * Abtastzeit) kann keine Bandpassfilterung durchgeführt werden.

Die Funktion arbeitet sinnvoll für:

OrdnungMax << 28 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl.

OrdnungMax = OrdnungMitte * sqrt (1 + BreiteProzent / 100)

<< soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte.

- Die Drehzahl sollte sich nur langsam ändern.
- Die resultierende Mittenfrequenz sollte nicht merklich unter 1 Promille der maximal möglichen sinken. Ist die Drehzahl oder OrdnungMitte wesentlich größer als sinnvoll, steigt die Rechenzeit enorm, ohne den Informationsgehalt zu erhöhen.
- Sinkt die resultierende Mittenfrequenz zu tief, ist das Ergebnis weniger präzise.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Zu beachten ist, dass Bandpässe eine gewisse Zeit beanspruchen, um einzuschwingen. Diese Zeit wächst extrem bei schmalen Filtern. Ein Bandpass der Breite 1% ist in diesem Sinn bereits extrem schmal. Eine Breite von 25% entspricht einem Terzfilter, eine Breite von 100% einem Oktavfilter.

Beispiele:

Aus dem zeitlichen Verlauf der 2.5ten Ordnungslinie soll eine Darstellung des Effektivwertes dieser Ordnungslinie abhängig von der Drehzahl

erstellt werden. Gegeben ist die Schwingung vib mit der Abtastzeit 0.0005 ms und die Drehzahl speed.

```
rpm_Min = 1000.0 ; Minimum des Drehzahlbereichs
rpm_Max = 6000.0 ; Maximum des Drehzahlbereichs
rpm_Delta = 100.0 ; Breite der einzelnen Drehzahlklassen
om = 2.5 ; die 2.5te Ordnung wird gewählt.
width = 30 ; 30% Gesamtbreite
fo = 6 ; Ein Bandpassfilter 6. Ordnung wird benutzt.
Interpolation = 0 ; 0 Standard (keine)
OLine = OtrRpmOrder ( vib, speed, rpm_Min, rpm_Max, rpm_Delta, om, width, fo, Interpolation )
```

Es gilt: OrdnungMax = 2.5 * sqrt (1 + 30 / 100) = 2.9

Und damit: OrdnungMax = 2.9 << 28 / (0.0005 * 6000) = 9.3

Siehe auch:

OtrRpmPresentation

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Aus dem Verlauf eines Signals und der Drehzahl über der Zeit wird ein Verlauf des Signals über der Drehzahl ermittelt. Der gewünschte Drehzahlbereich wird in Klassen gleicher Breite aufgeteilt.

Deklaration:

```
OtrRpmPresentation ( Schwingung, Drehzahl, Upm_Min, Upm_Max, Upm_Klassenbreite, Berechnung, Interpolation, RandOffen ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl.
Upm_Min	Unteres Ende des gewünschten Drehzahlbereichs.
Upm_Max	Oberes Ende des gewünschten Drehzahlbereichs.
Upm_Klassenbreite	Breite einer Drehzahl-Klasse (Intervall).
Berechnung	Wie werden Werte derselben Drehzahlklasse verrechnet?
	0 : Effektivwert (Standard)
	1 : Arithmetischer Mittelwert
	2 : Minimum
	3 : Maximum
	4 : Minimum der Absolutwerte
	5 : Maximum der Absolutwerte
	6 : Mittelwert der Absolutwerte
	7 : Summe
	8 : Standardabweichung, sqrt (1/N * ...)
	9 : Erster Wert
	10 : Letzter Wert
	11 : Anzahl der Messwerte des Schwingungssignals
	12 : Anzahl der Umdrehungen (Drehzahl auch im Kanal Schwingung angeben! Hierbei muss die Drehzahl in U/min skaliert sein!)
Interpolation	Wird das Ergebnis interpoliert?
	0 : Keine Interpolation
	1 : Konstante Interpolation (zentriert um Stützwerte)
	2 : Lineare Interpolation
RandOffen	Werden die Randbereiche als offen angenommen?
	0 : Rand geschlossen. Standard. Ist die Drehzahl außerhalb des gewünschten Bereichs, werden die Schwingungswerte ignoriert.
	1 : Rand offen. Nur für Histogramm-Anwendungen. Alles < Upm_Min kommt in Klasse von Upm_Min, alles > Upm_Max in Klasse von Upm_Max
Ergebnis	Signal, dargestellt abhängig von der Drehzahl

Beschreibung:

Der Drehzahlbereich beginnt immer bei Upm_Min, die Auflösung beträgt immer Upm_Klassenbreite. Die Angabe von Upm_Max wird lediglich dazu benutzt, die Anzahl der Werte des Ergebnisses zu bestimmen. Das Ergebnis hat teilweise den Charakter eines Histogramms, sodass eine Darstellung in Balken bzw. Treppen oft angebracht ist.

Schwingung und Drehzahl können beides Zeitdaten oder beides Winkeldaten sein.

Die Drehzahl muss nicht in U/min skaliert sein. Aber Drehzahl, Upm_Min, Upm_Max und Upm_Klassenbreite müssen alle dieselbe Skalierung (y-Einheit) aufweisen.

Für jede Drehzahlklasse (der Breite Upm_Klassenbreite) sollten ausreichend Messwerte im Schwingungssignal vorhanden sein. Sind gar keine

Werte vorhanden, bleibt das Ergebnis in dieser Drehzahlklasse null. Nur wenn eine Interpolation ungleich null gewählt ist, werden die nicht gefüllten Drehzahlklassen gefüllt, indem Werte durch Interpolation benachbarter Werte gebildet werden. Ist eine Interpolation gewählt, werden auch nicht gefüllte Klassen am Rand durch konstante Fortsetzung gefüllt.

Beispiele:

Aus dem zeitlichen Verlauf einer Ordnungslinie soll eine Darstellung des Effektivwertes dieser Ordnungslinie abhängig von der Drehzahl erstellt werden. Gegeben ist die Schwingung vib und die Drehzahl speed. Zeitlicher Verlauf der 1.5ten Ordnungslinie aus der Schwingung

```
tbp = OtrTrackingBandPass ( vib, speed, 1.5, 30, 4 )
rpm_Min = 1000.0 ; Minimum des Drehzahlbereichs
rpm_Max = 6000.0 ; Maximum des Drehzahlbereichs
rpm_Delta = 100.0  ; Breite der einzelnen Drehzahlklassen
Calc = 0 ; 0 = Effektivwert
Interpolation = 0 ; 0 Standard (keine)
RandOffen = 0 ; 0 Standard (geschlossen)
Order_rpm = OtrRpmPresentation ( tbp, speed, rpm_Min, rpm_Max, rpm_Delta, Calc, Interpolation, RandOffen)
```

Siehe auch:

OtrRpmPresentVector

OtrRpmPresentFast

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Aus dem Verlauf eines Spektrums und der Drehzahl über der Zeit wird ein Verlauf des Spektrums über der Drehzahl ermittelt. Vor allem für schnelle Hochläufe oder Ausläufe. Das Ergebnis ist entlang der Drehzahlachse i. Allg. nicht gleichmäßig geteilt.

Deklaration:

```
OtrRpmPresentFast ( Spektrenfolge, Drehzahl, Drehzahländerung, Drehzahlauflösung, Berechnung ) -> Ergebnis
```

Parameter:

Spektrenfolge	Zeitverlauf eines Spektrums
Drehzahl	Zeitverlauf der (sich schnell ändernden) Drehzahl.
Drehzahländerung	Die Berechnung findet nur statt, wenn sich die Drehzahl in die gewünschte Richtung ändert. Also nur Spektren bei steigender Drehzahl oder nur solche bei fallender Drehzahl beachten.
	1 : Nur steigende Drehzahl
	-1 : Nur fallende Drehzahl
Drehzahlauflösung	Liegen bei aufeinanderfolgenden Spektren die Drehzahlwerte ganz nah beieinander, können all diese Spektren zu einem einzigen verschmelzen. Denn in allen graphischen Darstellungen würde man sie ohnehin nicht auseinanderhalten können. Die Drehzahlauflösung gibt an, wie weit die Drehzahl eines neuen Spektrums abweichen muss, um separat dargestellt zu werden.
Berechnung	Wie werden Spektren mit fast gleicher Drehzahl verrechnet? Falls benachbarte Spektren in der Drehzahl nicht weniger als im Parameter Drehzahlauflösung angegeben auseinanderliegen, werden sie auf eine der folgenden Weisen miteinander verrechnet. Damit ergibt sich dann ein repräsentatives Spektrum.
	0 : Effektivwert (Standard)
	1 : Arithmetischer Mittelwert
	2 : Minimum
	3 : Maximum
	4 : Minimum der Absolutwerte
	5 : Maximum der Absolutwerte
	6 : Mittelwert der Absolutwerte
	7 : Summe
	9 : Erster Wert
	10 : Letzter Wert
Ergebnis	Spektren, dargestellt abhängig von der Drehzahl

Beschreibung:

Die Funktion dient der Analyse eines Hochlaufs einer Maschine oder eines Auslaufs.

Die Funktion erwartet als Eingangsdaten eine Folge von Spektren (Das ist der Zeitverlauf eines Spektrums) und eine dazu passende Folge von Drehzahlwerten (Das ist der Zeitverlauf der Drehzahl). Zu jedem einzelnen Spektrum gehört also ein Drehzahlwert. Jedes Paar von Spektrum und Drehzahl wird in die Ergebnis-Matrix geschrieben. Die Ergebnis-Matrix enthält für jede Drehzahl ein Spektrum.

Die Folge von Eingangsdaten ist ein segmentierter Datensatz. Dabei ist jedes Segment ein Spektrum. Solche Daten entstehen z.B. bei den Funktionen des Spektralpakets, die ein zeitliche Folge von Frequenzspektren liefern. Ein dazu passender Datensatz mit Drehzahlwerten muss existieren. Die Drehzahl ist passend nachabgetastet, z.B. durch gleitende Mittelwertbildung mit der Funktion MvMean().

Die Abtastzeit des Drehzahlsignals wird nicht beachtet. Die Funktion nimmt an, dass immer ein Spektrum und ein Drehzahlwert paarweise zusammengehören.

Das Ergebnis ist ein Datensatz mit Events (Matrix mit zusätzlichen Eigenschaften). Jedes Event stellt ein Spektrum dar und gilt für eine bestimmte Drehzahl. Jedem Event ist eine Drehzahl zugeordnet. Die Events sind immer so sortiert, dass die Drehzahl aufsteigend vorliegt.

Die Drehzahl muss nicht in U/min skaliert sein. Aber die Drehzahl und die Drehzahlauflösung müssen alle dieselbe Skalierung (y-Einheit) aufweisen.

Die Drehzahlauflösung kann auch 0.0 sein. Für schnelle Hochläufe (bzw. schnelle Ausläufe) einer Maschine ist die Drehzahlauflösung nicht von Bedeutung. Dann ist der Wert 0.0 die beste Wahl. Nur in Fällen, bei denen die Drehzahl sich ganz langsam ändert oder sogar konstant bleibt, spielt die Drehzahlauflösung überhaupt eine Rolle. Es gibt Drehzahlen, die so nah beieinander sind, dass man sie auch als gleich annehmen kann, z.B. im Sinn der Messgenauigkeit oder auch nur im Sinn der erforderlichen Genauigkeit einer späteren Auswertung. Bei einem Hochlauf von 1000 U/min auf 6000 U/min kann es z.B. für die Unterscheidung von Spektren nicht von Bedeutung sein, ob die Drehzahl 3345 U/min oder 3350 U/min beträgt. Dann kann die Drehzahlauflösung auf 5 U/min gesetzt werden. Ist die Drehzahlauflösung sehr klein, dann können sehr große

Datenmengen bei langsamen Drehzahländerungen entstehen.

Ändert sich die Drehzahl von einem zum nächsten Spektrum nicht (oder ist kleiner als die angegebene Drehzahlauflösung), so werden diese Spektren miteinander verrechnet. Damit kann für diese Drehzahl eine besseres oder repräsentativeres Spektrum ermittelt werden, z.B. durch Mittelung.

Der Parameter Berechnung ist nur von Bedeutung, wenn die Drehzahländerung von einem Spektrum zum nächsten kleiner als die Drehzahlauflösung ist. Die Berechnung legt fest, wie die einzelnen Spektren gleicher Drehzahl miteinander verrechnet werden. Als Vorschrift für die Berechnung kann z.B. ein Mittelwert oder eine Maximalwertbildung benutzt werden.

Bei langsamen Drehzahländerungen ist die Funktion OtrRpmPresentVector() vorzuziehen. Dort ist nämlich die Matrix entlang der Drehzahlachse gleichmäßig unterteilt, was meist Auswertungen und Darstellungen angenehmer macht.

Hochlauf oder Auslauf: Der Parameter Drehzahländerung gibt an, ob steigende oder fallende Drehzahlen beachtet werden. Gibt man z.B. einen Hochlauf mit steigenden Drehzahlen an, aber die Drehzahl fällt, dann werden die entsprechenden Spektren ignoriert und nicht in das Ergebnis geschrieben. Wenn also während eines Hochlaufs zwischendurch die Drehzahl kurz abnimmt und dann wieder weiter steigt, werden nur diese betroffenen Spektren des Bereichs mit fallender Drehzahl ignoriert. Wenn die Drehzahl dann wieder höher ist als die letzte Drehzahl des steigenden Bereichs, werden die Spektren wieder beachtet. Das Ergebnis wird nur einmalig in einer Richtung gefüllt. Folgt z.B. ein zweiter Hochlauf, wird er ignoriert. In solchen Fällen ist die Funktion OtrRpmPresentVector() vorzuziehen.

Beispiele:

Mit Hilfe einer Funktion des Spektralpaketes wird der Zeitverlauf eines Spektrums aus dem Schwingungskanal Vibration bestimmt. Der Schwingungskanal hat eine Abtastzeit von 1ms. Es sollen Spektren der Länge 1000 Punkte mit 75% Überlappung bestimmt werden. Gleichzeitig wurde die Drehzahl speed mit derselben Abtastzeit gemessen. Es soll ein schneller Hochlauf eines Motors gemessen werden.

```
Spectra = AmpSpectrumRMS ( Vibration, 1000, 0, 75, 1, 0, 0 ) ; aus dem Spektralpaket
speed1 = MvMean ( speed,  1.0, 0.25 )
UpDown = 1 ; 1 Hochlauf, -1 Auslauf
NResolution =0.0  ; Drehzahlauflösung
Calc = 1 ; 1 =Mittelwert
RpmSpectrum = OtrRpmPresentFast (Spectra, speed1, UpDown, NResolution, Calc)
```

Immer nach 250 Werten des Schwingungskanals wird ein Spektrum bestimmt, also nach 250ms. Alle 250ms wird also die Drehzahl benötigt. Diese soll aber über 1000ms gemittelt sein.

Siehe auch:

OtrRpmPresentVector

OtrRpmPresentVector

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Aus dem Verlauf eines Spektrums und der Drehzahl über der Zeit wird ein Verlauf des Spektrums über der Drehzahl ermittelt. Der gewünschte Drehzahlbereich wird in Klassen gleicher Breite aufgeteilt.

Deklaration:

```
OtrRpmPresentVector ( Spektrenfolge, Drehzahl, Upm_Min, Upm_Max, Upm_Klassenbreite, Berechnung, Interpolation, RandOffen, Drehzahländerung ) -> Ergebnis
```

Parameter:

Spektrenfolge	Zeitverlauf eines Spektrums
Drehzahl	Zeitverlauf der Drehzahl
Upm_Min	Unteres Ende des gewünschten Drehzahlbereichs.
Upm_Max	Oberes Ende des gewünschten Drehzahlbereichs.
Upm_Klassenbreite	Breite einer Drehzahl-Klasse (Intervall).
Berechnung	Wie werden Spektren derselben Drehzahlklasse verrechnet?
	0 : Effektivwert (Standard)
	1 : Arithmetischer Mittelwert
	2 : Minimum
	3 : Maximum
	4 : Minimum der Absolutwerte
	5 : Maximum der Absolutwerte
	6 : Mittelwert der Absolutwerte
	7 : Summe
	9 : Erster Wert
	10 : Letzter Wert
Interpolation	Wird das Ergebnis interpoliert?
	0 : Keine Interpolation
	1 : Konstante Interpolation (zentriert um Stützwerte)
	2 : Lineare Interpolation
RandOffen	Werden die Randbereiche als offen angenommen?
	0 : Rand geschlossen. Standard. Ist die Drehzahl außerhalb des gewünschten Bereichs, werden die Schwingungswerte ignoriert.
	1 : Rand offen. Nur für Histogramm-Anwendungen. Alles < Upm_Min kommt in Klasse von Upm_Min, alles > Upm_Max in Klasse von Upm_Max
Drehzahländerung	Die Berechnung findet nur statt, wenn sich die Drehzahl in die gewünschte Richtung ändert. Also z.B. nur Daten mit steigender Drehzahl oder nur solche mit fallender Drehzahl beachten.
	0 : Drehzahlverlauf egal
	1 : Nur steigende Drehzahl
	-1 : Nur fallende Drehzahl
Ergebnis	Spektren, dargestellt abhängig von der Drehzahl

Beschreibung:

Die Funktion erwartet als Eingangsdaten eine Folge von Spektren (Das ist der Zeitverlauf eines Spektrums) und eine dazu passende Folge von Drehzahlwerten (Das ist der Zeitverlauf der Drehzahl). Zu jedem einzelnen Spektrum gehört also ein Drehzahlwert. Jedes Paar von Spektrum und Drehzahl wird in die Ergebnis-Matrix geschrieben. Die Ergebnis-Matrix enthält für jeden Drehzahlbereich ein Spektrum.

Die Folge von Eingangsdaten ist ein segmentierter Datensatz. Dabei ist jedes Segment ein Spektrum. Solche Daten entstehen z.B. bei den Funktionen des Spektralpakets, die ein zeitliche Folge von Frequenzspektren liefern. Ein dazu passender Datensatz mit Drehzahlwerten muss existieren. Die Drehzahl ist passend nachabgetastet, z.B. durch gleitende Mittelwertbildung mit der Funktion MvMean().

Die Abtastzeit des Drehzahlsignals wird nicht beachtet. Die Funktion nimmt an, dass immer ein Spektrum und ein Drehzahlwert paarweise zusammengehören.

Der Drehzahlbereich beginnt immer bei Upm_Min, die Auflösung beträgt immer Upm_Klassenbreite. Die Angabe von Upm_Max wird lediglich dazu benutzt, die Anzahl der Werte des Ergebnisses zu bestimmen. Das Ergebnis hat teilweise den Charakter einer Matrix, sodass eine Darstellung in Balken

bzw. Treppen oft angebracht ist.

Das Ergebnis ist ein segmentierter Datensatz (Matrix). Jedes Segment stellt ein Spektrum dar und gilt für eine bestimmte Drehzahl.

Die Drehzahl muss nicht in U/min skaliert sein. Aber Drehzahl, Upm_Min, Upm_Max und Upm_Klassenbreite müssen alle dieselbe Skalierung (y-Einheit) aufweisen.

Für jede Drehzahlklasse (der Breite Upm_Klassenbreite) sollte wenigstens ein Spektrum vorhanden sein. Ist kein Spektrum vorhanden, bleibt das Ergebnis in dieser Drehzahlklasse null. Nur wenn eine Interpolation ungleich null gewählt ist, werden die nicht gefüllten Drehzahlklassen gefüllt, indem Werte durch Interpolation benachbarter Werte gebildet werden. Ist eine Interpolation gewählt, werden auch nicht gefüllte Klassen am Rand durch konstante Fortsetzung gefüllt.

Für extrem schnelle Hochläufe und schelle Ausläufe ist die Funktion nicht geeignet, weil dann viele Drehzahlbereiche der Matrix nicht gefüllt werden oder aber eine riesig Klassenbreite für die Drehzahl benutzt werden muss. Dann ist die Funktion OtrRpmPresentFast() vorzuziehen.

Beispiele:

Mit Hilfe einer Funktion des Spektralpaketes wird der Zeitverlauf eines Spektrums aus dem Schwingungskanal "Vibration" bestimmt. Der Schwingungskanal hat eine Abtastzeit von 1ms. Es sollen Spektren der Länge 1000 Punkte mit 50% Überlappung bestimmt werden. Gleichzeitig wurde die Drehzahl speed mit einer Abtastzeit von 10ms gemessen.

```
Spectra = AmpSpectrumRMS ( Vibration, 1000, 0, 50, 1, 0 ) ; aus dem Spektralpaket
speed1 = MvMean ( speed,  1.0, 0.5 )
rpm_Min = 1000.0 ; Minimum des Drehzahlbereichs
rpm_Max = 6000.0 ; Maximum des Drehzahlbereichs
rpm_Delta = 100.0  ; Breite der einzelnen Drehzahlklassen
Calc = 0 ; 0 = Effektivwert
Interpolation = 0 ; 0 Standard (keine)
Bounds = 0 ; 0 Standard (geschlossen)
UpDown = 0 ; Standard (Drehzahländerung beliebig)
RpmSpectrum = OtrRpmPresentVector (Spectra, speed1, rpm_Min, rpm_Max, rpm_Delta, Calc, Interpolation, Bounds, UpDown)
```

Immer nach 500 Werten des Schwingungskanals wird ein Spektrum bestimmt, also nach 500ms. Alle 500ms wird also die Drehzahl benötigt. Diese soll aber über1000ms gemittelt sein.

Siehe auch:

OtrRpmPresentation, OtrRpmPresentFast

OtrRpmSpectrum

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Das FFT-Spektrum (Effektivwerte!) wird aus den Zeitverläufen von Schwingung und Drehzahl in Abhängigkeit von der Drehzahl bestimmt. Der gewünschte Drehzahlbereich wird in Klassen gleicher Breite aufgeteilt.

Deklaration:

```
OtrRpmSpectrum ( Schwingung, Drehzahl, Upm_Min, Upm_Max, Upm_Klassenbreite, Fensterbreite, Fenstertyp, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert
Upm_Min	Unteres Ende des gewünschten Drehzahlbereichs. In U/min skaliert.
Upm_Max	Oberes Ende des gewünschten Drehzahlbereichs. In U/min skaliert.
Upm_Klassenbreite	Breite einer Drehzahl-Klasse (Intervall). In U/min skaliert.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Mittelungsart	Wie werden alle Spektren zur selben Drehzahlklasse gemittelt?
	0 : arithmet. Mittel
	1 : arithmet. Mittel, 50% Überlappung
	2 : arithmet. Mittel, 75% Überlappung
	3 : Maximum
	4 : Maximum, 50% Überlappung
	5 : Maximum, 75% Überlappung
	6 : Minimum
	7 : Minimum, 50% Überlappung
	8 : Minimum, 75% Überlappung
	9 : das erste
	10 : arithmet. Mittel, streng
	11 : arithmet. Mittel, streng, 50% Überlappung
	12 : arithmet. Mittel, streng, 75% Überlappung
	13 : Maximum, streng
	14 : Maximum, streng, 50% Überlappung
	15 : Maximum, streng, 75% Überlappung
	16 : Minimum, streng
	17 : Minimum, streng, 50% Überlappung
	18 : Minimum, streng, 75% Überlappung
	19 : das erste, streng
	20 : RMS

	21 : RMS, 50% Überlappung
	22 : RMS, 75% Überlappung
	23 : RMS, 90% Überlappung
	24 : RMS, 95% Überlappung
	25 : RMS, 98% Überlappung
	26 : RMS, 99% Überlappung
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	FFT-Spektrum abhängig von der Drehzahl

Beschreibung:

Die Fensterbreite (Punkteanzahl) für die FFT muss keine 2er-Potenz sein. Eine Fensterbreite von z.B. 500 oder 1000 führt zu "schönen" Frequenzlinienabständen.

- Die Spektrallinien sind als Effektivwerte angegeben.
- Die Mittelung arbeitet auf dem Betragsspektrum.

Strenge Ausführung der Mittelung:

- Spektren werden nur berücksichtigt, wenn während der ganzen Zeit für ein Spektrum die Drehzahl innerhalb einer Drehzahlklasse ist. Die Drehzahl sollte sich also nur langsam ändern. Wenn sich die Drehzahl schnell ändert und die Spektren sollen berücksichtigt werden, ist ein entsprechend großes Upm_Klassenbreite zu wählen!

Nicht strenge Ausführung der Mittelung:

- Die mittlere Drehzahl während eines Spektrums bestimmt die Drehzahlklasse. Das kann sehr ungenau werden, wenn sich die Drehzahl sehr schnell ändert.

Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.

- Falls in einer Drehzahlklasse kein Spektrum bestimmt wird, wird dieses Spektrum mit Nullen gefüllt.
- Das Ergebnis ist ein segmentierter Datensatz. Jedes Segment ist ein Spektrum.

Beispiele:

Aus dem zeitlichen Verlauf einer Schwingung vib und der Drehzahl speed soll das Spektrum abhängig von der Drehzahl bestimmt werden.

```
rpm_Min = 1000.0 ; Minimum des Drehzahlbereichs
rpm_Max = 6000.0 ; Maximum des Drehzahlbereichs
rpm_Delta = 100.0 ; Breite der einzelnen Drehzahlklassen
WindowSize = 1000 ; Breite des Fensters für die FFT, als Anzahl von Messwerten
WindowType = 0; 0 Rechteck
AvgType = 0 ; 0 (arithmet. Mittel)
RpmSpectrum = OtrRpmSpectrum ( vib, speed, rpm_Min, rpm_Max, rpm_Delta, WindowSize, WindowType, AvgType, 0 )
```

Bei einer Abtastzeit von 0.5ms und einer Fensterbreite von 1000 Punkten wird ein Spektrum mit Frequenzlinienabstand 2 Hz berechnet.

Siehe auch:

OtrRpmPresentVector, OtrRpmPresentFast, OtrRpmSpectrumFast

OtrRpmSpectrumFast

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Das FFT-Spektrum (Effektivwerte!) wird aus den Zeitverläufen von Schwingung und Drehzahl in Abhängigkeit von der Drehzahl bestimmt. Vor allem für schnelle Hochläufe oder Ausläufe. Das Ergebnis ist entlang der Drehzahlachse i. Allg. nicht gleichmäßig geteilt.

Deklaration:

```
OtrRpmSpectrumFast ( Schwingung, Drehzahl, Fensterbreite, Fenstertyp, Überlappung, Drehzahländerung,
Drehzahlauflösung, Berechnung [, Basis2] ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der (sich schnell ändernden) Drehzahl.
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Überlappung in %, z.B. 0.0, 50 oder 75. Um diesen Anteil überlappen sich die einzelnen Zeitfenster, aus denen Spektren gebildet werden.
	0 : Keine Überlappung
	50 : 50% Überlappung bzw. 1/2
	75 : 75% Überlappung bzw. 3/4
Drehzahländerung	Die Berechnung findet nur statt, wenn sich die Drehzahl in die gewünschte Richtung ändert. Also nur Spektren bei steigender Drehzahl oder nur solche bei fallender Drehzahl beachten.
	1 : Nur steigende Drehzahl
	-1 : Nur fallende Drehzahl
Drehzahlauflösung	Liegen bei aufeinanderfolgenden Spektren die Drehzahlwerte ganz nah beieinander, können all diese Spektren zu einem einzigen verschmelzen. Denn in allen graphischen Darstellungen würde man sie ohnehin nicht auseinanderhalten können. Die Drehzahlauflösung gibt an, wie weit die Drehzahl eines neuen Spektrums abweichen muss, um separat dargestellt zu werden.
Berechnung	Wie werden Spektren mit fast gleicher Drehzahl verrechnet? Falls benachbarte Spektren in der Drehzahl nicht weniger als im Parameter Drehzahlauflösung angegeben auseinanderliegen, werden sie auf eine der folgenden Weisen miteinander verrechnet. Damit ergibt sich dann ein repräsentatives Spektrum.
	0 : Effektivwert (Standard)
	1 : Arithmetischer Mittelwert
	2 : Minimum
	3 : Maximum
	4 : Minimum der Absolutwerte
	5 : Maximum der Absolutwerte
	6 : Mittelwert der Absolutwerte
	7 : Summe
	9 : Erster Wert
	10 : Letzter Wert
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Spektren, dargestellt abhängig von der Drehzahl

Beschreibung:

Die Funktion dient der Analyse eines (schnellen) Hochlaufs einer Maschine oder eines Auslaufs.

Die Funktion erwartet als Eingangsdaten die Zeitverläufe einr Schwingungssignals und eines Drehzahlsignals. Aus dem Schwingungssignal werden Spektren bestimmt. Zu jedem Spektrum wird die mittlere Drehzahl bestimmt. Aus den Paaren von Spektrum und Drehzahl wird eine Matrix erzeugt. Diese Matrix stellt das Spektrum drehzahlabhängig dar.

Das Ergebnis ist ein Datensatz mit Events (Matrix mit zusätzlichen Eigenschaften). Jedes Event stellt ein Spektrum dar und gilt für eine bestimmte Drehzahl. Jedem Event ist eine Drehzahl zugeordnet. Die Events sind immer so sortiert, dass die Drehzahl aufsteigend vorliegt.

Die Drehzahl muss nicht in U/min skaliert sein. Aber die Drehzahl und die Drehzahlauflösung müssen alle dieselbe Skalierung (y-Einheit) aufweisen.

Die Drehzahlauflösung kann auch 0.0 sein. Für schnelle Hochläufe (bzw. schnelle Ausläufe) einer Maschine ist die Drehzahlauflösung nicht von Bedeutung. Dann ist der Wert 0.0 die beste Wahl. Nur in Fällen, bei denen die Drehzahl sich ganz langsam ändert oder sogar konstant bleibt, spielt die Drehzahlauflösung überhaupt eine Rolle. Es gibt Drehzahlen, die so nah beieinander sind, dass man sie auch als gleich annehmen kann, z.B. im Sinn der Messgenauigkeit oder auch nur im Sinn der erforderlichen Genauigkeit einer späteren Auswertung. Bei einem Hochlauf von 1000 U/min auf 6000 U/min kann es z.B. für die Unterscheidung von Spektren nicht von Bedeutung sein, ob die Drehzahl 3345 U/min oder 3350 U/min beträgt. Dann kann die Drehzahlauflösung auf 5 U/min gesetzt werden. Ist die Drehzahlauflösung sehr klein, dann können sehr große Datenmengen bei langsamen Drehzahländerungen entstehen.

Ändert sich die Drehzahl von einem zum nächsten Spektrum nicht (oder ist kleiner als die angegebene Drehzahlauflösung), so werden diese Spektren miteinander verrechnet. Damit kann für diese Drehzahl eine besseres oder repräsentativeres Spektrum ermittelt werden, z.B. durch Mittelung.

Der Parameter Berechnung ist nur von Bedeutung, wenn die Drehzahländerung von einem Spektrum zum nächsten kleiner als die Drehzahlauflösung ist. Die Berechnung legt fest, wie die einzelnen Spektren gleicher Drehzahl miteinander verrechnet werden. Als Vorschrift für die Berechnung kann z.B. ein Mittelwert oder eine Maximalwertbildung benutzt werden.

Bei langsamen Drehzahländerungen ist die Funktion OtrRpmSpectrum() vorzuziehen. Dort ist nämlich die Matrix entlang der Drehzahlachse gleichmäßig unterteilt, was meist Auswertungen und Darstellungen angenehmer macht.

Hochlauf oder Auslauf:

- Der Parameter Drehzahländerung gibt an, ob steigende oder fallende Drehzahlen beachtet werden. Gibt man z.B. einen Hochlauf mit steigenden Drehzahlen an, aber die Drehzahl fällt, dann werden die entsprechenden Spektren ignoriert und nicht in das Ergebnis geschrieben. Wenn also während eines Hochlaufs zwischendurch die Drehzahl kurz abnimmt und dann wieder weiter steigt, werden nur diese betroffenen Spektren des Bereichs mit fallender Drehzahl ignoriert. Wenn die Drehzahl dann wieder höher ist als die letzte Drehzahl des steigenden Bereichs, werden die Spektren wieder beachtet. Das Ergebnis wird nur einmalig in einer Richtung gefüllt. Folgt z.B. ein zweiter Hochlauf, wird er ignoriert. In solchen Fällen ist die Funktion OtrRpmSpectrum() vorzuziehen.

Die Kanäle Schwingung und Drehzahl können auch unterschiedliche Abtastraten haben.

Beispiele:

Während des schnellen Hochlaufs eines Motors soll das FFT-Spektrum abhängig von der Drehzahl bestimmt werden. Die Messsignale Vibration (Schwingung) und speed (Drehzahl) liegen vor. FFTs der Breite 1000 sollen berechnet werden.

```
WindowSize = 1000 ; Breite des Fensters für die FFT, als Anzahl von Messwerten
WindowType = 0; 0 Rechteck
Overlap = 75 ; Überlappung in %
UpDown = 1 ; 1 Hochlauf, -1 Auslauf
NResolution =0.0  ; Drehzahlauflösung
Calc = 1 ; 1 =Mittelwert
FFT_Spectrum = OtrRpmSpectrumFast ( Vibration, speed, WindowSize, WindowType, Overlap, UpDown, NResolution, Calc, 0)
```

Zur Berechnung wird eine deutliche Überlappung der Zeitfenster der FFT gewählt. Damit kann man auch bei schnellen Hochläufen eine größere Anzahl von Spektren erhalten.

Siehe auch:

OtrRpmPresentVector, OtrRpmPresentFast

OtrRpmThirds

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Das Terz-Spektrum wird aus den Zeitverläufen von Schwingung und Drehzahl in Abhängigkeit von der Drehzahl bestimmt. Der gewünschte Drehzahlbereich wird in Klassen gleicher Breite aufgeteilt.

Deklaration:

```
OtrRpmThirds ( Schwingung, Drehzahl, Upm_Min, Upm_Max, Upm_Klassenbreite, f1, f2, Frequenzbewertung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert
Upm_Min	Unteres Ende des gewünschten Drehzahlbereichs. In U/min skaliert.
Upm_Max	Oberes Ende des gewünschten Drehzahlbereichs. In U/min skaliert.
Upm_Klassenbreite	Breite einer Drehzahl-Klasse (Intervall). In U/min skaliert.
f1	Mittenfrequenz der untersten Terz in Hz
f2	Mittenfrequenz der obersten Terz in Hz
Frequenzbewertung	Mit welcher Frequenzbewertung wird das Resultat gewichtet?
	0 : linear
	1 : A-Bewertung
	2 : B-Bewertung
	3 : C-Bewertung
	4 : D-Bewertung
Ergebnis	Terzspektrum abhängig von der Drehzahl

Beschreibung:

Die beiden Frequenzgrenzen f1 und f2 sollten als Terzmittenfrequenz angegeben werden, z.B. f1 = 8 Hz und f2 = 12500 Hz. f1 < f2. Die oberste Terz muss mit ihrem Frequenzband vollständig innerhalb der halben Abtastfrequenz liegen.

Die einzelnen Werte der Terzen sind als Effektivwerte angegeben.

Während zu Beginn der Messung einmalig die einzelnen Terz-Filter (Bandpässe) einschwingen, werden die Werte des Eingangssignals ignoriert. Das Einschwingen wird bei der 1kHz Terz zu 35ms angenommen. Diese Dauer ist umgekehrt proportional zur Terzfrequenz. Bei sehr niedrigen Terzen wird diese Dauer beachtlich. Eine entsprechend lange dauernde Messung ist dann vorzusehen.

Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.

Falls in einer Drehzahlklasse keine Werte vorliegen, wird dieses Terz-Spektrum mit Nullen gefüllt. Für einen sinnvollen Betrieb sollten für jede Drehzahlklasse ausreichend viele Messwerte vorliegen. Dabei sind besonders die untersten Terzen zu berücksichtigen.

Das Ergebnis ist ein segmentierter Datensatz. Jedes Segment enthält ein Terzspektrum. Die x-Koordinate des Resultates zählt die Terzen (genauso wie die Famos-Funktion OctA). Für eine sinnvolle Darstellung im Kurvenfenster ist dort die Terzbeschriftung zu wählen.

Die Terzfilter und Bewertungen entsprechen IEC 651 (Schallpegelmesser), DIN 45652 (Terzfilter für elektroakustische Messungen) und EN61260-1:2014 bzw. IEC61260-1:2014 (Bandfilter für Oktaven und Bruchteile von Oktaven, Filterklasse 1).

Beispiele:

Aus dem zeitlichen Verlauf einer Schwingung vib und der Drehzahl speed soll das Terz-Spektrum abhängig von der Drehzahl bestimmt werden. Die Abtastzeit des Schwingungssignals ist 0.025 ms.

```
rpm_Min = 1000.0 ; Minimum des Drehzahlbereichs
rpm_Max = 6000.0 ; Maximum des Drehzahlbereichs
rpm_Delta = 100.0 ; Breite der einzelnen Drehzahlklassen
fEval = 1 ; 0 (linear) 1 (A-Bewertung)
f1 = 10
f2 = 12500
Thirds = OtrRpmThirds ( vib, speed, rpm_Min, rpm_Max, rpm_Delta, f1, f2, fEval )
```

Siehe auch:

OtrRpmSpectrum, SpecThirds, OctI

OtrTachoMode

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Von welcher Art ist das Tachosignal, also z.B. schon eine Drehzahl oder sind es die Pulses eines Encoders? Für die anderen Funktionen des Kits mit Tachosignal als Übergabe-Parameter wird damit die Deutung des Tachosignals festgelegt.

Deklaration:

```
OtrTachoMode ( Signaltyp, Encodertyp, Encoderpulse, MinDrehzahl )
```

Parameter:

Signaltyp	Von welchem Typ ist das Tachosignal?
	0 : Drehzahl in U/min
	1 : Anzahl Pulse pro Abtastschritt
	2 : Cronos Impulszeitpunkt
	3 : Abgetastetes Rechtecksignal
	4 : Sinusförmiges Signal
Encodertyp	Typ des Encoders. Hat der Encoder alle Zähne?
	0 : Standard
	1 : 1 Zahn fehlt
	2 : 2 Zähne fehlen
Encoderpulse	Anzahl der Pulse des Encoders. So viele Pulse erzeugt der Encoder pro Umdrehung. >= 1. Die Anzahl muss i. Allg. nicht ganzzahlig sein.
MinDrehzahl	Minimale Drehzahl, in U/min skaliert. >= 0. Bei Encodern mit fehlenden Zähnen die Drehzahl, bei der noch eine Erkennung der Lücke möglich ist.

Beschreibung:

Von welcher Art ist das Tachosignal, also z.B. schon eine Drehzahl oder sind es die Pulses eines Encoders? Für die anderen Funktionen des Kits mit Tachosignal als Übergabe-Parameter wird damit die Deutung des Tachosignals festgelegt. Der Aufruf dieser Funktion hat also selbst keine unmittelbare Aktion zur Folge. Im Gedächtnis des Kits wird lediglich festgelegt, wie bei nachfolgenden Funktionsaufrufen das Tachosignal zu deuten ist. Nur die Funktionen, deren Parameter ausdrücklich mit Tachosignal bezeichnet ist, sind davon betroffen. Andere Funktionen, die ein Drehzahlsignal als Parameter haben, erwarten auch stets nur eine Drehzahl, in U/min skaliert.

Signaltyp 0: Drehzahl in U/min

- Das Tachosignal ist ein Drehzahlsignal, das fest in U/min skaliert ist.

Signaltyp 1: Anzahl Pulse pro Abtastschritt

- Das Tachosignal enthält eine Folge von (ganzen) Zahlen. Jede Zahl ist die bereits gezählte Anzahl der Pulse innerhalb des aktuellen Abtastschrittes. Eine Zählung der Pulse ist bereits durchgeführt worden. Es wird angenommen, dass die Drehzahl direkt proportional zur gezählten Anzahl von Pulsen ist. Die Funktion glättet das Drehzahlsignal so, dass eine verbesserte Schätzung der Drehzahl entsteht. Enthält das Signal die Wertefolge { ...., 3, 4, 4, 4, 3, 4, 4, 4, 3... }, so kann für diesen Teil eine Drehzahl von etwa 3.75 geschätzt werden. Die Funktion fasst jeden Wert des Signals als eine Anzahl von Pulsen auf. Enthält das Signal die Wertefolge { 0, 0, 1, 0, 1, 2, 1 }, so werden 5 Pulse detektiert. Hinweis: Aus dem vorliegenden Signal kann die Drehzahl oft nicht sehr genau ermittelt werden.

Signaltyp 2: Cronos Impulszeitpunkt

- Dieser Signaltyp liegt nur vor, falls mit dem Messgerät imc-Cronos gearbeitet wird. Dabei muss der Inkrementalgebereingang des Gerätes auf "Messung Impulszeitpunkt" gestellt sein. Eine äußerst präzise Erfassung der Pulse wird durchgeführt. Aus diesem Signal lässt sich die Drehzahl sehr präzise bestimmen.

Signaltyp 3: Abgetastetes Rechtecksignal

- Bei jedem Übergang des Signals von "Kleiner gleich 0.0" auf "Größer 0.0" wird angenommen, dass der Encoder sich um ein Inkrement weiter gedreht hat. Enthält das Signal die Wertefolge { 0, 0, 1, 1, 1, 0, 0, 1, 1, 0 }, so werden 2 Pulse detektiert. Wenn eine analoge Spannung abgetastet wird und kein Komparator in Hardware vorliegt, muss die analoge Spannung so bearbeitet werden, dass ein deutlicher Nulldurchgang erfolgt. Ist Rauschen auf dem Signal, muss gegebenenfalls vorher noch geglättet werden und ein Schmitt-Trigger angewendet werden.

Signaltyp 4: Sinusförmiges Signal

- Auch sinusförmige oder andere Signale mit eindeutigem Nulldurchgang bei positiver Flanke können verarbeitet werden. Ist Rauschen auf dem Signal, muss gegebenenfalls vorher noch geglättet werden und ein Schmitt-Trigger angewendet werden. Es wird angenommen, dass beim Nulldurchgang in positiver Richtung (steigende Flanke) ein Puls des Gebers auftritt. Enthält das Signal die Wertefolge { -3.0, -1.0, +1.5, +2.8, +1.3, +0.1, -0.6 }, so wurde beim Übergang von -1.0 auf +1.5 ein Puls detektiert.

Besonderheiten bei Gebern mit fehlenden Pulsen:

- Encodertyp = 0 ist Standard. Nur bei Signaltyp 2 (Cronos Impulszeitpunkt) kann ein anderer Encodertyp gewählt werden. Die Anzahl der Encoderpulse wird stets inklusive dem fehlenden Zahn angegeben und muss ganzzahlig sein. Z.B. für einen Geber, der alle 10 Grad einen Puls liefert und damit eigentlich 36 Zähne haben müsste, wird auch 36 als Encoderpulse angegeben. Der Geber erzeugt aber nur 35 Pulse, weil ihm einer fehlt. Auch üblich ist ein Geber, der alle 6 Grad einen Puls erzeugt. Hier werden 60 Zähne angegeben, obwohl ihm 2 fehlen und er damit nur 58 wirklich hat. Es wird angenommen, dass der erste Zahn nach der Lücke der Nullimpuls ist (hat bei dieser Funktion aber keine Auswirkung). Geber mit fehlenden Impulsen können nur bei Messung mit Cronos benutzt werden. Die Erkennung der fehlenden Zähne ist nur möglich, wenn die Drehzahl einigermaßen konstant ist um die Lücke herum. Besonders bei extrem niedrigen Drehzahlen kann das nicht garantiert werden. Da dann die Erkennung der Lücke nicht eindeutig ist, muss die minimale Drehzahl auf einen Wert ungleich null gesetzt werden. Bei höheren Drehzahlen ist meist aufgrund der Trägheit der Mechanik die Zahnlücke eindeutig zu erkennen. Die Funktion versucht, sich nach einem Fehler in der Pulsfolge (oder einer vermeintlich falsch interpretierten Pulsfolge) wieder erneut zu synchronisieren. Trotzdem können zwischendurch falsche Drehzahlwerte aufgetreten sein.

Minimale Drehzahl:

- Dieser Wert kann meist 0.0 sein. Die minimale Drehzahl ist von Bedeutung, um zwischen einem Stillstand und einer schleichenden Bewegung zu unterscheiden. In Situationen, in denen der Antrieb steht, könnte sonst eine ganz winzige Drehzahl ungleich null geschätzt werden. Durch Setzen der minimalen Drehzahl auf einen Wert über null wird bei Unterschreiten dieses Grenzwertes eine klare null im Drehzahlverlauf erzeugt. I. Allg. ist es bei extrem großen Pulsabständen nicht mehr möglich, zwischen einem Stillstand und einer ganz langsamen kontinuierlichen Bewegung zu unterscheiden. Die minimale Drehzahl gibt an, welches die kleinste interessierende Drehzahl ist. Ist der Abstand der Pulse größer als für diese kleinste Drehzahl, wird die Drehzahl zu 0.0 angenommen. Bei Encodern mit fehlenden Zähnen muss die minimale Drehzahl größer 0.0 sein. Hinweis: Dieses Null-Setzen der Drehzahl verändert (verfälscht) das Integral über die Drehzahl. Deshalb sollte solch ein Drehzahlsignal dann später nicht integriert werden, um den Winkel zu bestimmen.

Hinweis:

- Vor Aufruf von anderen Funktionen mit Tachosignal als Parameter, z.B. OtrTachoToSpeed(), sollte wenigstens einmal OtrTachoMode() aufgerufen werden, um die Art des Tachosignals eindeutig festzulegen. Wird OtrTachoMode() nicht aufgerufen, wird angenommen, dass das Tacho bereits eine Drehzahl in U/min liefert. Die jeweils letzte Einstellung von OtrTachoMode() wird solange benutzt, bis OtrTachoMode() erneut aufgerufen wird. Dabei ist zu beachten, dass das Gedächtnis für die Dauer der Applikation anhält. Um also eine Abfolge von Befehlen (Sequenz) robust zu gestalten gegenüber der Vorgeschichte (also was auch immer vorher für Befehle bzw. Sequenzen ausgeführt wurden), sollte stets ein Aufruf von OtrTachoMode() zu Beginn erfolgen.

Beispiele:

Die Drehzahl an einem Zahnrad soll bestimmt werden. Dazu wurde mit einem induktiven Aufnehmer ein rechteckiges Spannungssignal erzeugt. Diese Spannung wurde mit konstanter Abtastrate von 1kHz aufgezeichnet und steht im Kanal mit dem Namen Signal zur Verfügung. Die Spannung beträgt etwa 0V..1V in der Zahnlücke, etwa 18V..22V an der Spitze des Zahns. Das Zahnrad hat 8 Zähne.

```
OtrTachoMode ( 3, 0, 8, 10.0)
EncoderPulses = stri ( Signal, 15, 5 )
Speed =OtrTachoToSpeed ( EncoderPulses )
```

Da die Spannung nicht geeignet vorliegt, wird sie so verändert, dass ein sauberer Nulldurchgang entsteht. Ist Rauschen auf dem Signal, muss gegebenenfalls noch geglättet werden und/oder ein Schmitt-Trigger angewendet werden.

Einmalig wird mit OtrTachoMode () die Art des Tachosignals eingestellt. Anschließend braucht dann nur noch OtrTachoToSpeed () für verschiedene Messungen und Kanäle aufgerufen zu werden.

Der Antrieb läuft mit Drehzahlen im Bereich 500...3000U/min. Das ergibt eine Pulsfrequenz von bis zu 400Hz(8*50Hz) Drehzahlen von weniger als 10U/min sollen nicht mehr angezeigt werden. Stattdessen soll eine saubere Null-Linie für die Drehzahl angezeigt werden.

Siehe auch:

OtrOrderSpectrum, OtrResampleAAF, OtrResample, OtrTachoToSpeed, OtrTachoToDist, OtrEncoderRevs01

OtrTachoToDist

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Aus dem Tachosignal, das z.B. in Form von Encoder-Pulsen vorliegt, wird ein Umdrehungsdatensatz berechnet, der die Umdrehungen zählt. Vor Aufruf dieser Funktion muss mit der Funktion OtrTachoMode() die Art des Tachosignals festgelegt worden sein.

Deklaration:

```
OtrTachoToDist ( Tachosignal ) -> Umdrehungsverlauf
```

Parameter:

Tachosignal	Tachosignal, entsprechend vorherigem Aufruf von OtrTachoMode() gedeutet.
Umdrehungsverlauf	

Beschreibung:

Die Funktion integriert die Pulse des Encoders. Das Ergebnis enthält also die stets akkumulierte Summe aller bisher gezählten Pulse. Das Ergebnis ist so skaliert, dass es die Umdrehungen (Überrollungen) des Antriebs zählt. Hat das Ergebnis an einer Stelle den Wert 1.0, so bedeutet dies, dass seit dem Beginn eine volle Umdrehung zurückgelegt wurde. Ist z.B. der Wert dann später 2.0, so sind bis dahin 2 volle Umdrehungen zurückgelegt. Die Funktion ermittelt aber nicht nur volle Umdrehungen, sondern auch Bruchteile.

Die Integration beginnt i. Allg. mit dem ersten Puls. Liegt ein Geber mit fehlenden Pulsen vor, beginnt die Integration mit dem ersten Puls hinter der Lücke. Damit entsteht ein phasenrichtiges Winkelsignal.

Ein gleichwertiges Ergebnis kann nicht (!) durch Integration der Drehzahl aus OtrTachoToSpeed() erzielt werden.

Um eine Anzeige in Grad zu erhalten, kann das Resultat mit 360 Grad multipliziert werden.

Vor Aufruf dieser Funktion sollte wenigstens einmal OtrTachoMode() aufgerufen werden.

Beispiele:

Der Winkel soll an einem Polrad bestimmt werden, das 36 Zähne mit einer Lücke hat. Die Drehzahl liegt zwischen 800 und 7000 U/min. Die Erfassung erfolgt mit Cronos PL im Modus Impulszeitpunkt, womit eine präzise zeitliche Bestimmung jeder einzelnen Flanke des Gebersignals erfolgt.

```
OtrTachoMode ( 2, 1, 36, 500)
Revolutions =OtrTachoToDist ( Tacho )
```

Das Ergebnis Revolutions enthält die gezählten Umdrehungen. Zuvor wird mit OtrTachoMode () die Art des Tachosignals eingestellt.

Siehe auch:

OtrOrderSpectrum, OtrResampleAAF, OtrResample, OtrTachoMode, OtrTachoToSpeed, OtrEncoderRevs01

OtrTachoToSpeed

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Aus dem Tachosignal, das z.B. in Form von Encoder-Pulsen vorliegt, wird ein Drehzahlsignal berechnet, das in U/min skaliert ist. Vor Aufruf dieser Funktion muss mit der Funktion OtrTachoMode() die Art des Tachosignals festgelegt worden sein.

Deklaration:

```
OtrTachoToSpeed ( Tachosignal ) -> Drehzahlverlauf
```

Parameter:

Tachosignal	Tachosignal, entsprechend vorherigem Aufruf von OtrTachoMode() gedeutet.
Drehzahlverlauf	

Beschreibung:

Vor Aufruf dieser Funktion sollte wenigstens einmal OtrTachoMode() aufgerufen werden.

Die mit dieser Funktion ermittelte Drehzahl sollte nicht integriert werden. Falls das Integral (also der Winkel bzw. die Überrollungen) erforderlich ist, sollte stets OtrTachoToDist() benutzt werden. Die reinen Ordnungsanaylse-Funktikonen, die ein Abtasten über dem Winkel durchführen, z.B. OtrOrderSpectrum(), sollten auch nicht die Drehzahl aus OtrTachoToSpeed() erhalten. Stattdessen sollte OtrTachoMode() passend eingestellt werden und das Tachosignal direkt diesen Funktionen übergeben werden. Andere Funktionen hingegen benötigen nur die Drehzahl, nicht den Winkel, z.B. OtrTrackingLowPass(). Für diese Funktionen ist das Resutat von OtrTachoToSpeed() wiederum bestens geeignet.

Beispiele:

Die Drehzahl an einem Zahnrad soll bestimmt werden. Dazu wurde mit einem induktiven Aufnehmer ein rechteckiges Spannungssignal erzeugt. Diese Spannung wurde mit konstanter Abtastrate von 100Hz aufgezeichnet und steht im Kanal mit dem Namen Signal zur Verfügung. Die Spannung beträgt etwa 0V..1V in der Zahnlücke, etwa 4V..4V an der Spitze des Zahns. Das Zahnrad hat 12 Zähne.

```
OtrTachoMode ( 3, 0, 12, 0.0)
EncoderPulses = stri ( Signal, 3.5, 1.5 )
speed =OtrTachoToSpeed ( EncoderPulses )
```

Da die Spannung nicht geeignet vorliegt, wird sie so verändert, dass ein sauberer Nulldurchgang entsteht. Zuvor wird mit OtrTachoMode () die Art des Tachosignals eingestellt.

Siehe auch:

OtrOrderSpectrum, OtrResampleAAF, OtrResample, OtrTachoMode, OtrTachoToDist, OtrEncoderRevs01, OtrTachoToSpeedX, PulseDuration

OtrTachoToSpeedX

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Aus dem Tachosignal wird ein Drehzahlsignal vom Datentyp XY berechnet, das in U/min skaliert ist. Vor Aufruf dieser Funktion muss mit der Funktion OtrTachoMode() die Art des Tachosignals festgelegt worden sein.

Deklaration:

```
OtrTachoToSpeedX ( Tachosignal ) -> XY_Drehzahlverlauf
```

Parameter:

Tachosignal	Tachosignal, entsprechend vorherigem Aufruf von OtrTachoMode() gedeutet.
XY_Drehzahlverlauf	

Beschreibung:

Vor Aufruf dieser Funktion sollte wenigstens einmal OtrTachoMode() aufgerufen werden.

Nur die Signaltypen 'Cronos Impulszeitpunkt' und 'Sinusförmige Signale' werden unterstützt.

Geber mit fehlenden Zähnen werden ebenfalls nicht unterstützt.

Beim Signaltyp 'Cronos Impulszeitpunkt' darf nur höchstens ein Puls pro Abtastwert enthalten sein. Falls mehr als 1 Puls vorhanden ist, liefert die Funktion an den betreffenden Stellen einen Wert von -1.

Das Ergebnis liegt als XY-Datensatz vor. Zu jedem gefundenen Puls wird ein XY-Paar dem Ergebnis hinzugefügt.

Die x-Koordinate des Ergebnisses liefert die aufbereiteten Zeitpunkte der Pulse zur weiteren Verrechnung.

Die y-Koordinate enthält die Drehzahl, die sich aus dem Abstand zum nächst folgenden Impuls ergibt. Wenn eine Darstellung in Treppen erfolgt, repräsentiert der Wert der Drehzahl nicht nur den Punkt, sondern das gesamte Intervall, für das sie gilt. Weil am allerletzten Puls kein neuer Wert für die Drehzahl ermittelt werden kann, wird der Vorgängerwert benutzt.

Beispiele:

Die Drehzahl an einem Zahnrad soll bestimmt werden. Ein Messkanal mit Impulszeitpunkten liegt vor.

```
OtrTachoMode ( 2, 0, 1, 0)
speed =OtrTachoToSpeedX ( CronosPulseTimes )
pulse_times = speed.x
```

Siehe auch:

OtrTachoMode, OtrTachoToSpeed

OtrTimeOrderSpectrum

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Das Ordnungsspektrum wird aus den Zeitverläufen von Schwingung und Tachosignal in Abhängigkeit von der Zeit bestimmt.

Deklaration:

```
OtrTimeOrderSpectrum ( Schwingung, Tachosignal, dx, Auflösung, OrdnungMax, Fenstertyp, Speed_Low ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Tachosignal	Tachosignal, Deutung entsprechend OtrTachoMode(). Standard ist ein Drehzahlverlauf, skaliert in U/min.
dx	Abtastzeit bzw. zeitliche Auflösung des Resultats. In diesem Abstand werden Ordnungsspektren ins Ergebnis eingetragen.
Auflösung	Die Auflösung des Ordnungsspektrums. 0.1, wenn 0.1 Ordnungen der Abstand der Linien im Ordnungsspektrum ist. Die Auflösung muss ein ganzzahliger Teiler von 1.0 sein, also [1, 1/2, 1/3, 1/4, ..]
OrdnungMax	Die höchste im Ordnungsspektrum angezeigte Ordnung(slinie).
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
	6 : Hamming (RMS=1)
	7 : Hanning (RMS=1)
	8 : Blackman (RMS=1)
	9 : Blackman / Harris (RMS=1)
	10 : Flat Top (RMS=1)
Speed_Low	Untergrenze der Drehzahl, in U/min skaliert. Ist der Betrag der Drehzahl <= diesem Wert, wird das Ordnungsspektrum auf null gesetzt.
Ergebnis	Ordnungsspektrum abhängig von der Zeit

Beschreibung:

Die Funktion ermittelt zu jedem Zeitpunkt im Ergebnis ein Ordnungsspektrum. Das erste erhält die Zeit x0 des Schwingungssignals. Dann folgen weitere im Abstand, den der Parameter dx vorgibt.

Der Kehrwert der Auflösung gibt an, über wie viele Umdrehungen das Ordnungsspektrum gebildet wird. Z.B. bei Auflösung = 0.1 wird jedes Spektrum aus 10 Umdrehungen bestimmt.

Die Spektrallinien sind als Effektivwerte angegeben.

Da intern die FFT mit einer etwas größeren Anzahl von Daten (einer Zweierpotenz) berechnet wird, werden einige Spektrallinien abgeschnitten. Im sichtbaren Spektrum ist also nicht mehr die volle Leistung enthalten. Das Rechteck-Fenster für die FFT ist sehr empfohlen, wenn ein hohe Frequenz-Auflösung gewünscht ist.

Ein mitlaufender Butterworth-Tiefpass wird als Antialiasing-Filter eingesetzt. Das so gefilterte Signal wird über dem Winkel abgetastet. Dann wird eine FFT berechnet. Aus dieser Vorgehensweise folgt, dass beim Abtasten zwischendurch deutlich höhere Ordnungslinien im Signal vorhanden sind als letztendlich im resultierenden Spektrum. Die 3dB-Ordnung des Antialiasing-Filter liegt deutlich hinter den im resultierenden Spektrum liegenden Ordnungslinien.

Die Funktion arbeitet sinnvoll für:

- OrdnungMax < 10 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl.

Ein berechnetes Ordnungsspektrum resultiert aus einem gewissen Zeitabschnitt der Eingangsdaten. Typisch liegen dabei mehrere Umdrehungen (entsprechend Drehzahlsignal) in diesem Zeitabschnitt. Bei niedriger Drehzahl kann dieser Zeitabschnitt lang sein. Der Zeitpunkt für die Übernahme ins Ergebnis ist die Mitte dieses Zeitabschnittes.

Der Parameter dx wird typisch so gewählt, dass bei erwarteter höchster Drehzahl eine gewisse Überlappung der Ordnungsspektren entsteht. Die Überlappung liegt dabei typisch zwischen 50% und 90%. Die kürzeste zeitliche Dauer zu einem Ordnungsspektrum ist 60 / (Maximale_Drehzahl * Auflösung). Bei 50% Überlappung wird diese Dauer durch 2, bei 90% Überlappung durch 10 geteilt. Wenn dx zu klein ist, erhält man zu viele (fast) gleiche Ordnungsspektren. Wenn zu groß, gehen eventuell wichtige Ordnungsspektren verloren.

dx wird auf ein ganzes Vielfaches der Abtastzeit des Schwingungssignals gerundet.

Der Parameter Speed_Low sorgt dafür, dass bei längeren Zeitdatensätzen auch niedrige Drehzahlen oder gar ein Stillstand (Drehzahl null) behandelt werden. Da in solchen Zeitbereichen keine Abtastung über dem Winkel erfolgt, entstehen keine neuen Ordnungsspektren. Alte Spektren werden dann den Bereich nicht mehr auffüllen. Stattdessen entstehen Spektren mit Nullen.

Liegt im Zeitabschnitt eines Ordnungsspektrums mindestens ein durch Speed_Low erfasster Drehzahlwert vor, wird das gesamte Ordnungsspektrum auf null gesetzt.

Zu Beginn oder am Ende des gesamten Zeitbereichs kommt es vor, dass keine Ordnungsspektren am Rand vorliegen. Dann wird zum Rand hin mit dem nächsten vorhandenen Ordnungsspektrum aufgefüllt.

Liegen für mittlere Zeitpunkte des Ergebnisse keine neuen Ordnungsspektren vor, wird der Vorgänger benutzt.

Das mitlaufende Antialiasing-Filter hat eine gewisse Einschwingzeit. Die Ergebnisse dieser Filterung werden ignoriert, solange wie das Filter (noch stark) einschwingt. Deshalb wirken sich die ersten Messwerte nicht auf die ersten Ordnungsspektren aus.

Die Fensterfunktion für die benutzte FFT hat i.a. einen Mittelwert von 1. Nur die mit RMS=1 gekennzeichneten haben einen Effektivwert von 1.

Gegenüber Mittelwert=1 ist bei RMS=1 das Ergebnis durch sqrt(ENBW=Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch sqrt(1.5) im Fall des Hanning-Fensters.

- Die Drehzahl sollte sich nur langsam ändern. Ist die Drehzahl oder OrdnungMax wesentlich größer als sinnvoll, steigt die Rechenzeit enorm, ohne den Informationsgehalt zu erhöhen.
- Aus dem Tachosignal muss durch Integration (Aufsummieren) der Drehwinkel ermittelt werden. Ist das Tachosignal selbst ein Drehzahlverlauf (und kein Pulssignal), muss deshalb die Drehzahl sehr genau vorliegen.
- Die Abtastzeit des Tachosignals darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Das Ergebnis ist ein segmentierter Datensatz. Jedes Segment ist ein Ordnungsspektrum.
- Vor Aufruf dieser Funktion sollte wenigstens einmal OtrTachoMode() aufgerufen werden.
- Für das Abtasten wird der Betrag der Drehzahl benutzt.

Beispiele:

Aus dem zeitlichen Verlauf einer Schwingung vib und der Drehzahl speed soll das Ordnungsspektrum abhängig von der Zeit bestimmt werden. Die Zeitsignale sind mit 0.2 ms abgetastet.

Die maximal vorliegende Drehzahl beträgt 6000 U/min.

```
OtrTachoMode ( 0, 0, 0, 0)
dx = 0.05 ; Nach Ablauf von so vielen Sekunden erscheint ein neues Ordnungsspektrum im Ergebnis.
Resolution = 0.1 ; Auflösung des Spektrums, also 1/10 Ordnungen sichtbar, Berechnung über 10 Umdrehungen
OrderMax = 6.0 ; bis zu dieser Ordnung sollen Linien im Spektrum angezeigt werden.
Windowtype = 0 ; 0 Standard (Rechteck)
SpeedLow = 100 ; Sinkt die Drehzahl auf diesen Wert oder tiefer, soll das als Stillstand gedeutet werden. Das Ergebnis wird in diesem Zeitbereich auf null gesetzt.
OTime = OtrTimeOrderSpectrum ( vib, speed, dx, Resolution, OrderMax, Windowtype, SpeedLow )
```

Hier gilt: OrderMax = 6.0 < 10 / (0.0002 * 6000) = 8.3

Aufgrund von Resolution erfolgt die Berechnung eines Ordnungsspektrum über 10 Umdrehungen. Bei maximaler Drehzahl dauern diese 10 Umdrehungen so viele Sekunden: 10/(6000/60) = 0.1. Bei einer

angenommenen Überlappung von 50% muss dx = 0.05 [Sekunden] gewählt werden.
Mit OtrTachoMode() wird vorher die Art des verwendeten Tachosignals festgelegt.

Siehe auch:
OtrOrderSpectrum, OtrTachoMode

OtrTrackingBandPass

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Mitlaufendes Bandpassfilter. Ein Schwingungssignal wird bandpassgefiltert, wobei die Mittenfrequenz des Filters von der Drehzahl abhängt. Der zeitliche Verlauf einer Ordnung wird bestimmt.

Deklaration:

```
OtrTrackingBandPass ( Schwingung, Drehzahl, OrdnungMitte, BreiteProzent, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert.
OrdnungMitte	Ordnung(slinie), bei der die Mittenfrequenz des Bandpasses liegt.
BreiteProzent	Gesamtbreite in Prozent. Empfohlen 10%...100%. Wertebereich [0.01 ... 10000.0]. Z.B. bei 30% Breite ist das Verhältnis von oberer zu unterer Grenzfrequenz des Bandpasses 1.30.
FilterOrdnung	Die Filter-Ordnung des Bandpassfilters (2, 4, 6, 8, 10)
Ergebnis	Zeitsignal mit dem bandpassgefiltertem Signal

Beschreibung:

Die Funktion bestimmt den zeitlichen Verlauf des Signalanteils, der zu einer bestimmten Ordnung gehört. Durch einen schmalen Bandpass kann so z.B. der Verlauf jeder beliebigen Bruchordnung bestimmt werden. Ggf. sollte anschließend die Funktion OtrTrackingExpoRms() angewendet werden.

Der Absolutbetrag der Drehzahl wird benutzt.

Die intern gewählte Mittenfrequenz des Filters liegt bei

Mittenfrequenz = OrdnungMitte * (Aktuelle_Drehzahl / 60)

Die obere Grenzfrequenz liegt oberhalb der Mittenfrequenz und ergibt sich aus der Breite des Filters. Die obere Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa 1 / (0.48 * Abtastzeit) kann keine Bandpassfilterung durchgeführt werden.

Die Funktion arbeitet sinnvoll für:

OrdnungMax << 28 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl.

OrdnungMax = OrdnungMitte * sqrt (1 + BreiteProzent / 100)

<< soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte.

- Die Drehzahl sollte sich nur langsam ändern.
- Die resultierende Mittenfrequenz sollte nicht merklich unter 1 Promille der maximal möglichen sinken. Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Sinkt die resultierende Mittenfrequenz zu tief, ist das Ergebnis weniger präzise.
- Zu beachten ist, dass Bandpässe eine gewisse Zeit beanspruchen, um einzuschwingen. Diese Zeit wächst extrem bei schmalen Filtern. Ein Bandpass der Breite 1% ist in diesem Sinn bereits extrem schmal. Eine Breite von 25% entspricht einem Terzfilter, eine Breite von 100% einem Oktavfilter.
- Das Filter hat Butterworth-Charakateristik. Es arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion FiltBp().

Beispiele:

Ein Schwingungssignal vib ist mit 0.5 ms abgetastet. Die Drehzahl speed kann bis 8000 U/min hochgehen. Der Zeitverlauf der 1.5ten Ordnung ist zu bestimmen.

```
om = 1.5 ; die 1.5te Ordnung wird gewählt.
fo = 6 ; Ein Bandpassfilter 6. Ordnung wird benutzt.
width = 30 ; 30% Gesamtbreite
tbp = OtrTrackingBandPass ( vib, speed, om, width, fo )
```

Es gilt: OrdnungMax = 1.5 * sqrt (1 + 30 / 100) = 1.7

Und damit: OrdnungMax = 1.7 << 28 / (0.0005 * 8000) = 7.0

Bestimmung des gleitenden Effektivwertes:

```
bp_rms = OtrTrackingExpoRms ( tbp, rpm, 3.0 )
```

Siehe auch:

OtrTrackingLowPass, OtrTrackingBandStop, OtrTrackingExpoRms, OtrRpmOrder, OtrFrequLine, FiltBp

OtrTrackingBandPassZ

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Mitlaufendes Bandpassfilter ohne Phasenverschiebung. Ein Schwingungssignal wird bandpassgefiltert, wobei die Mittenfrequenz des Filters von der Drehzahl abhängt. Der zeitliche Verlauf einer Ordnung wird bestimmt.

Deklaration:

```
OtrTrackingBandPassZ ( Schwingung, Drehzahl, OrdnungMitte, BreiteProzent, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert.
OrdnungMitte	Ordnung(slinie), bei der die Mittenfrequenz des Bandpasses liegt.
BreiteProzent	Gesamtbreite in Prozent. Empfohlen 10%...100%. Wertebereich [0.01 ... 10000.0]. Z.B. bei 30% Breite ist das Verhältnis von oberer zu unterer Grenzfrequenz des Bandpasses 1.30.
FilterOrdnung	Die Filter-Ordnung des Bandpassfilters (4, 8, 12, 16, 20)
Ergebnis	Zeitsignal mit dem bandpassgefiltertem Signal

Beschreibung:

Die Funktion bestimmt den zeitlichen Verlauf des Signalanteils, der zu einer bestimmten Ordnung gehört. Durch einen schmalen Bandpass kann so z.B. der Verlauf jeder beliebigen Bruchordnung bestimmt werden. Ggf. sollte anschließend die Funktion OtrTrackingExpoRms() angewendet werden.

Der Absolutbetrag der Drehzahl wird benutzt.

Die intern gewählte Mittenfrequenz des Filters liegt bei

Mittenfrequenz = OrdnungMitte * (Aktuelle_Drehzahl / 60)

Die obere Grenzfrequenz liegt oberhalb der Mittenfrequenz und ergibt sich aus der Breite des Filters. Die obere Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa 1 / (0.48 * Abtastzeit) kann keine Bandpassfilterung durchgeführt werden.

Die Funktion arbeitet sinnvoll für:

OrdnungMax << 28 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl.

OrdnungMax = OrdnungMitte * sqrt (1 + BreiteProzent / 100)

<< soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte.

- Die Drehzahl sollte sich nur langsam ändern.
- Die resultierende Mittenfrequenz sollte nicht merklich unter 1 Promille der maximal möglichen sinken. Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Sinkt die resultierende Mittenfrequenz zu tief, ist das Ergebnis weniger präzise.
- Zu beachten ist, dass Bandpässe eine gewisse Zeit beanspruchen, um einzuschwingen. Diese Zeit wächst extrem bei schmalen Filtern. Ein Bandpass der Breite 1% ist in diesem Sinn bereits extrem schmal. Eine Breite von 25% entspricht einem Terzfilter, eine Breite von 100% einem Oktavfilter.
- Das Filter hat Butterworth-Charakateristik. Es arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion FiltBpZ().

Der Datensatz wird einmal vorwärts, einmal rückwärts gefiltert. Dabei wird aber ein Filter benutzt, das zwar die vorgegebene Charakteristik, aber als Ordnung nur die Hälfte der angegebenen Ordnung hat.

Der Amplituden-Frequenzgang ist so normiert, dass bei der Grenzfrequenz eine Dämpfung von 3dB vorliegt.

Das Filter ist nicht kausal. Damit gilt die zeitliche Reihenfolge von Ursache und Wirkung nicht mehr.

Einschwingvorgänge gibt es in beide Richtungen, insbesondere auch nicht nur zu Beginn, sondern auch am Ende des Datensatzes.

Die Phase hat idealerweise den Wert null. Praktisch trifft das aber nur auf den eingeschwungenen Zustand zu, also nicht auf die Randbereiche.

Beispiele:

Ein Schwingungssignal vib ist mit 0.5 ms abgetastet. Die Drehzahl speed kann bis 8000 U/min hochgehen. Der Zeitverlauf der 1.5ten Ordnung ist zu bestimmen.

Das Filter soll das Signal nicht in seiner Phase verschieben.

```
om = 1.5 ; die 1.5te Ordnung wird gewählt.
fo = 6 ; Ein Bandpassfilter 6. Ordnung wird benutzt.
width = 30 ; 30% Gesamtbreite
```

```
tbp = OtrTrackingBandPassZ ( vib, speed, om, width, fo )
```

Es gilt: OrdnungMax = 1.5 * sqrt (1 + 30 / 100) = 1.7

Und damit: OrdnungMax = 1.7 << 28 / (0.0005 * 8000) = 7.0

Bestimmung des gleitenden Effektivwertes:

```
bp_rms = OtrTrackingExpoRms ( tbp, rpm, 3.0 )
```

Siehe auch:

OtrTrackingLowPass, OtrTrackingBandStop, OtrTrackingExpoRms, OtrRpmOrder, OtrFrequLine, FiltBp

OtrTrackingBandStop

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Mitlaufendes Bandsperrefilter. Ein Schwingungssignal wird mit einer Bandsperre gefiltert, wobei die Mittenfrequenz des Filters von der Drehzahl abhängt.

Deklaration:

```
OtrTrackingBandStop ( Schwingung, Drehzahl, OrdnungMitte, BreiteProzent, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert.
OrdnungMitte	Ordnung(slinie), bei der die Mittenfrequenz der Bandsperre liegt.
BreiteProzent	Gesamtbreite in Prozent. Empfohlen 10%...100%. Wertebereich [0.01 ... 10000.0]. Z.B. bei 30% Breite ist das Verhältnis von oberer zu unterer Grenzfrequenz der Bandsperre 1.30.
FilterOrdnung	Die Filter-Ordnung des Bandsperrefilters (2, 4, 6, 8, 10)
Ergebnis	Zeitsignal mit dem bandsperregefiltertem Signal

Beschreibung:

Die Funktion bestimmt den zeitlichen Verlauf der Signalanteile, die gerade nicht zu einer bestimmten Ordnung gehören. Durch eine schmale Bandsperre kann so z.B. eine beliebige Ordnung aus dem Signal eliminiert werden.

Der Absolutbetrag der Drehzahl wird benutzt.

Die intern gewählte Mittenfrequenz des Filters liegt bei

Mittenfrequenz = OrdnungMitte * (Aktuelle_Drehzahl / 60)

Die obere Grenzfrequenz liegt oberhalb der Mittenfrequenz und ergibt sich aus der Breite des Filters. Die obere Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa 1 / (0.48 * Abtastzeit) kann keine Bandsperrefilterung durchgeführt werden.

Die Funktion arbeitet sinnvoll für:

OrdnungMax << 28 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl.

OrdnungMax = OrdnungMitte * sqrt (1 + BreiteProzent / 100)

<<soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte.

- Die Drehzahl sollte sich nur langsam ändern.
- Die resultierende Mittenfrequenz sollte nicht merklich unter 1 Promille der maximal möglichen sinken.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Zu beachten ist, dass Bandsperre eine gewisse Zeit beanspruchen, um einzuschwingen. Diese Zeit ist besonders groß bei schmalen Filtern. Eine Bandsperre der Breite 1% ist in diesem Sinn bereits extrem schmal.
- Das Filter hat Butterworth-Charakateristik. Es arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion FiltBs(), aber mit anderer Wahl der Startwerte.

Beispiele:

Ein Schwingungssignal vib ist mit 1 ms abgetastet. Die Drehzahl speed kann bis 8000 U/min hochgehen. Die 2.5te Ordnung ist zu eliminieren.

```
om = 2.5 ; die 2.5te Ordnung wird gewählt.
fo = 4 ; Ein Bandsperrefilter 4. Ordnung wird benutzt.
width = 30 ; 30% Gesamtbreite
tbs = OtrTrackingBandStop ( vib, speed, om, width, fo )
```

Es gilt: OrdnungMax = 2.5 * sqrt (1 + 30 / 100) = 2.9

Und damit: OrdnungMax = 2.9 << 28 / (0.001 * 8000) = 3.5

Siehe auch:

OtrTrackingLowPass, OtrTrackingBandPass, OtrTrackingExpoRms, OtrRpmOrder, OtrFrequLine, FiltBs

OtrTrackingBandStopZ

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Mitlaufendes Bandsperrefilter ohne Phasenverschiebung. Ein Schwingungssignal wird mit einer Bandsperre gefiltert, wobei die Mittenfrequenz des Filters von der Drehzahl abhängt.

Deklaration:

```
OtrTrackingBandStopZ ( Schwingung, Drehzahl, OrdnungMitte, BreiteProzent, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert.
OrdnungMitte	Ordnung(slinie), bei der die Mittenfrequenz der Bandsperre liegt.
BreiteProzent	Gesamtbreite in Prozent. Empfohlen 10%...100%. Wertebereich [0.01 ... 10000.0]. Z.B. bei 30% Breite ist das Verhältnis von oberer zu unterer Grenzfrequenz der Bandsperre 1.30.
FilterOrdnung	Die Filter-Ordnung des Bandsperrefilters (4, 8, 12, 16, 20)
Ergebnis	Zeitsignal mit dem bandsperregefiltertem Signal

Beschreibung:

Die Funktion bestimmt den zeitlichen Verlauf der Signalanteile, die gerade nicht zu einer bestimmten Ordnung gehören. Durch eine schmale Bandsperre kann so z.B. eine beliebige Ordnung aus dem Signal eliminiert werden.

Der Absolutbetrag der Drehzahl wird benutzt.

Die intern gewählte Mittenfrequenz des Filters liegt bei

Mittenfrequenz = OrdnungMitte * (Aktuelle_Drehzahl / 60)

Die obere Grenzfrequenz liegt oberhalb der Mittenfrequenz und ergibt sich aus der Breite des Filters. Die obere Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa 1 / (0.48 * Abtastzeit) kann keine Bandsperrefilterung durchgeführt werden.

Die Funktion arbeitet sinnvoll für:

OrdnungMax << 28 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl.

OrdnungMax = OrdnungMitte * sqrt (1 + BreiteProzent / 100)

<<soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte.

- Die Drehzahl sollte sich nur langsam ändern.
- Die resultierende Mittenfrequenz sollte nicht merklich unter 1 Promille der maximal möglichen sinken.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Zu beachten ist, dass Bandsperre eine gewisse Zeit beanspruchen, um einzuschwingen. Diese Zeit ist besonders groß bei schmalen Filtern. Eine Bandsperre der Breite 1% ist in diesem Sinn bereits extrem schmal.
- Das Filter hat Butterworth-Charakateristik. Es arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion FiltBsZ(), aber mit anderer Wahl der Startwerte.

Der Datensatz wird einmal vorwärts, einmal rückwärts gefiltert. Dabei wird aber ein Filter benutzt, das zwar die vorgegebene Charakteristik, aber als Ordnung nur die Hälfte der angegebenen Ordnung hat.

Der Amplituden-Frequenzgang ist so normiert, dass bei der Grenzfrequenz eine Dämpfung von 3dB vorliegt.

Das Filter ist nicht kausal. Damit gilt die zeitliche Reihenfolge von Ursache und Wirkung nicht mehr.

Einschwingvorgänge gibt es in beide Richtungen, insbesondere auch nicht nur zu Beginn, sondern auch am Ende des Datensatzes.

Die Phase hat idealerweise den Wert null. Praktisch trifft das aber nur auf den eingeschwungenen Zustand zu, also nicht auf die Randbereiche.

Beispiele:

Ein Schwingungssignal vib ist mit 1 ms abgetastet. Die Drehzahl speed kann bis 8000 U/min hochgehen. Die 2.5te Ordnung ist zu eliminieren.

Das Filter soll das Signal nicht in seiner Phase verschieben.

```
om = 2.5 ; die 2.5te Ordnung wird gewählt.
fo = 4 ; Ein Bandsperrefilter 4. Ordnung wird benutzt.
width = 30 ; 30% Gesamtbreite
tbs = OtrTrackingBandStopZ ( vib, speed, om, width, fo )
```

Es gilt: OrdnungMax = 2.5 * sqrt (1 + 30 / 100) = 2.9

Und damit: OrdnungMax = 2.9 << 28 / (0.001 * 8000) = 3.5

Siehe auch:

OtrTrackingLowPass, OtrTrackingBandPass, OtrTrackingExpoRms, OtrRpmOrder, OtrFrequLine, FiltBs

OtrTrackingExpoRms

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Berechnung des gleitenden Effektivwertes mit exponentiell gewichteter Mittelung, wobei die Zeitkonstante von der Drehzahl abhängt.

Deklaration:

```
OtrTrackingExpoRms ( Schwingung, Drehzahl, GlättungUmdrehungen ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert.
GlättungUmdrehungen	So vielen Umdrehungen entspricht die Zeitkonstante.
Ergebnis	Geglätteter Zeitverlauf

Beschreibung:

Wird beim exponentiellen Effektivwert mit der Funktion ExpoRms () eine Zeitkonstante angegeben, so ist das die Zeitkonstante für die Exponentialfunktion. Bei der Funktion OtrTrackingExpoRms () wird die Konstante nicht als Zeit angegeben, sondern als eine Anzahl von Umdrehungen. Je nach vorliegender Drehzahl wird die Zeitkonstante dann bestimmt. Die resultierende Zeitkonstante ist umgekehrt proportional zur Drehzahl.

Zeitkonstante = GlättungUmdrehungen * Aktuelle_Drehzahl / 60

GlättungUmdrehungen ist eine reelle Zahl.

Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.

Die Funktion arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion ExpoRms(), aber mit anderer Wahl der Startwerte.

Beispiele:

Durch Bandpassfilterung wird der Schwingungsverlauf einer Ordnungslinie gewonnen. Dieser Zeitverlauf soll als Effektivwert abhängig von der Zeit dargestellt werden. Gegeben sind eine Schwingung vib und ein Drehzahlverlauf speed.

```
tbp = OtrTrackingBandPass ( vib, speed, 1.5, 30, 4 )
_Smooth = 3.0 ; Anzahl der Umdrehungen
bp_rms = OtrTrackingExpoRms ( tbp, speed, _Smooth )
```

Der gleitende Effektivwert wird so gebildet, dass die Zeitkonstante 3 Umdrehungen beträgt.

Siehe auch:

OtrTrackingBandPass, OtrTrackingBandStop, ExpoRms

OtrTrackingHighPass

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Mitlaufendes Hochpassfilter. Schleppfilter. Tracking Filter. Ein Schwingungssignal wird hochpassgefiltert, wobei die Grenzfrequenz des Filters von der Drehzahl abhängt.

Deklaration:

```
OtrTrackingHighPass ( Schwingung, Drehzahl, Ordnung3dB, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert
Ordnung3dB	Ordnung(slinie), bei der der Hochpass um 3dB dämpft
FilterOrdnung	Die Filter-Ordnung des Hochpassfilters (1 ... 10)
Ergebnis	Gefiltertes Schwingungssignal.

Beschreibung:

Der Absolutbetrag der Drehzahl wird benutzt.

Die intern gewählte Grenzfrequenz des Filters liegt bei

Grenzfrequenz = Ordnung3dB * (Aktuelle_Drehzahl / 60)

Die Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen, um eine Filterwirkung zu erzielen. Die Funktion arbeitet sinnvoll für:

Ordnung3dB <<24 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl. << soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte. Umgekehrt heißt es, dass die maximale Drehzahl nicht zu hoch werden sollte.

- Die Drehzahl sollte sich nur langsam ändern. Die Drehzahl sollte nicht wesentlich unter 1% der maximal möglichen sinken.
- Die obere Grenzfrequenz des Hochpassfilters muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa (0.4 * Abtastfrequenz) kann keine Filterung mehr durchgeführt werden.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Das Filter hat Butterworth-Charakateristik. Es arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion FiltHp(), aber mit anderer Wahl der Startwerte.

Beispiele:

Ein Schwingungssignal vib ist mit 0.2 ms abgetastet. Die Drehzahl speed kann bis 6000 U/min hochgehen. Anteile unterhalb der 10. Ordnung sollen unterdrückt werden.

```
o3dB = 10.0 ; Der Hochpass dämpft um 3dB bei dieser Ordnung
fo = 2 ; Ein Hochpassfilter 2. Ordnung wird benutzt
tlp = OtrTrackingHighPass ( vib, speed, o3dB, fo )
```

Es gilt: OrdnungMax = 10.0 << 24 / (0.0002 * 6000) = 20.0.

Das Filter ist so dimensioniert, dass es bei der 10. Ordnung um 3 dB dämpft.

Siehe auch:

OtrTrackingBandPass, OtrTrackingLowPass, OtrResampleAAF, OtrRpmPresentation, FiltHp

OtrTrackingHighPassZ

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Mitlaufendes Hochpassfilter ohne Phasenverschiebung. Schleppfilter. Tracking Filter. Ein Schwingungssignal wird hochpassgefiltert, wobei die Grenzfrequenz des Filters von der Drehzahl abhängt.

Deklaration:

```
OtrTrackingHighPassZ ( Schwingung, Drehzahl, Ordnung3dB, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert
Ordnung3dB	Ordnung(slinie), bei der der Hochpass um 3dB dämpft
FilterOrdnung	Die Filter-Ordnung des Hochpassfilters (2, 4, 6, .. 20)
Ergebnis	Gefiltertes Schwingungssignal.

Beschreibung:

Der Absolutbetrag der Drehzahl wird benutzt.

Die intern gewählte Grenzfrequenz des Filters liegt bei

Grenzfrequenz = Ordnung3dB * (Aktuelle_Drehzahl / 60)

Die Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen, um eine Filterwirkung zu erzielen. Die Funktion arbeitet sinnvoll für:

Ordnung3dB <<24 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl. << soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte. Umgekehrt heißt es, dass die maximale Drehzahl nicht zu hoch werden sollte.

- Die Drehzahl sollte sich nur langsam ändern. Die Drehzahl sollte nicht wesentlich unter 1% der maximal möglichen sinken.
- Die obere Grenzfrequenz des Hochpassfilters muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa (0.4 * Abtastfrequenz) kann keine Filterung mehr durchgeführt werden.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Das Filter arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion FiltHpZ(), aber mit anderer Wahl der Startwerte.

Der Datensatz wird einmal vorwärts, einmal rückwärts gefiltert. Dabei wird aber ein Filter benutzt, das zwar die vorgegebene Charakteristik, aber als Ordnung nur die Hälfte der angegebenen Ordnung hat.

Der Amplituden-Frequenzgang ist so normiert, dass bei der Grenzfrequenz eine Dämpfung von 3dB vorliegt.

Das Filter ist nicht kausal. Damit gilt die zeitliche Reihenfolge von Ursache und Wirkung nicht mehr.

Einschwingvorgänge gibt es in beide Richtungen, insbesondere auch nicht nur zu Beginn, sondern auch am Ende des Datensatzes.

Die Phase hat idealerweise den Wert null. Praktisch trifft das aber nur auf den eingeschwungenen Zustand zu, also nicht auf die Randbereiche.

Beispiele:

Ein Schwingungssignal vib ist mit 0.2 ms abgetastet. Die Drehzahl speed kann bis 6000 U/min hochgehen. Anteile unterhalb der 10. Ordnung sollen unterdrückt werden.

Das Filter soll das Signal nicht in seiner Phase verschieben.

```
o3dB = 10.0 ; Der Hochpass dämpft um 3dB bei dieser Ordnung
fo = 2 ; Ein Hochpassfilter 2. Ordnung wird benutzt
thp = OtrTrackingHighPassZ ( vib, speed, o3dB, fo )
```

Es gilt: OrdnungMax = 10.0 << 24 / (0.0002 * 6000) = 20.0.

Das Filter ist so dimensioniert, dass es bei der 10. Ordnung um 3 dB dämpft.

Siehe auch:

OtrTrackingHighPass, OtrTrackingLowPassZ, OtrResampleAAF, OtrRpmPresentation, FiltHpZ, FiltHp

OtrTrackingLowPass

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Mitlaufendes Tiefpassfilter. Schleppfilter. Tracking Filter. Ein Schwingungssignal wird tiefpassgefiltert, wobei die Grenzfrequenz des Filters von der Drehzahl abhängt.

Deklaration:

```
OtrTrackingLowPass ( Schwingung, Drehzahl, Ordnung3dB, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert
Ordnung3dB	Ordnung(slinie), bei der der Tiefpass um 3dB dämpft
FilterOrdnung	Die Filter-Ordnung des Tiefpassfilters (1 ... 10)
Ergebnis	Gefiltertes Schwingungssignal.

Beschreibung:

Der Absolutbetrag der Drehzahl wird benutzt.

Die intern gewählte Grenzfrequenz des Filters liegt bei

Grenzfrequenz = Ordnung3dB * (Aktuelle_Drehzahl / 60)

Die Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen, um eine Filterwirkung zu erzielen. Die Funktion arbeitet sinnvoll für:

Ordnung3dB <<24 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl. << soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte. Umgekehrt heißt es, dass die maximale Drehzahl nicht zu hoch werden sollte.

- Die Drehzahl sollte sich nur langsam ändern. Die Drehzahl sollte nicht wesentlich unter 1% der maximal möglichen sinken.
- Die obere Grenzfrequenz des Tiefpassfilters muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa (0.4 * Abtastfrequenz) kann keine Filterung mehr durchgeführt werden.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Das Filter hat Butterworth-Charakateristik. Es arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion FiltTp(), aber mit anderer Wahl der Startwerte.

Beispiele:

Ein Schwingungssignal vib ist mit 0.2 ms abgetastet. Die Drehzahl speed kann bis 6000 U/min hochgehen. Anteile oberhalb der 10. Ordnung sollen unterdrückt werden.

```
o3dB = 10.0 ; Der Tiefpass dämpft um 3dB bei dieser Ordnung
fo = 6 ; Ein Tiefpassfilter 6. Ordnung wird benutzt
tlp = OtrTrackingLowPass ( vib, speed, o3dB, fo )
```

Es gilt: OrdnungMax = 10.0 << 24 / (0.0002 * 6000) = 20.0.

Das Tiefpassfilter ist so dimensioniert, dass es bei der 10. Ordnung um 3 dB dämpft, bei der 22. Ordnung um 40 dB. Unterhalb der 8.3ten Ordnung beträgt der Amplitudenfehler bereits weniger als 5%.

Siehe auch:

OtrTrackingBandPass, OtrResampleAAF, OtrRpmPresentation, FiltTp

OtrTrackingLowPassZ

Verfügbar ab: Enterprise Edition (OrderTracking-Kit)

Mitlaufendes Tiefpassfilter ohne Phasenverschiebung. Schleppfilter. Tracking Filter. Ein Schwingungssignal wird tiefpassgefiltert, wobei die Grenzfrequenz des Filters von der Drehzahl abhängt.

Deklaration:

```
OtrTrackingLowPassZ ( Schwingung, Drehzahl, Ordnung3dB, FilterOrdnung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals
Drehzahl	Zeitverlauf der Drehzahl. In U/min skaliert
Ordnung3dB	Ordnung(slinie), bei der der Tiefpass um 3dB dämpft
FilterOrdnung	Die Filter-Ordnung des Tiefpassfilters (2, 4, 6, .. 20)
Ergebnis	Gefiltertes Schwingungssignal.

Beschreibung:

Der Absolutbetrag der Drehzahl wird benutzt.

Die intern gewählte Grenzfrequenz des Filters liegt bei

Grenzfrequenz = Ordnung3dB * (Aktuelle_Drehzahl / 60)

Die Grenzfrequenz muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen, um eine Filterwirkung zu erzielen. Die Funktion arbeitet sinnvoll für:

Ordnung3dB <<24 / (Abtastzeit_Schwingung * max (Drehzahl))

wobei Abtastzeit_Schwingung die Abtastzeit des Signals Schwingung ist und max (Drehzahl) der maximal auftretende Wert der Drehzahl. << soll andeuten, dass die vorgebbare Ordnung deutlich kleiner sein sollte. Umgekehrt heißt es, dass die maximale Drehzahl nicht zu hoch werden sollte.

- Die Drehzahl sollte sich nur langsam ändern. Die Drehzahl sollte nicht wesentlich unter 1% der maximal möglichen sinken.
- Die obere Grenzfrequenz des Tiefpassfilters muss immer wesentlich unter der halben Abtastfrequenz des Schwingungssignals liegen. Oberhalb von etwa (0.4 * Abtastfrequenz) kann keine Filterung mehr durchgeführt werden.
- Die Abtastzeit der Drehzahl darf gleich oder ein ganzes Vielfaches der Abtastzeit der Schwingung sein.
- Das Filter arbeitet bei konstanter Drehzahl vergleichbar mit der Funktion FiltLpZ(), aber mit anderer Wahl der Startwerte.

Der Datensatz wird einmal vorwärts, einmal rückwärts gefiltert. Dabei wird aber ein Filter benutzt, das zwar die vorgegebene Charakteristik, aber als Ordnung nur die Hälfte der angegebenen Ordnung hat.

Der Amplituden-Frequenzgang ist so normiert, dass bei der Grenzfrequenz eine Dämpfung von 3dB vorliegt.

Das Filter ist nicht kausal. Damit gilt die zeitliche Reihenfolge von Ursache und Wirkung nicht mehr.

Einschwingvorgänge gibt es in beide Richtungen, insbesondere auch nicht nur zu Beginn, sondern auch am Ende des Datensatzes.

Die Phase hat idealerweise den Wert null. Praktisch trifft das aber nur auf den eingeschwungenen Zustand zu, also nicht auf die Randbereiche.

Beispiele:

Ein Schwingungssignal vib ist mit 0.2 ms abgetastet. Die Drehzahl speed kann bis 6000 U/min hochgehen. Anteile oberhalb der 10. Ordnung sollen unterdrückt werden.

Das Filter soll das Signal nicht in seiner Phase verschieben.

```
o3dB = 10.0 ; Der Tiefpass dämpft um 3dB bei dieser Ordnung
fo = 6 ; Ein Tiefpassfilter 6. Ordnung wird benutzt
tlp = OtrTrackingLowPassZ ( vib, speed, o3dB, fo )
```

Es gilt: OrdnungMax = 10.0 << 24 / (0.0002 * 6000) = 20.0.

Das Filter ist so dimensioniert, dass es bei der 10. Ordnung um 3 dB dämpft.

Siehe auch:

OtrTrackingLowPass, OtrTrackingBandPassZ, OtrResampleAAF, OtrRpmPresentation, FiltLpZ, FiltTp

ParametersPassed?

Die Funktion liefert die Anzahl der Parameter, die beim Aufruf einer Sequenz oder einer Sequenzfunktion übergeben wurden.

Deklaration:

```
ParametersPassed? ( ) -> Anzahl
```

Parameter:

Anzahl	Die Anzahl der beim Aufruf der Sequenz bzw. Sequenzfunktion übergebenen Parameter.

Beschreibung:

In einer Sequenzfunktion mit optionalen Parametern sollte vor der Verwendung eines optionalen Parameters im Code zunächst geprüft werden, ob dieser vom Aufrufer übergeben wurde und somit die entsprechende Variable überhaupt vorhanden ist. Mittels der Funktion ParametersPassed?() kann die Anzahl der tatsächlich übergebenen Parameter bestimmt werden. Die fehlenden Parameter können dann z.B. entweder selbst mit Standardwerten angelegt werden oder deren Verwendung durch die bedingte Ausführung von Code-Zweigen verhindert werden. Das erste Beispiel demonstriert entsprechende Techniken.

Auch beim Aufruf von klassischen Unter-Sequenzen mit dem SEQUENCE-Befehl ist die Anzahl der tatsächlich übergebenen Parameter erst zur Laufzeit bekannt. ParametersPassed?() liefert hier ebenfalls die Anzahl der übergebenen Parameter und damit die Information, wie viele der formalen Parameter PA1 .. PA20 gültig sind.

Die Funktion liefert -1, wenn sie weder innerhalb einer Sequenzfunktion noch in einer mit dem SEQUENCE-Befehl gestarteten Sequenz aufgerufen wird.

Beispiele:

Eine Sequenzfunktion "CalcSum" ist mit 4 Parametern definiert, wobei die letzten 2 optional sind. Die Funktion soll die Summe über alle angegebenen Parameter berechnen.

Deklaration:

```
!CalcSum( summand1, summand2 [, summand3] [, summand4]) => totalSum
```

Im Code der Sequenzfunktion muss also berücksichtigt werden, dass [summand3] und/oder [summand4] vom Aufrufer nicht übergeben wurden. Zugriffe auf diese beiden Parameter im Code würden dann zu einem Fehler führen, da die Variablen dann zur Laufzeit nicht bekannt sind.

Variante #1: Bedingte Ausführung von Code-Zweigen, abhängig von der Anzahl der übergebenen Variablen:

```
parCount = ParametersPassed?()
; parCount hat den Wert 2, 3 oder 4

totalSum = summand1 + summand2
SWITCH parCount
   CASE 3
      ; der 4. Parameter fehlt
      totalSum = summand1 + summand2 + summand3
   CASE 4
      ; alle Parameter vorhanden
      totalSum = summand1 + summand2 + summand3 + summand4
END
```

Variante #2: Nicht übergebene Parameter selbst anlegen und mit Standardwerten initialisieren:

```
parCount = ParametersPassed?()  ; 2, 3 or 4

IF parCount < 3
   ; der 3. Parameter fehlt, mit Standardwert initialisieren
   summand3 = 0
END
IF parCount < 4
   ; der 4. Parameter fehlt, mit Standardwert initialisieren
   summand4 = 0
END

totalSum = summand1 + summand2 + summand3 + summand4
```

Variante #3: Die übergebenen Parameter werden in einer Schleife verarbeitet. Setzt voraus, dass die Parameter-Benennung dem Schema "FesterName"+ laufende Nummer gehorcht und kann bei einer größeren Zahl von optionalen Parametern sinnvoll sein.

```
parCount = ParametersPassed?()  ; 2, 3 or 4
totalSum = summand1 + summand2

FOR i = 3 TO parCount
   txParName = "summand"+ TForm(i, "")
```

```
   totalSum = totalSum + <txParName>
END
```

Die folgende Untersequenz wendet auf den ersten übergebenen Parameter ein Tiefpass-Filter (Butterworth, 4. Ordnung) an. Der zweite Parameter spezifiziert die Grenzfrequenz. Wenn dieser weggelassen wird, wird der Standardwert 10Hz verwendet. Wenn gar kein oder mehr als 2 Parameter übergeben wurden, bricht die Sequenz mit einer Fehlermeldung ab.

```
OnError("ReturnFail")
parCount = ParametersPassed?()

SWITCH parCount
   CASE 1
      f_cutoff = 10 ; default value
   CASE 2
      f_cutoff = PA2
   DEFAULT
      ThrowError("Unexpected number of passed parameters!")
END
PA1 = FiltLP(PA1, 0, 0, 4, f_cutoff)
```

Siehe auch:

SEQUENCE, BoxVarSelector, VarGetInit

Unterstützt ab:

Version 2024

PAUSE

Die Sequenzausführung wird unterbrochen und eine Nachricht angezeigt, die vom Anwender quittiert werden muss.

Deklaration:

```
PAUSE Ausgabetext
```

Parameter:

Ausgabetext	Beliebiger Text, der angezeigt wird

Beschreibung:

Die Sequenz wird angehalten und ein Dialogfeld mit den Schaltflächen "OK" und "Abbrechen" sowie dem angegebenen Text gezeigt. Sie können dann Zwischenergebnisse innerhalb der Sequenz betrachten oder anders in die Sequenz eingreifen. Wenn Sie die Schaltfläche "OK" wählen, wird die Sequenz weiter ausgeführt. Wählen Sie die Schaltfläche "Abbrechen", wird die Ausführung der Sequenz abgebrochen.

- Sie können alternativ auch die Funktion BoxOutput() benutzen, um dem Benutzer Informationen anzuzeigen. Zur Eingabe von Texten oder Werten durch den Nutzer stehen die Funktionen BoxValue?() und BoxText?() zur Verfügung. Für einfache Entscheidungen ('Ja'/'Nein'-Abfragen) können Sie die Funktion BoxMessage() verwenden.
- Um die Sequenzausführung für eine vorgebbare Wartezeit zu unterbrechen, können Sie die Funktion Sleep() verwenden.
- Multithreading: Die Funktion darf nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

Beispiele:

```
SHOW x
PAUSE Erster Schritt
x = x + 1
PAUSE Berechnung fortführen?
x = x + 2
```

Der Originaldatensatz x und das erste Zwischenergebnis können in Ruhe betrachtet werden

Siehe auch:

BoxOutput, BoxMessage, BoxValue?, Sleep, EXITSEQUENCE

Peaks

Liefert die Anzahl der Pulse (Schwingungen) im Datensatz
Alternativer Name: **Pulse**

Deklaration:

```
Peaks ( Daten ) -> EwAnzahl
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND]
EwAnzahl	Anzahl der Schwingungen (Pulse)

Beschreibung:

Die Funktion Peaks() zählt die Impulse und Schwingungen in einem Datensatz. Ein Impuls beginnt, wenn die Werte eines Datensatzes größer Null werden. Ein Impuls endet, wenn die Werte kleiner oder gleich Null werden. Es wird mit der Auflösung von einem halben Impuls gezählt.

Ein Datensatz mit den Werten 0, 0, 0, 1, 0, 0 enthält demnach einen Impuls genauso wie der Datensatz mit den Werten 1, 2, 1, -1, -1, 2.

Ein Datensatz mit den Werten 0, 1, 0, 1 enthält 1.5 Impulse, ein Datensatz mit den Werten -1, 1, -1, 1, -1 genau 2 Impulse.

Vor Anwendung der Funktion Peaks() sollte das Signal durch Subtrahieren eines Offsets und Glätten oder auch mit der Schmitt-Trigger-Funktion geeignet konditioniert werden.

Beispiele:

```
three = Peaks(cos(Ramp(0, 0.1, 200)))
```

Ein Kosinus-förmiger Datensatz enthält 3 volle Perioden der Schwingung.

```
_peaks = Peaks(Smo5(NDdata) - 5)
```

Ein Datensatz, dessen Impulse mit einem y-Offset von 5 beaufschlagt sind, wird vor der Impulszählung geglättet und vom y-Offset befreit. Das Glätten soll ein falsches Impulszählen bei Störungen an den Impulsflanken verhindern.

```
_peaks = Peaks(sTri(NDdata, 5, 10))
```

Das Zählen von Impulsen ist besonders zuverlässig, wenn die Impulse vorher mit der Schmitt-Trigger-Funktion geformt wurden.

```
frequ = Peaks(NDdata)/(Leng?(NDdata) * xDel?(NDdata))
```

Die Impulsfrequenz wird ermittelt, indem die Zahl der Pulse durch die Zeitdauer dividiert wird.

Siehe auch:

STri, PulseDuration, OtrTachoToSpeed, Smo

Perio

Von einem periodischen Datensatz werden wahlweise die Mittelwerte, die Standardabweichungen, die obere oder die untere Hüllkurve über alle Perioden berechnet oder es wird eine spezielle Periode ausgegeben.

Deklaration:

```
Perio ( Daten, EwPeriodenLänge, EwOption ) -> Ergebnis
```

Parameter:

Daten	Periodischer Datensatz, der bearbeitet werden soll. Erlaubte Datentypen: [ND]
EwPeriodenLänge	Anzahl der Werte einer Periode
EwOption	Definiert die Art der Berechnung
	-1 : Ausgabe der Mittelwerte über alle Perioden
	-2 : Ausgabe der Standardabweichungen über alle Perioden
	-3 : Ausgabe der oberen Hüllkurve über alle Perioden (Maxima)
	-4 : Ausgabe der unteren Hüllkurve über alle Perioden (Minima)
	>=0 : Ausgabe der eingegebenen Periode
Ergebnis	Ergebnis entsprechend [EwOption]

Beschreibung:

Der Datensatz NDdaten wird in Perioden eingeteilt. Eine Periode enthält so viele Werte, wie als Parameter [EwPeriodenLänge] eingegeben werden. So viele Werte enthält auch der Datensatz NDperiode, der ausgegeben wird. Je nach Eingabe vom Parameter [EwOption] werden die Mittelwerte, die Standardabweichungen, die obere oder die untere Hüllkurve über alle Perioden berechnet oder es wird eine spezielle Periode ausgewählt.

Bei Eingabe von -1 für den Parameter [EwOption] werden die Mittelwerte über alle Perioden berechnet und ausgegeben. Insgesamt werden so viele Mittelwerte bestimmt, wie Werte in einer Periode enthalten sind. Der erste ausgegebene Mittelwert ist der Mittelwert des jeweils ersten Werts aller Perioden. Der zweite ausgegebene Mittelwert ist der Mittelwert des jeweils zweiten Werts aller Perioden usw.

Bei Eingabe von -2 für den Parameter [EwOption] werden die Standardabweichungen über alle Perioden berechnet und ausgegeben. Insgesamt werden so viele Streuungswerte bestimmt, wie Werte in einer Periode enthalten sind. Der erste ausgegebene Streuungswert ist die Standardabweichung des jeweils ersten Werts aller Perioden. Der zweite ausgegebene Streuungswert ist die Standardabweichung des jeweils zweiten Werts aller Perioden usw.

Bei Eingabe von -3 für den Parameter [EwOption] wird die obere Hüllkurve über alle Perioden berechnet und ausgegeben. Insgesamt werden so viele obere Hüllwerte (Maxima) bestimmt, wie Werte in einer Periode enthalten sind. Der erste ausgegebene obere Hüllwert ist das Maximum des jeweils ersten Werts aller Perioden. Der zweite ausgegebene obere Hüllwert ist das Maximum des jeweils zweiten Werts aller Perioden usw.

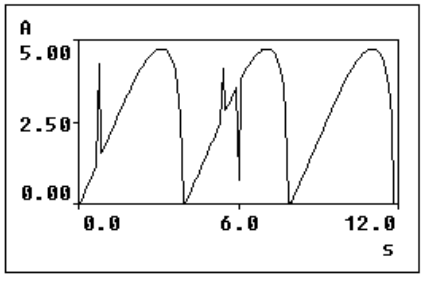
Bei Eingabe von -4 für den Parameter [EwOption] wird die untere Hüllkurve über alle Perioden berechnet und ausgegeben. Insgesamt werden so viele untere Hüllwerte bestimmt, wie Werte in einer Periode enthalten sind. Der erste ausgegebene untere Hüllwert ist das Minimum des jeweils ersten Werts aller Perioden. Der zweite ausgegebene untere Hüllwert ist das Minimum des jeweils zweiten Werts aller Perioden usw.

Bei Eingabe einer ganzen Zahl größer oder gleich 0 für den Parameter [EwOption] wird die dieser Zahl entsprechende Periode ausgegeben. Bei Eingabe von 0 wird die erste Periode ausgegeben. Bei Eingabe von 1 wird die zweite Periode ausgegeben usw.

- Falls die Periodenlänge größer als die Datensatzlänge ist, werden bei Eingabe von -2 für den Parameter [EwOption] alle Streuungswerte 0 gesetzt und bei Eingabe von -1, -3 , -4 oder 0 für den Parameter [EwOption] wird der ausgegebene Datensatz NDperiode mit Nullen ergänzt.
- Auch wenn die Datensatzlänge kein Vielfaches der Periodenlänge beträgt, werden bei Eingabe von -1, -2, -3 oder -4 für den Parameter [EwOption] alle auftretenden Werte des Datensatzes in die Berechnungen einbezogen und bei Eingabe von dem der letzten Periode entsprechenden Wert (>=0) wird der ausgegebene Datensatz NDperiode mit Nullen ergänzt.

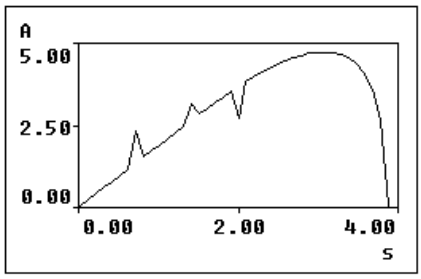
Beispiele:

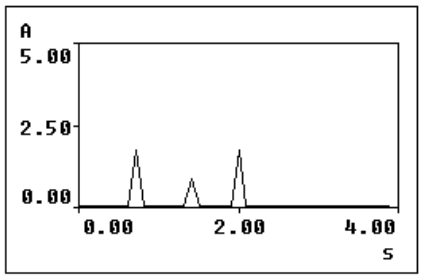
Der zur folgenden Abbildung gehörige Datensatz NDdaten soll mit der Funktion Perio bearbeitet werden. Der Datensatz besteht aus 120 Werten. Eine Periode soll aus 40 Werten bestehen:



```
NDmittel = Perio(NDdaten, 40, -1)
NDstreu = Perio(NDdaten, 40, -2)
```

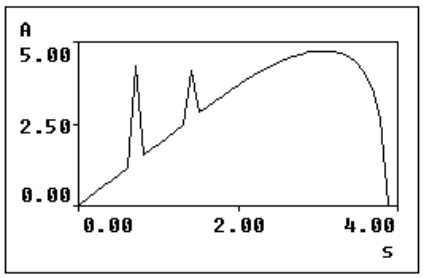
Als Ergebnis der Anwendung der Funktion Perio wird in imc FAMOS bei Eingabe von -1 für den Parameter EWwahl (Berechnung der Mittelwerte) der auf der linken Abbildung dargestellte Datensatz NDmittel ausgegeben. Bei Eingabe von -2 für den Parameter [EwOption] (Berechnung der Standardabweichungen) wird in imc FAMOS als Ergebnis der auf der rechten Abbildung dargestellte Datensatz NDstreu ausgegeben:

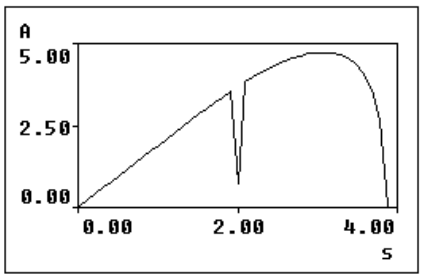





```
NDoben  = Perio(NDdaten, 40, -3)
NDunten = Perio(NDdaten, 40, -4)
```

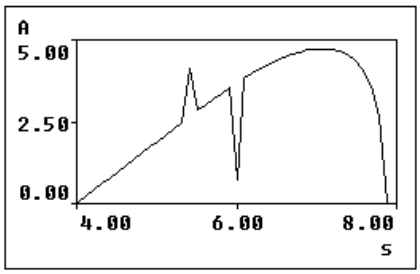
Als Ergebnis der Anwendung der Funktion Perio wird in imc FAMOS bei Eingabe von -3 für den Parameter EWwahl (Berechnung der oberen Hüllkurve) der auf der linken Abbildung dargestellte Datensatz NDoben ausgegeben. Bei Eingabe von -4 für den Parameter [EwOption] (Berechnung der unteren Hüllkurve) wird in imc FAMOS als Ergebnis der auf der rechten Abbildung dargestellte Datensatz NDunten ausgegeben:






```
NDspezi = Perio(NDdaten, 40, 1)
```

Als Ergebnis der Anwendung der Funktion Perio wird in imc FAMOS bei Eingabe von 1 für den Parameter [EwOption] (Ausgabe der zweiten Periode) der auf der folgenden Abbildung dargestellte Datensatz NDspezi ausgegeben:



PhaseContinuous

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein Phasenverlauf wird kontinuierlich (stetig) gemacht. 360 Grad Sprünge werden entfernt.

Deklaration:

```
PhaseContinuous ( Phase ) -> Ergebnis
```

Parameter:

Phase	Die Funktion kann auf komplexe Datensätze angewendet werden, die mit Betrag und Phase dargestellt sind. Dann wird nur die Phase bearbeitet. Insbesondere sind segmentierte Datensätze auch erlaubt. Die Funktion kann auch auf Phasenverläufe direkt (keine komplexen Datensätze) angewendet werden.
Ergebnis	Korrigierter Phasenverlauf

Beschreibung:

Da bei FFT-Berechnungen und einigen anderen Analysen die Phase nur auf Vielfache von 360 Grad bestimmt werden kann, ergeben sich oft scheinbare Phasensprünge, z.B. von -180 Grad auf +180 Grad. Eigentlich sollte aber die Phase unter -180 Grad fallen, z.B. auf -270 Grad.

Die Funktion entfernt alle Sprünge von mehr als 180 Grad durch Hinzufügen von Vielfachen von 360 Grad. Anhand der y-Einheit der Phase wird erkannt, ob im Gradmaß (360 Grad) oder Bogenmaß (2*PI) korrigiert wird.

Die Phase des ersten Messwertes wird allerdings ignoriert. Denn das ist bei Frequenz null der Gleich-Anteil (DC), dessen Phase 0 oder 180 Grad beträgt. D.h. der Sprung zwischen 1. und 2. Messwert bleibt unkorrigiert.

Beispiele:

```
Phase = PhaseContinuous ( Phase )
```

Phasenverlauf ist bereits in einem Datensatz. Die 360 Grad Sprünge werden entfernt.

```
frf = FrequencyResponse ( in, out, 1000, 0, 50, 0 )
frf = PhaseContinuous ( frf )
```

Hier wird die Übertragungsfunktion in ihrer Phase stetig gemacht.

Siehe auch:

PhaseMod, NorthCorrection, mod, FrequencyResponse

PhaseMod

Verfügbar ab: Professional Edition

Windrichtungen, Winkel, oder Phasen werden in einen üblichen Wertebereich transformiert, z.B. 0 .. 360 Grad

Deklaration:

```
PhaseMod ( Daten [, Bereich] ) -> Ergebnis
```

Parameter:

Daten	Eingangsdaten
Bereich	Bereich (optional , Standardwert: "360")
	"360" : 0..360 Grad
	"180" : -180..+180 Grad
	"2pi" : 0..2*pi
	"pi" : -pi..pi
Ergebnis	Ergebnis

Beschreibung:

Winkel werden durch passende Addition eines Vielfachen von 360° wieder in den gewünschten Zahlenbereich transformiert, ohne dass der Winkel dabei inhaltlich verändert wird.

Für den Bereich 0..360 Grad wird z.B. bei Winkeln über 360 Grad ein Wert von 360 subtrahiert, bei Winkeln unter 0 wird 360 addiert.

Die Funktion ersetzt eine geeignete modulo Rechnung

Wird die Phase eines komplexen Datensatzes korrigiert, wird dessen Komponente .P als Eingangsdatensatz angegeben.

Nur äquidistante Eingangsdaten werden unterstützt. Falls die Eingangsdaten eine Zeitspur haben, muss die Funktion auf .Y des Datensatzes angewendet werden.

Beispiele:

Phase nach einer Berechnung in den Bereich -180 bis +180 Grad transformieren

```
Phase = PhaseMod( Phase, "180" )
```

Berechnung der gleitend über je 10s gemittelten Windrichtung

```
NC = NorthCorrection( Channel, 10 )
NC_Mean = MvMean( NC, 10, 10 )
Wind = PhaseMod( NC_Mean, "360" )
```

Siehe auch:

NorthCorrection, mod, PhaseContinuous

PI

Kreiszahl PI = 3.1415...

Beispiele:

Die Konstante PI als Maß für einen Winkel in Bogenmaß (Radiant) entspricht 180 Grad:

```
Degree180 = PI * INDEGR
```

Erzeugung eines Datensatzes, der genau die erste Halbwelle der Sinusfunktion zeigt:

```
HalfWave = sin(Ramp(0, PI/100, 100))
```

Siehe auch:

PI2

PI2

2 * Kreiszahl PI = 6.2831...

Beispiele:

Die Konstante PI2 als Maß für einen Winkel in Bogenmaß (Radiant) entspricht 360 Grad:

```
Degree360 = PI2 * INDEGR
```

Erzeugung eines Datensatzes, der genau die erste Welle der Sinusfunktion zeigt:

```
wave = sin(Ramp(0, PI2/100, 100))
```

Siehe auch:

PI

PnClose

Das aktive Panel wird geschlossen.

Deklaration:

```
PnClose ( EwOption )
```

Parameter:

EwOption	Optionsparameter bzw. Rückgabewert eines Paneldialogs.

Beschreibung:

Das aktive Panel oder alle Panele werden geschlossen.

Die Bedeutung des Optionsparameters ist abhängig vom Kontext des Aufrufs:

Aufruf innerhalb einer Ereignissequenz eines Panel-Dialogs, der mit der Funktion Dialog() gestartet wurde:

Der Panel-Dialog wird geschlossen und der übergebene Parameter wird als Rückgabewert des Dialog()-Befehls verwendet. Die Funktion verhält sich so analog zur Funktion DlgCloseDialog() bei Anwenderdefinierten Dialogen.

Sonst:

Ein Wert von 0 bedeutet, dass das aktive Panel geschlossen wird. Eine 1 schließt alle geöffneten Panele.

Die Panele werden nicht gesichert, eventuelle Änderungen gehen verloren.

Diese Funktion wurde mit FAMOS V2022 eingeführt und ersetzt die Funktion DbClosePanel() älterer Versionen.

Beispiele:

Eine Panel-Datei wird geladen. Es werden diverse Aktualisierungen durchgeführt, danach wird das Panel gedruckt und wieder geschlossen.

```
err = PnLoad("d:\templates\result.panel")
IF err <> 0
   BoxMessage("Fehler", GetLastError(), "!1")
ELSE
   ; Verschiedene Aktualisierungen
   ; ...
   PnPrint(0)
   PnClose(0)
END
```

Ein Panel 'InputValue.panel' besteht u.a. aus einem Eingabefeld "input" zur Eingabe eines positiven Zahlenwertes sowie 2 Schaltflächen 'OK' und 'Abbrechen'. Der Dialog()-Befehl liefert den eingegebenen Wert zurück bzw. -1, falls abgebrochen werden soll.

Ereignis-Sequenz 'Knopf gedrückt' für den 'OK'-Knopf:

```
value = PnGetValue("input")
PnClose(value)
```

Ereignis-Sequenz 'Knopf gedrückt' für den 'Abbrechen'-Knopf

```
PnClose(-1)
```

Ereignis-Sequenz 'Schließen' (Anwender verwendet Systemmenü zum Schließen):

```
; Selbes Verhalten wie 'Abbrechen'-Knopf
PnClose(-1)
```

Aufruf des Dialogs:

```
value = Dialog("InputValue.panel", "", 0)
IF value < 0
   EXITSEQUENCE 0
END
;Weiter in der Sequenz...
...
```

Siehe auch:

PnLoad, DbShow, Dialog

Unterstützt ab:

Version 2022

PnDeleteItem

Anwendungsbereich: Panele

Löscht einen bzw. alle Einträge (Liste, Klappliste etc.)

Deklaration:

```
PnDeleteItem ( TxElementName, Index )
```

Parameter:

TxElementName	Name des zu ändernden Elements.
Index	Index des zu löschenden Eintrages. Der erste Eintrag hat den Index 1. Um alle Einträge zu löschen, geben Sie eine 0 an.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld

Beispiele:

In einem Listenfeld mit Mehrfachselektion werden alle selektierten Einträge gelöscht:

```
Count = PnGetItemCount("list1")
i = Count
WHILE i > 0
   IF PnIsItemSelected("list1", i)
      PnDeleteItem("list1", i)
   END
   i = i - 1
END
```

Siehe auch:

PnGetItemCount, PnGetItemText, PnSetItemText, PnInsertItem, PnFindItem

PnEnable

Anwendungsbereich: Panele

Das angegebene Element wird gesperrt (somit durch den Anwender nicht bedienbar) bzw. wieder freigegeben.

Deklaration:

```
PnEnable ( TxElementName, Auftrag )
```

Parameter:

TxElementName	Name des zu steuernden Panel-Elements.
Auftrag	Auftrag
	0 : Element sperren
	1 : Element freigeben

Beschreibung:

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

Eine Panel-Seite mit dem Namen 'Filter' enthält einen Knopf 'Execute', mit dem ein aktuell selektierter Kanal (1.Messung/1.Kanal) gefiltert werden soll. Wenn durch die aktuelle Selektion in Messungs- und Kanalliste kein solcher Kanal ausgewählt ist, wird der Knopf gesperrt.

Ereignis-Sequenz 'Datenselektion geändert''

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   PnEnable("Filter.Execute", 1)
ELSE
   PnEnable("Filter.Execute", 0)
END
```

Siehe auch:

PnShow

PnExportGraphics

Anwendungsbereich: Panele

Eine Seite des aktiven Panels wird in ein wählbares Grafik-Format exportiert.

Deklaration:

```
PnExportGraphics ( TxDateiName, Seitenwahl, Format, Auflösung, Farbanzahl, Null ) -> Erfolg
```

Parameter:

TxDateiName	Dateiname, unter dem die exportierte Datei zu speichern ist.
Seitenwahl	Auswahl der zu exportierenden Seite.
	-1 : Die aktive Seite wird exportiert.
	>=1 : Seitennummer, nur die angegebene Seite wird exportiert.
Format	Grafik-Format
	0 : Portable Networks Graphic-FileFormat (*.png)
	1 : JPEG-FileFormat (*.jpg)
	2 : Windows Bitmap (*.bmp)
	3 : Windows Enhanced Metafile (*.emf)
Auflösung	Gibt die zu wählende Auflösung bei den Bitmap-Formaten an. Die Einheit ist 'dpi' (Dots per Inch = Punkte pro Zoll). Übliche Werte sind z.B. 150dpi oder 300dpi. Der Wert muss im Bereich 72 - 1200dpi liegen. Für [Format] = 3 (Metadatei) ist dieser Wert auf 0 zu setzen.
Farbanzahl	Gibt den zu erzeugenden Farbtyp bei den Bitmap-Formaten an. Für [Format] = 3 (Metadatei) auf 0 zu setzen.
	0 : Nicht verwendet (EMF)
	1 : Echtfarben
	2 : 256 Farben
Null	Reserviert, immer auf 0 zu setzen
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Eine Seite des aktiven Panels wird unter dem angegebenen Dateinamem in das gewählte Grafik-Format exportiert.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird die Standarderweiterung für das gewählte Dateiformat verwendet.

Falls kein voller Pfadname angeben ist, wird bei aktivem Projekt das aktuelle Projektverzeichnis verwendet. Ansonsten wird das in den FAMOS-Voreinstellungen festgelegte Standardverzeichnis für Panel-Dateien verwendet.

Beispiele:

Eine Panelseite enthält einen Knopf 'Export'. Beim Drücken des Knopfes durch den Anwender wird in der zugehörigen Ereignis-Sequenz zunächst ein auf der selben Seite befindliches Label mit dem aktuellem Datum und Uhrzeit gefüllt und danach diese Seite im PNG-Format exportiert Auflösung 300dpi, 24 Mio Farben).

Ereignis-Sequenz 'Gedrückt' des 'Export'-Knopfes

```
TxDate = TimeToText( TimeSystem?(), 3)
PnSetText( "Date", TxDate)
PnExportGraphics("d:\exports\panel.png", -1, 0, 300, 1, 0)
```

Siehe auch:

PnPrint, PnExportPDF

PnExportPDF

Anwendungsbereich: Panele

Das aktive Panel wird als PDF-Dokument exportiert.

Deklaration:

```
PnExportPDF ( TxDateiname, Seitenwahl, Option [, Methode] ) -> Erfolg
```

Parameter:

TxDateiname	Name der zu erzeugenden Datei.
Seitenwahl	Auswahl der zu exportierenden Seite(n).
	-1 : Die aktive Seite wird exportiert.
	0 : Das gesamte Panel (alle Seiten) werden exportiert.
	>=1 : Seitennummer, nur die angegebene Seite wird exportiert.
Option	Optionsparameter
	0 : Wenn die Datei bereits existiert, wird diese überschrieben.
	1 : Wenn die Datei bereits existiert, werden die neuen Seiten angehängt.
Methode	Export-Methode (optional , Standardwert: 0)
	0 : Die globale Voreinstellung ("Optionen"/"Datei-Export"/"PDF") wird verwendet.
	1 : Automatische Auswahl des Verfahrens, so dass die resultierende Datei minimale Größe hat.
	2 : Bitmap: Die gesamte Seite wird als Bitmap gespeichert und in das PDF-Dokument eingebettet. Kompatibel zu FAMOS-Versionen <= 7.2
	3 : Vektorgrafik bevorzugt: Die einzelnen Grafikobjekte nach Möglichkeit einzeln im PDF-Dokument eingebettet. Liefert die beste Qualität.
	4 : Vektorgrafik bevorzugt. Alternative Export-Methode mit dem virtuellen Druckertreiber "Win2PDF". Kann in vielen Szenarien Geschwindigkeitsvorteile bringen und die Dateigröße der erzeugten PDF-Datei reduzieren. Der Treiber muss zuvor installiert worden sein (https://www.win2pdf.com). Kostenlose 30-Tage-Testversion verfügbar. Die Einstellung "Druckqualität" in "Systemsteuerung"->"Drucker und Scanner"->"Win2PDF" wird beachtet. 300 DPI sind normalerweise ein guter Kompromiss zwischen Dateigröße und Qualität.
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Das aktive Panel wird unter dem angegebenen Dateinamem in das PDF-Format exportiert.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird '.pdf' ergänzt.

Falls kein voller Pfadname angeben ist, wird bei aktivem Projekt das aktuelle Projektverzeichnis verwendet. Ansonsten wird das in den FAMOS-Voreinstellungen festgelegte Standardverzeichnis für Panel-Dateien verwendet.

Beispiele:

Eine Panelseite enthält einen Knopf 'Report generieren'. Beim Drücken des Knopfes durch den Anwender wird in der zugehörigen Ereignis-Sequenz zunächst ein auf der selben Seite befindliches Label mit dem aktuellem Datum und Uhrzeit gefüllt und danach alle Seiten des Panels in eine PDF-Datei exportiert.

Ereignis-Sequenz 'Gedrückt' des 'Report generieren'-Knopfes

```
TxDate = TimeToText( TimeSystem?(), 3)
PnSetText( "Date", TxDate)
PnExportPDF("d:\reports\report.pdf", 0, 0)
```

Siehe auch:

PnPrint, PnExportGraphics

PnFindItem

Anwendungsbereich: Panele

Sucht nach einem Eintrag (Liste, Klappliste etc.) mit dem angegebenen Inhalt.

Deklaration:

```
PnFindItem ( TxElementName, TxInhalt ) -> Index
```

Parameter:

TxElementName	Name des zu untersuchenden Elements.
TxInhalt	Text für den zu suchenden Eintrag.
Index	Index des gesuchten Eintrages (>=0), falls gefunden. 0 sonst.

Beschreibung:

Die Funktion unterscheidet nicht bezüglich Groß-/Kleinschreibung.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, CCV-Auswahl

Beispiele:

In einem Listenfeld (mit Einfachselektion) wird ein Eintrag gesucht. Falls der Eintrag vorhanden ist, wird dieser selektiert und ggf. in die Sicht gerollt.

```
i = PnFindItem("list", "100.0")
IF i > 0
   PnSelectItem("list", i)
END
```

Siehe auch:

PnGetItemCount, PnGetItemText, PnSetItemText, PnInsertItem, PnDeleteItem

PnGetActivePage

Anwendungsbereich: Panele

Die Seitennummert der gerade aktiven (sichtbaren) Seite im aktuellen Panel wird ermittelt.

Deklaration:

```
PnGetActivePage ( ) -> Seitennummer
```

Parameter:

Seitennummer	Aktuelle Seitennummer

Beschreibung:

Beispiele:

Ein Panel enthält mehrere Seiten mit einen Knopf '>>'. Bei Betätigung des Knopfes soll auf die jeweils nächste Seite weitergeschaltet werden.

```
Page = PnGetActivePage()
IF Page < PnGetPageCount()
   PnSetActivePage(Page +1)
END
```

Siehe auch:

PnSetActivePage

PnGetFileSelection

Anwendungsbereich: Panele

Die aktuelle Selektion in der Dateiliste eines Dateiexplorer-Elements wird abgefragt.

Deklaration:

```
PnGetFileSelection ( TxElementName, Rückgabe-Format ) -> Selektion
```

Parameter:

TxElementName	Name des abzufragenden Dateiexplorer-Elements.
Rückgabe-Format	Steuert, in welchem Format die selektierten Einträge zurückgegeben werden.
	0 : Kompletter Pfadname
	1 : Dateiname + Erweiterung
	2 : Anzeigename in der Dateiliste.
Selektion	Die aktuell selektierte Datei (Datentyp Text, bei Dateilisten mit Einfachselektion) bzw. die selektierten Dateien (Datentyp Textfeld, bei Listen mit Mehrfachselektion). Leerer Text bzw. Textfeld der Länge 0, wenn keine Selektion ermittelt werden kann.

Beschreibung:

Mit der Funktion wird die aktuelle Selektion in der Dateiliste abgefragt.

Verzeichnisnamen werden ohne schließendem '\' geliefert.

Sonderfall: Falls die Dateiliste nicht sichtbar ist (Modus "Nur Verzeichnisbaum") und die Baumanzeige für die Anzeige von Dateien konfiguriert ist, wird die Selektion im Baum abgefragt. Dies kann dann entweder ein Verzeichnis oder eine Datei sein.

Pfad-/Dateiname vs. Anzeigename: Der im Element angezeigte Name einer Datei oder eines Verzeichnisses kann vom echten Namen im Dateisystem abweichen. Dies ist beispielsweise bei Spezialordnern der Fall, bei denen der Anzeigename lokalisiert wird (z.B. Windows mit Anzeigesprache "deutsch", Anzeigename "c:\Programme", echter Pfad: "c:\Program files") oder bei Dateien, bei denen möglicherweise die Dateierweiterung weggelassen wird (abhängig von der Windows-Explorer-Einstellung "Dateierweiterung bei bekannten Datentypen ausblenden").

Beispiele:

Die aktuell selektierten Messwert-Dateien in einem Dateiexplorer werden geladen.

Widget mit Einzelselektion:

```
TxFileName = PnGetFileSelection( "FileExplorer1", 0)
FileLoad( TxFileName, "", 0)
```

Widget mit Multiselektion:

```
SelectedFiles = PnGetFileSelection( "FileExplorer1", 0)
FOREACH ELEMENT TxFileName in SelectedFiles
   FileLoad( TxFileName, "", 0)
END
```

Siehe auch:

PnGetFolder, PnSetFolder, PnSetFileSelection

PnGetFolder

Anwendungsbereich: Panele

Ermittelt das aktuelle Verzeichnis für den Dateiexplorer.

Deklaration:

```
PnGetFolder ( TxElementName ) -> Verzeichnis
```

Parameter:

TxElementName	Name des abzufragenden Dateiexplorer-Elements.
Verzeichnis	Aktuelles Verzeichnis.

Beschreibung:

Es wird immer der echte Pfad im Dateisystem zurückgegeben. Dieser kann vom angezeigten Namen abweichen, beispielsweise für Spezialordner, bei denen ein lokalisierter Name angezeigt wird (z.B. Windows mit Anzeigesprache "deutsch", Anzeigename "c:\Programme\", echter Pfad: "c:\Program files\").

Der Verzeichnisname wird mit einem abschließenden '\' geliefert.

Wenn ein Knoten selektiert ist, der keinem realen Ordner entspricht (z.B. "Dieser PC" oder "Netzwerkumgebung"), wird ein leerer Text zurückgeliefert.

Beispiele:

Das aktuell eingestellte Verzeichnis an einem Dateiexplorer-Widget wird als neues Standardverzeichnis für Messwert-Dateien verwendet.

```
TxFolder = PnGetFolder( "FileExplorer1")
SetOption("Dir.DataFiles", TxFolder)
```

Siehe auch:

PnSetFolder, PnGetFileSelection, PnSetFileSelection

PnGetItemCount

Anwendungsbereich: Panele

Ermittelt die Anzahl der Einträge in dem angegebenen Element (Liste, Klappliste etc.)

Deklaration:

```
PnGetItemCount ( TxElementName ) -> Anzahl
```

Parameter:

TxElementName	Name des abzufragenden Elementes.
Anzahl	Anzahl der Einträge.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, CCV-Auswahl, Optionsgruppe

Beispiele:

In einem Listenfeld mit Mehrfachselektion werden alle selektierten Einträge gelöscht:

```
Count = PnGetItemCount("list1")
i = Count
WHILE i > 0
   IF PnIsItemSelected("list1", i)
      PnDeleteItem("list1", i)
   END
   i = i - 1
END
```

Siehe auch:

PnGetItemText, PnSetItemText, PnInsertItem, PnFindItem, PnDeleteItem

PnGetItemText

Anwendungsbereich: Panele

Gibt den Text für einen Eintrag in einem Element (Listenfeld, Klappliste etc.) zurück.

Deklaration:

```
PnGetItemText ( TxElementName, Index ) -> TxInhalt
```

Parameter:

TxElementName	Name des abzufragenden Elements.
Index	Index des abzufragenden Eintrages. Der erste Eintrag hat den Index 1.
TxInhalt	Text für den angegebenen Eintrag.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, CCV-Auswahl, Optionsgruppe

Beispiele:

Aus einer Liste, welche Namen von Messwert-Dateien im FAMOS-Format enthält, wird der aktuell selektierte Eintrag ausgelesen und die entsprechende Datei geladen.

```
i = PnGetSelectedItem("listFiles")
IF i > 0
   FileName$ = PnGetItemText("listFiles", i)
   fh = FileOpenDSF( FileName$,0)
   IF fh > 0
      ;...
      FileClose(fh)
   END
END
```

Siehe auch:

PnGetItemCount, PnSetItemText, PnInsertItem, PnFindItem, PnDeleteItem

PnGetPageCount

Anwendungsbereich: Panele

Die Anzahl der Seiten im aktiven Panel wird ermittelt.

Deklaration:

```
PnGetPageCount ( ) -> Seitenzahl
```

Parameter:

Seitenzahl	Anzahl der Seiten

Beschreibung:

Beispiele:

Das aktuelle Panel enthält mehrere Seiten, die alle in der Fußzeile ein Label mit dem Namen 'Date' enthalten. Alle diese Felder werden mit dem aktuellen Datum aktualisiert.

```
Page = PnGetPageCount()
TxDate = TimeToText( TimeSystem?(), 3)
WHILE Page > 0
   PnSetActivePage(Page)
   PnSetText( "Date", TxDate)
   Page = Page-1
END
```

Siehe auch:

PnSetActivePage

PnGetPageIndex

Anwendungsbereich: Panele

Für das aktive Panel wird zu einem Seitennamen die zugehörige Seitennummer ermittelt.

Deklaration:

```
PnGetPageIndex ( Seitenname ) -> Seitennummer
```

Parameter:

Seitenname	Name (Titel) der abzufragenden Seite.
Seitennummer	Die Nummer (>=1) der entsprechenden Seite. 0, wenn keine Seite mit dem angegebenen Namen existiert.

Beschreibung:

Seitennamen im Panel sind sensitiv bzgl. Groß-/Kleinschreibung. Der Seitenname muss also in dieser Hinsicht exakt angegeben werden.

Beispiele:

Im aktiven Panel wird die Seite mit dem Namen "Final report" in eine PDF-Datei exportiert.

```
PnExportPDF("d:\reports\report.pdf", PnGetPageIndex("Final report"), 0)
```

Siehe auch:

PnGetPageName

Unterstützt ab:

Version 2022

PnGetPageName

Anwendungsbereich: Panele

Für das aktive Panel wird zu einer Seitennummer der zugehörige Seitenname ermittelt.

Deklaration:

```
PnGetPageName ( Seitennummer ) -> Seitenname
```

Parameter:

Seitennummer	Nummer der abzufragenden Seite (>=1).
Seitenname	Der Name (Titel) der Seite.

Beschreibung:

Beispiele:

Vor dem Drucken der aktiven Seite wird der Anwender gefragt, ob er die gewählte Seite tatsächlich ausdrucken möchte.

```
PageIndex = PnGetActivePage()
PageName = PnGetPageName(PageIndex)
ok = BoxMessage("Print", "Do you really want to print the page [" + PageName + "]?","?2")
IF ok = 1
   PnPrint(PageIndex)
END
```

Siehe auch:

PnGetPageIndex

Unterstützt ab:

Version 2022

PnGetPosition

Anwendungsbereich: Panele

Die Position eines Panel-Elements wird abgefragt.

Deklaration:

```
PnGetPosition ( TxElementName, Position ) -> Position
```

Parameter:

TxElementName	Name des abzufragenden Panel-Elements.
Position	Positionsparameter
	0 : X-Position der linken oberen Ecke
	1 : Y-Position der linken oberen Ecke
	2 : Breite
	3 : Höhe
Position	Aktuelle Position [in Millimeter] des Panel-Elements

Beschreibung:

Mit dieser Funktion kann die Position und die Größe des Panel-Elements auf der Panel-Seite abgefragt werden.

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

Eine Panel-Seite mit dem Namen 'Calculation' enthält ein mehrzeiliges Eingabefeld 'Status', das um 3.3 mm nach oben verschoben wird und dessen Höhe um 3.3 mm vergrößert wird. Die X-Position und die Breite bleiben unverändert.

```
X = PnGetPosition("Calculation.Status", 0))
Y = PnGetPosition("Calculation.Status", 1)
Height = PnGetPosition("Calculation.Status", 3)
PnSetPosition("Calculation.Status", X, Y - 3.3, 0, Height + 3.3)
```

Siehe auch:

PnSetPosition

PnGetSelectedItem

Anwendungsbereich: Panele

Ermittelt den aktuell selektierten Eintrag in einer Liste mit Einfachselektion.

Deklaration:

```
PnGetSelectedItem ( TxElementName ) -> Index
```

Parameter:

TxElementName	Name des zu untersuchenden Elements.
Index	Index des selektierten Eintrages (>=0). 0, wennn kein Eintrag selektiert ist.

Anwendbar auf:

Listenfeld (mit Einfachselektion), Klappliste, Kombinationsfeld, CCV-Auswahl, Optionsgruppe

Beispiele:

Aus einer Liste, welche Namen von Messwert-Dateien im FAMOS-Format enthält, wird der aktuell selektierte Eintrag ausgelesen und die entsprechende Datei geladen.

```
i = PnGetSelectedItem("listFiles")
IF i > 0
   FileName$ = PnGetItemText("listFiles", i)
   fh = FileOpenDSF( FileName$,0)
   IF fh > 0
      ;...
      FileClose(fh)
   END
END
```

Siehe auch:

PnSelectItem, PnIsItemSelected

PnGetSelectedItemCount

Anwendungsbereich: Panele

Ermittelt die Anzahl der selektierten Einträge in einer Liste mit Mehrfachselektion.

Deklaration:

```
PnGetSelectedItemCount ( TxElementName ) -> Anzahl
```

Parameter:

TxElementName	Name des zu untersuchenden Elements.
Anzahl	Anzahl der selektierten Einträge. 0, wennn kein Eintrag selektiert ist.

Beschreibung:

Anwendbar auf:

Listenfeld (Mehrfachselektion)

Beispiele:

Auf Knopfdruck sollen die selektierten Einträge in einem Listenfeldes mit Multiselektion ausgewertet werden. Der Knopf soll nur bedienbar sein, wenn mindestens 1 Eintrag selektiert ist. Dazu wird im Ereignis 'Ausgewählt' die Zahl der selektierten Einträge geprüft:

--> Ereignis-Sequenz 'Ausgewählt' des Listenfeldes

```
Count = PnGetSelectedItemCount(PA1)
PnEnable("Button1", Count > 0)
```

Siehe auch:

PnIsItemSelected, PnSetItemSelection, PnGetSelectedItem

PnGetText

Anwendungsbereich: Panele

Der Inhalt bzw. die Beschriftung des angegebenen Elements wird abgefragt.

Deklaration:

```
PnGetText ( TxElementName ) -> TxText
```

Parameter:

TxElementName	Name des abzufragenden Panel-Elements.
TxText	Aktueller Inhalt bzw. Beschriftung des Elements

Beschreibung:

Die Funktion kann nur auf solche Elemente angewendet werden, die eine Beschriftung besitzen oder deren aktueller Zustand durch einen Text ausgedrückt werden kann. Die Interpretation des Textes ist abhängig vom Typ des Elementes, z.B.:

Element	Bedeutung
Knopf	Beschriftung des Knopfes.
Label	Inhalt des Labels.
Eingabefeld (einzeilig)	Inhalt des Eingabefeldes.
Klappliste	Aktuell selektierter Eintrag.
Kombinationsfeld	Inhalt des Eingabefeldes.

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

In einem Panel gibt es ein Eingabefeld zur Eingabe eines Variablenkommentars sowie einen Knopf 'Übernehmen'. Bei Betätigung dieses Knopfes soll der aktuelle Inhalt des Eingabefeldes als Kommentar in diejenige Variable eingetragen werden, die aktuell in der Variablenliste (Messungen) als 1.Messung/1.Kanal ausgewählt ist.

Ereignis-Sequenz 'Gedrückt' des 'Übernehmen'-Knopfes

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   TxComment = PnGetText("Comment")
   SetComm(<TxVarName>, TxComment)
END
```

Siehe auch:

PnGetValue, PnSetText, PnSetActivePage

PnGetValue

Anwendungsbereich: Panele

Der aktuelle numerische Wert des angegebenen Elements wird abgefragt.

Deklaration:

```
PnGetValue ( TxElementName ) -> Wert
```

Parameter:

TxElementName	Name des abzufragenden Panel-Elements.
Wert	Aktueller Wert des Panel-Elements

Beschreibung:

Die Funktion kann nur auf solche Panel-Elemente angewendet werden, deren aktueller Zustand durch eine Zahl ausgedrückt werden kann. Die Interpretation des Wertes ist abhängig vom Typ des Elementes.

Element	Bedeutung
Eingabefelder	Falls der Text im Eingabefeld in eine Zahl konvertierbar ist, wird diese zurückgegeben. Ansonsten 0.
Schalter	Liefert 1, falls der Schalter gedrückt bzw. 'angekreuzt' ist, ansonsten 0.
Optionsgruppe	Liefert den Index der selektierten Option.
Bereichswahl, Zeitspanne	Liefert die Untergrenze des eingestellten Bereichs.

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

Ein Panel hat eine Seite 'Filter', die zur Einstellung von Filterparametern für einen Tiefpass dient. Zwei Eingabefelder 'Order' und 'Input_Freq' dienen zur Vorgabe von Filterordnung und Grenzfrequenz. Ein Schalter 'Bessel' gibt vor, ob mit Butterworth- oder Bessel-Charakteristik (Schalter ein) gerechnet werden soll. Beim Laden des Panels werden die Felder mit Standardwerten initialisiert. Auf Knopfdruck wird die Filterung dann für die aktuelle Selektion in der Variablenliste (Messungen), hier: 1.Kanal/1.Messung, ausgeführt.

Ereignis-Sequenz 'Panel Initialisierung'

```
PnSetValue("Filter.Order", 4)
PnSetValue("Filter.Input_Freq", 100)
PnSetValue("Filter.Bessel", 1)
```

Ereignis-Sequenz 'Gedrückt' des 'Filtern!'-Knopfes

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   Order = PnGetValue( "Order")
   Freq  = PnGetValue("Input_Freq")
   Type  = PnGetValue("Bessel")
   IF Order > 1 AND Freq > 0
      IF Type = 1  ; Bessel
         filtrat = FiltLP( <TxVarName>, 1, 0, Order, Freq)
      ELSE         ; Butterworth
         filtrat = FiltLP( <TxVarName>, 0, 0, Order, Freq)
      END
   END
END
```

Siehe auch:

PnSetValue, PnGetText

PnGetValue2

Anwendungsbereich: Panele

Der aktuelle 2. numerische Wert des angegebenen Elements wird abgefragt.

Deklaration:

```
PnGetValue2 ( TxElementName ) -> Wert
```

Parameter:

TxElementName	Name des abzufragenden Panel-Elements.
Wert	Aktueller 2. Wert des Panel-Elements

Beschreibung:

Die Funktion kann nur auf solche Panel-Elemente angewendet werden, deren aktueller Zustand durch 2 Zahlen ausgedrückt werden kann.

Element	Bedeutung
Zeitspanne	Liefert die obere Grenze des selektierten Bereichs als Zeitwert im imc FAMOS-Zeitformat.
Bereichswahl	Liefert die obere Grenze des selektierten Bereichs.

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Zeitspanne, Bereichswahl

Beispiele:

Ein Panel enthält ein Zeitspannen-Widget "dateRange" und ein Listenfeld "list". Das Listenfeld listet alle Dateien aus einem gegebenen Verzeichnis auf, deren Dateizeit in dem durch "dateRange" festgelegten Bereich liegt. Ein Knopf "Aktualiseren!" bewirkt das Auslesen des aktuell eingestellten Datumbereichs und das Aktualisieren der Dateiliste.

Ereignis-Sequenz 'Gedrückt' des 'Aktualisieren!'-Knopfes

```
t_min = PnGetValue("dateRange")
t_max = PnGetValue2("dateRange")

PnDeleteItem("list", 0)
fileList = FsGetFileNames("c:\data", "*.*", 0, 0, 1)
FOREACH ELEMENT path in fileList
   t_file = FsGetFileTime(path)
   IF t_file >= t_min AND t_file <= t_max
      PnInsertItem("list", 0, FsSplitPath(path, 4), 0)
   END
END
```

Siehe auch:

PnGetValue, PnSetValue, PnSetValue2

PnInsertItem

Anwendungsbereich: Panele

Fügt einem Element (Listenfeld, Klappliste etc.) neue Einträge hinzu.

Deklaration:

```
PnInsertItem ( TxElementName, Index, TxInhalt, Option )
```

Parameter:

TxElementName	Name des zu ändernden Elements.
Index	Einfügeposition. Der erste Eintrag hat den Index 1. Um den Eintrag am Ende anzuhängen, geben Sie eine 0 an.
TxInhalt	Neuer Eintrag bzw. Einträge. Erlaubt sind die Datentypen Text und Textfeld.
Option	Optionsparameter
	0 : Standard.
	1 : Der neue Eintrag wird gegebenenfalls ins Bild gerollt.

Beschreibung:

Wenn als 3. Parameter ein Textfeld angegeben wird, werden alle Elemente des Textfeldes nacheinander eingefügt.

Wenn die Liste sortiert ist, wird die angebene Einfügeposition ignoriert und der neue Eintrag entsprechend der gewählten Sortierung eingeordnet.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld

Beispiele:

Ein Listenfeld wird mit den Namen aller Dateien gefüllt, die sich in einem vorgegebenen Verzeichnis befinden.

```
Dir$ = FsDlgSelectDirectory("Verzeichnis auswählen", "", 0)
FileListID = FsFileListNew( Dir$, "*.*", 0, 0, 0)
FileCount = FsFileListGetCount(FileListID)
i = 1
WHILE i <= FileCount
   File$ = FsSplitPath(FsFileListGetName(FileListID, i), 4)
   PnInsertItem("listFiles", i, File$, 0)
   i = i + 1
END
FsFileListClose(FileListID)
```

Selbe Aufgabe, aber effektiver durch Verwendung einer Textfeld-Variablen:

```
Dir$ = FsDlgSelectDirectory("Verzeichnis auswählen", "", 0)
Files$ = FsGetFileNames( Dir$,"*.*", 0, 0, 0)
Files$ = TxRegExMatch( Files$, "^(.+)\\([^/]+)$", "", 0, 2)
PnInsertItem("listFiles", 0, Files$, 0)
```

Ein Listenfeld wird mit den Namen aller Variablen gefüllt, die sich beim Start des Pogramms in der FAMOS-Variablenliste befinden. Anschließend wird der erste Eintrag selektiert.

```
Count = VarGetInit(0)
i = 1
WHILE i <= Count
   PnInsertItem("ListVariables", i, VarGetName?(i), 0)
   i = i + 1
END
PnSelectItem("ListVariables", 1)
```

Ein Listenfeld wird mit allen Werten eines (kurzen) Datensatzes gefüllt.

```
Count = leng?(MyData)
i = 1
WHILE i <= Count
   TxVal = TForm( MyData[i], "")
   PnInsertItem("list1", 0, TxVal, 0)
   i = i + 1
END
```

Siehe auch:

PnGetItemCount, PnGetItemText, PnSetItemText, PnFindItem, PnDeleteItem

PnInsertPage

Anwendungsbereich: Panele

In das aktive Panel wird eine neue Seite eingefügt.

Deklaration:

```
PnInsertPage ( TxVorlagenDatei, SeitenNummer, Seitenname, EinfügePosition ) -> Erfolg
```

Parameter:

TxVorlagenDatei	Dateiname der Paneldatei, die die Vorlage für die neu einzufügende Seite enthält. Wenn der Parameter leer ist, wird das aktuelle Panel verwendet.
SeitenNummer	Legt fest, welche Seite aus [TxVorlagenDatei] zu verwenden ist.
Seitenname	Der Name der neuen Seite. Wenn bereits eine Seite mit dem selben Namen existiert, liefert die Funktion eine Fehlermeldung. Wenn ein leerer Text angegeben wird, wird der Name der neuen Seite automatisch generiert.
EinfügePosition	Die neue Seite wird an der hier angegebenen Position eingefügt. Eine 0 bedeutet, dass am Ende angehängt wird.
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Mit dieser Funktion kann dem aktuellen Panel eine neue Seite zugefügt werden. Die neue Seite kann entweder ein Duplikat einer Seite des aktuelles Panels sein oder aus einer anderen Paneldatei importiert werden.

Wenn bei dem Dateinamen der Vorlagendatei kein kompletter Pfad angegeben ist, wird die Datei nacheinander in den folgenden Verzeichnissen gesucht:

- Projektverzeichnis: Wenn ein Projekt aktiv ist, wird zunächst im aktuellen Projektverzeichnis gesucht.
- Standardverzeichnis für Panel-Dateien: FAMOS-Voreinstellung für Panele/Dialoge/Sequenzen.

Beispiele:

Der Datensatz "u0" soll in Form einer 2-spaltigen Tabelle (x,y) dokumentiert werden. Die Paneldatei 'table_template.panel' dient als Seiten-Vorlage, diese enthält u.a. ein Tabellenobjekt mit 50 Zeilen. Entsprechend der Länge des Datensatzes wird die benötigte Anzahl von Seiten generiert und abschließend ausgedruckt.

```
Length = leng?(u0)
Rows = 50
FirstSample = 1
WHILE FirstSample <= Length
   IF FirstSample = 1
      ; Vorlage öffnen
      PnLoad("table_template")
   ELSE
      ; neue Seite anhängen
      PnInsertPage("table_template", 1, "", 0)
   END
   ; y-Daten für Tabelle ausschneiden
   y = CutIndex( u0, FirstSample, FirstSample+Rows-1)
   PnTableSetColumn( "table1", 2, 1, y)
   ; x-Daten für Tabelle konstruieren
   x = Ramp( (FirstSample-1)*xdel?(u0)+ xoff?(u0), xdel?(u0), leng?(y))
   PnTableSetColumn( "table1", 1, 1, x)
   FirstSample = FirstSample+Rows
END
PnPrint(0)
PnClose(0)
```

Siehe auch:

PnLoad, PnSave

PnIsItemSelected

Anwendungsbereich: Panele

Ermittelt, ob ein Eintrag in einer Liste mit Mehrfachselektion selektiert ist.

Deklaration:

```
PnIsItemSelected ( TxElementName, Index ) -> IstSelektiert
```

Parameter:

TxElementName	Name des zu untersuchenden Elements.
Index	Index des zu prüfenden Eintrages. Der erste Eintrag hat den Index 1.
IstSelektiert	Selektionszustand des Eintrages
	0 : Nicht selektiert
	1 : Selektiert

Anwendbar auf:

Listenfeld(Mehrfachselektion)

Beispiele:

Ein Listenfeld in einem Panel wird mit allen Werten eines (kurzen) Datensatzes gefüllt. Auf Knopdfruck werden alle selektierten Werte auf 0 gesetzt.

--> Ereignis-Sequenz 'Panel Initialisierung'

```
;Füllen des Listenfeldes mit den Werten des Datensatzes:
Count = leng?(MyData)
i = 1
WHILE i <= Count
   TxVal = TForm( MyData[i], "")
   PnInsertItem("list1", 0, TxVal, 0)
   i = i + 1
END
```

--> Ereignis-Sequenz 'Knopf gedrückt'

```
;Die selektierten Samples werden auf 0 gesetzt.
Count = PnGetItemCount("list1")
i = 1
WHILE i <= Count
   IF PnIsItemSelected("list1", i)
      MyData[i] = 0
      PnSetItemText("list1", 1, "0")
   END
   i = i + 1
END
```

Siehe auch:

PnGetSelectedItemCount, PnSetItemSelection, PnGetSelectedItem

PnLoad

Eine Panel-Datei wird geladen und angezeigt.

Deklaration:

```
PnLoad ( TxDateiname ) -> Erfolg
```

Parameter:

TxDateiname	Name der zu öffnenden Panel-Datei.
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die angegebene Panel-Datei wird geladen.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird ".panel" angenommen.

Wenn bei dem Dateinamen kein kompletter Pfad angegeben ist, wird die Datei nacheinander in den folgenden Verzeichnissen gesucht:

- Projektverzeichnis: Wenn ein Projekt aktiv ist, wird zunächst im aktuellen Projektverzeichnis gesucht.
- Standardverzeichnis für Panel-Dateien: FAMOS-Voreinstellung für Panele/Dialoge/Sequenzen.

Um ein Panel im Dialog-Modus zu starten, verwenden Sie den Befehl Dialog().

Diese Funktion wurde mit FAMOS V2022 eingeführt und ersetzt die Funktion DbLoadPanel() älterer Versionen.

Multithreading: Alle Funktionen zur Panel-Fernsteuerung dürfen überall aufgerufen werden und wirken global. Das hier geladene Panel ist also in allen Ausführungs-Threads verwendbar.

Beispiele:

Eine Panel-Datei wird geladen. Es werden diverse Aktualisierungen durchgeführt, danach wird das Panel gedruckt und wieder geschlossen.

```
err = PnLoad("d:\templates\result.panel")
IF err <> 0
   BoxMessage("Fehler", GetLastError(), "!1")
ELSE
   ; Verschiedene Aktualisierungen
   ; ...
   PnPrint(0)
   PnClose(0)
END
```

Siehe auch:

PnClose, Dialog

Unterstützt ab:

Version 2022

PnPrint

Anwendungsbereich: Panele

Das aktive Panel wird gedruckt.

Deklaration:

```
PnPrint ( Seitenwahl )
```

Parameter:

Seitenwahl	Auswahl der zu druckenden Seite(n).
	-1 : Die aktive Seite wird gedruckt.
	0 : Das gesamte Panel (alle Seiten) wird gedruckt.
	>=1 : Seitennummer, nur die angegebene Seite wird gedruckt.

Beschreibung:

Das aktive Panel wird gedruckt. Die aktuellen Druckeinstellungen ("Datei"/"Drucken") werden berücksichtigt.

Beispiele:

Eine Panelseite enthält einen Knopf 'Drucken'. Beim Drücken des Knopfes durch den Anwender wird in der zugehörigen Ereignis-Sequenz zunächst ein auf der selben Seite befindliches Label mit dem aktuellem Datum und Uhrzeit gefüllt und danach die Seite gedruckt.

-> Ereignis-Sequenz 'Gedrückt' des 'Drucken'-Knopfes

```
TxDate = TimeToText( TimeSystem?(), 3)
PnSetText( "Date", "Printed: " + TxDate)
PnPrint(-1)
```

Siehe auch:

PnExportPDF, PnExportGraphics

PnRemovePage

Anwendungsbereich: Panele

Im aktiven Panel wird eine Seite gelöscht.

Deklaration:

```
PnRemovePage ( SeitennummerOderTitel ) -> Erfolg
```

Parameter:

SeitennummerOderTitel	Gibt die Seitennummer (1..) oder den Titel der zu löschenden Seite an.
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Mit dieser Funktion wird eine Seite im aktiven Panel gelöscht. Die zu löschende Seite kann über ihren Namen oder über den Index angegeben werden.

Beispiele:

Löscht die Seite mit dem Namen "Intro":

```
PnRemovePage("Intro")
```

Ein Panel enthält 10 Seiten zur Anzeige von maximal 10 Kanälen, die aus einer Datei geladen werden.

Anhängig von der tatsächlich vorhandenen Kanalzahl werden die überflüssigen Seiten gelöscht:

```
fh = FileOpenDSF(MyFileName, 0)
channelCount = FileObjNum?(fh)
...
FOR i = 10 TO channelCount+1 STEP -1
   PnRemovePage(i)
END
```

Siehe auch:

PnShowPage

PnSave

Anwendungsbereich: Panele

Das aktive Panel wird gespeichert.

Deklaration:

```
PnSave ( TxDateiName ) -> Erfolg
```

Parameter:

TxDateiName	Dateiname, unter dem das Panel zu speichern ist.
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Das aktive Panel wird unter dem angegebenen Dateinamem gespeichert.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird '.panel' ergänzt.

Falls kein voller Pfadname angeben ist, wird bei aktivem Projekt das aktuelle Projektverzeichnis verwendet. Ansonsten wird das in den FAMOS-Voreinstellungen festgelegte Standardverzeichnis für Panel-Dateien verwendet.

Beispiele:

Der Datensatz "u0" soll in Form einer 2-spaltigen Tabelle (x,y) dokumentiert werden. Die Paneldatei 'table_template.panel' dient als Seiten-Vorlage, diese enthält u.a. ein Tabellenobjekt mit 50 Zeilen. Entsprechend der Länge des Datensatzes wird die benötigte Anzahl von Seiten generiert und abschließend unter neuem Namen gespeichert.

```
Length = leng?(u0)
Rows = 50
FirstSample = 1
WHILE FirstSample <= Length
   IF FirstSample = 1
      ; Vorlage öffnen
      PnLoad("table_template")
   ELSE
      ; neue Seite anhängen
      PnInsertPage("table_template", 1, "", 0)
   END
   ; y-Daten für Tabelle ausschneiden
   y = CutIndex( u0, FirstSample, FirstSample+Rows-1)
   PnTableSetColumn( "table1", 2, 1, y)
   ; x-Daten für Tabelle konstruieren
   x = Ramp( (FirstSample-1)*xdel?(u0)+ xoff?(u0), xdel?(u0), leng?(y))
   PnTableSetColumn( "table1", 1, 1, x)
   FirstSample = FirstSample+Rows
END
TxDateiName = BoxText?("Dateiname eingeben", "", 0)
PnSave(TxDateiName)
PnClose(0)
```

Siehe auch:

PnLoad, PnExportPDF

PnSelectItem

Anwendungsbereich: Panele

Selektiert einen Eintrag in einer Liste mit Einfachselektion

Deklaration:

```
PnSelectItem ( TxElementName, Index )
```

Parameter:

TxElementName	Name des anzusprechenden Elements.
Index	Index des zu selektierenden Eintrages. Der erste Eintrag hat den Index 1.

Anwendbar auf:

Listenfeld mit Einfachselektion, Klappliste, Kombinationsfeld, CCV-Auswahl, Optionsgruppe

Beispiele:

In einem Listenfeld (mit Einfachselektion) wird ein Eintrag gesucht. Falls der Eintrag vorhanden ist, wird dieser selektiert und ggf. in die Sicht gerollt.

```
i = PnFindItem("list", "100.0")
IF i > 0
   PnSelectItem("list", i)
END
```

Siehe auch:

PnGetSelectedItem, PnSetItemSelection

PnSetActivePage

Anwendungsbereich: Panele

Die aktive (sichtbare) Seite wird festgelegt.

Deklaration:

```
PnSetActivePage ( Seitennummer )
```

Parameter:

Seitennummer	Gibt die Seitennummer (1..) der zu aktivierenden Seite an.

Beschreibung:

Multithreading: Die Funktionen zur Panel-Fernsteuerung dürfen überall aufgerufen werden und wirken global. Die hier selektierte Seite ist also für alle Ausführungs-Threads gültig.

Beispiele:

Das aktuelle Panel enthält mehrere Seiten, die alle in der Fußzeile ein Label mit dem Namen 'Date' enthalten. Alle diese Felder werden mit dem aktuellen Datum aktualisiert.

```
Page = PnGetPageCount()
TxDate = TimeToText( TimeSystem?(), 3)
WHILE Page > 0
   PnSetActivePage(Page)
   PnSetText( "Date", TxDate)
   Page = Page-1
END
```

Siehe auch:

PnGetPageCount

PnSetFileSelection

Anwendungsbereich: Panele

Die Selektion in der Dateiliste eines Dateiexplorers wird gesetzt.

Deklaration:

```
PnSetFileSelection ( TxElementName, TxNeueSelektion, Format ) -> Erfolg
```

Parameter:

TxElementName	Name des zu ändernden Dateiexplorer-Elements.
TxNeueSelektion	Zu selektierende Dateinamen. Datentyp Text oder Textfeld (wenn das Element Mehrfachselektion unterstützt). Leerer Text, um die Selektion zu löschen.
Format	Gibt an, in welchem Format die neue Selektion angegeben wird.
	0 : Pfadname. Es kann sowohl der komplette Pfad als auch nur Dateiname + Erweiterung angegeben werden.
	2 : Anzeigename in der Dateiliste.
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Mit der Funktion wird die aktuelle Selektion in der Dateiliste eingestellt. Es können Dateien und Unterverzeichnisse selektiert werden. Diese müssen im aktuellen Verzeichnis vorhanden und aktuell sichtbar sein, d.h. ein Wechsel des aktuellen Verzeichnisses findet nicht statt. Wenn die Dateiliste für Multiselektion konfiguriert ist, kann auch ein Textfeld mit mehreren zu selektierenden Dateien übergeben werden.

Sonderfall: Falls die Dateiliste nicht sichtbar ist (Modus "Nur Verzeichnisbaum") und die Baumanzeige für die Anzeige von Dateien konfiguriert ist, wird die Selektion im Baum ausgeführt. In diesem Fall ist nur die Option 0 (kompletter Dateiname) zulässig. Wenn es sich um einen gültigen Dateinamen handelt, wird der Baum entsprechend aufgeklappt und die Datei selektiert, es findet also ggf. ein Wechsel des aktuellen Verzeichnisses statt.

Pfad-/Dateiname vs. Anzeigename: Der im Element angezeigte Name einer Datei oder eines Verzeichnisses kann vom echten Namen im Dateisystem abweichen. Dies ist beispielsweise bei Spezialordnern der Fall, bei denen der Anzeigename lokalisiert wird (z.B. Windows mit Anzeigesprache "deutsch", Anzeigename "c:\Programme", echter Pfad: "c:\Program files") oder bei Dateien, bei denen möglicherweise die Dateierweiterung weggelassen wird (abhängig von der Windows-Explorer-Einstellung "Dateierweiterung bei bekannten Datentypen ausblenden").

Beispiele:

Das aktuelle Verzeichnis für einen Dateiexplorer wird gesetzt. Alle Dateien mit der Erweiterung "raw" werden selektiert:

```
DatFolder = "c:\Experiment001\data"
AllDatFiles = FsGetFileNames( DatFolder, "*.raw", 0, 0, 0)
PnSetFolder( "FileExplorer1", DatFolder)
PnSetFileSelection( "FileExplorer1", AllDatFiles, 0)
```

Siehe auch:

PnGetFolder, PnSetFolder, PnSetFileSelection

PnSetFolder

Anwendungsbereich: Panele

Setzt das aktuelle Verzeichnis für den Dateiexplorer auf den angebenen Pfad.

Deklaration:

```
PnSetFolder ( TxElementName, TxNeuesVerzeichnis ) -> Erfolg
```

Parameter:

TxElementName	Name des zu ändernden Dateiexplorer-Elements.
TxNeuesVerzeichnis	Voller Pfadname des gewünschten Verzeichnisses.
Erfolg	Erfolg der Funktion. 0 wenn die Funktion erfolgreich ausgeführt werden konnte. -1 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die Funktion setzt das aktuelle Verzeichnis für den Dateiexplorer auf den angegeben Pfad.

Es muss immer der echte Pfad im Dateisystem angegeben werden. Dieser kann vom angezeigten Namen abweichen, beispielsweise für Spezialordner, bei denen ein lokalisierter Name angezeigt wird (z.B. Windows mit Anzeigesprache "deutsch", Anzeigename "c:\Programme", echter Pfad: "c:\Program files").

Hinweis:Um das Wurzelverzeichnis (also das Startverzeichnis für die angezeigte Verzeichnishierarchie) einzustellen, verwenden Sie die Funktion PnSetProperty(.., "Rootfolder", ..).

Beispiele:

Das aktuelle Verzeichnis für ein Dateiexplorer-Widget wird auf den voreingestellten Standardpfad für Messwert-Dateien gesetzt:

```
TxFolder = GetOption("Dir.DataFiles")
PnSetFolder( "FileExplorer1", TxFolder)
```

Siehe auch:

PnGetFolder, PnGetFileSelection, PnSetFileSelection, PnSetProperty

PnSetItemSelection

Anwendungsbereich: Panele

Setzt oder entfernt die Selektion für einen Eintrag in einer Liste mit Mehrfachselektion.

Deklaration:

```
PnSetItemSelection ( TxElementName, Index, AnAus )
```

Parameter:

TxElementName	Name des zu untersuchenden Elements.
Index	Index des zu ändernden Eintrages. Der erste Eintrag hat den Index 1. Um alle Einträge zu (de-)selektieren, geben Sie eine 0 an.
AnAus	Selektion an/aus
	0 : Selektion entfernen
	1 : Eintrag selektieren

Anwendbar auf:

Listenfeld (Mehrfachselektion)

Beispiele:

Die aktuelle Selektion in einer Liste mit Mehrfachselektion wird invertiert:

```
Count = PnGetItemCount("list1")
i = 1
   WHILE i <= Count
   Sel = PnIsItemSelected("list1", i)
   PnSetItemSelection("list1", i, NOT(Sel))
   i = i + 1
END
```

Siehe auch:

PnGetSelectedItemCount, PnIsItemSelected, PnSelectItem

PnSetItemText

Anwendungsbereich: Panele

Ersetzt einen Eintrag (Listenfeld, Klappliste etc.) mit den angegebenen Text.

Deklaration:

```
PnSetItemText ( TxElementName, Index, TxNeuerInhalt )
```

Parameter:

TxElementName	Name des zu ändernden Elements.
Index	Index des zu ändernden Eintrages. Der erste Eintrag hat den Index 1.
TxNeuerInhalt	Neuer Text für den zu ersetzenden Eintrag.

Anwendbar auf:

Listenfeld, Klappliste, Kombinationsfeld, Optionsgruppe

Beispiele:

Ein Listenfeld in einem Panel wird mit allen Werten eines (kurzen) Datensatzes gefüllt. Auf Knopdfruck werden alle selektierten Werte auf 0 gesetzt.

--> Ereignis-Sequenz 'Panel Initialisierung'

```
;Füllen des Listenfeldes mit den Werten des Datensatzes:
Count = leng?(MyData)
i = 1
WHILE i <= Count
   TxVal = TForm( MyData[i], "")
   PnInsertItem("list1", 0, TxVal, 0)
   i = i + 1
END
```

--> Ereignis-Sequenz 'Knopf gedrückt'

```
;Die selektierten Samples werden auf 0 gesetzt.
Count = PnGetItemCount("list1")
i = 1
WHILE i <= Count
   IF PnIsItemSelected("list1", i)
      MyData[i] = 0
      PnSetItemText("list1", i, "0")
   END
   i = i + 1
END
```

Siehe auch:

PnGetItemCount, PnGetItemText, PnInsertItem, PnFindItem, PnDeleteItem

PnSetPosition

Anwendungsbereich: Panele

Setzt die Position und die Größe eines Panel-Elements.

Deklaration:

```
PnSetPosition ( TxElementName, links, oben, breit, hoch )
```

Parameter:

TxElementName	Name des zu steuernden Panel-Elements.
links	X-Position der linken oberen Ecke.
oben	Y-Position der linken oberen Ecke.
breit	Breite
hoch	Höhe

Beschreibung:

Mit dieser Funktion kann die Position und die Größe des Panel-Elements auf der Panel-Seite gesteuert werden. Die Angabe dieser Werte erfolgt in Millimetern.

Wenn [breit] oder [hoch] 0 sind, bleibt die aktuelle Breite bzw. Höhe erhalten, nur die Lage wird entsprechend korrigiert.

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

Eine Panel-Seite mit dem Namen 'Calculation' enthält ein mehrzeiliges Eingabefeld 'Status', das um 3.3 mm nach oben verschoben wird und dessen Höhe um 3.3 mm vergrößert wird. Die X-Position und die Breite bleiben unverändert.

```
X = PnGetPosition("Calculation.Status", 0))
Y = PnGetPosition("Calculation.Status", 1)
Height = PnGetPosition("Calculation.Status", 3)
PnSetPosition("Calculation.Status", X, Y - 3.3, 0, Height + 3.3)
```

Siehe auch:

PnGetPosition, PnShow

PnSetProperty

Anwendungsbereich: Panele

Eine Eigenschaft des gewählten Elements wird neu gesetzt.

Deklaration:

```
PnSetProperty ( TxElementName, TxPropID, Wert )
```

Parameter:

TxElementName	Name des Panel-Elements.
TxPropID	Kennung der zu setzenden Eigenschaft.
	"FillColor" : Hintergrundfarbe
	"FrameColor" : Rahmenfarbe
	"TextColor" : Textfarbe
	"Image" : Bilddatei
	"RootFolder" : Wurzelverzeichnis
	"FileFilter" : Datei-Filter
	"Minimum" : Minimum
	"Maximum" : Maximum
	"Increment" : Schrittweite
	"Resolution" : Auflösung
	"SortOrder" : Sortierung
	"SortColumn" : Sortierspalte
Wert	Zahlenwert oder Text, auf den die Eigenschaft gesetzt werden soll.

Beschreibung:

Die zu setzende Eigenschaft wird durch vordefinierte Bezeichner ausgewählt. Die änderbaren Eigenschaften sind abhängig vom Typ des PanelPanel-Elements.

Die folgenden Eigenschaften sind bisher definiert:

Kennung	Bedeutung	Anwendbar auf:
"FillColor"	Hintergrundfarbe (RGB-Wert)	Alle Elemente mit Hintergrundfarbe
"FrameColor"	Rahmenfarbe (RGB-Wert)	Alle Elemente mit farbigem Rahmen
"TextColor"	Textfarbe (RGB-Wert)	Elemente mit Textanzeige
"Image"	Bilddatei	'Bild'-Element
"RootFolder"	Wurzelverzeichnis	'DateiExplorer'-Element
"FileFilter"	Datei-Filter	'DateiExplorer'-Element (z.B. "*.raw" oder "*.raw;*.dat")
"Minimum"	Minimum des einzustellenden Bereichs	'Schieberegler'-Element, 'Spin', 'Bereichswahl', 'Zeitspanne'
"Maximum"	Maximum des einzustellenden Bereichs	'Schieberegler'-Element, 'Spin', 'Bereichswahl', 'Zeitspanne'
"Increment"	Schrittweite bei Wertänderungen	'Spin'-Element
"Resolution"	Auflösung der Werte- bzw. Zeit-Achse	'Bereichswahl'-Element, 'Zeitspanne' (0: Sekunde, 1: Minute, 2: Stunde, 3: Tag, 4: Woche, 5: Monat, 6: Jahr)
"SortOrder"	Sortierung	'DateiExplorer'-Element (1: aufsteigend, 2: absteigend)
"SortColumn"	Index der zu sortierenden Spalte	'DateiExplorer'-Element

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

Eine Panelseite enthält ein Label 'Max' zur Anzeige des Maximums der gerade selektierten Variablen (1.Messung, 1.Kanal). Bei jeder Änderung der Selektion in Messungs- oder Kanalliste wird das Feld entsprechend aktualisiert. Wenn das Maximum einen bestimmten Grenzwert überschreitet, wird die Hintergrundfarbe des Feldes auf Rot geändert.

Ereignis-Sequenz 'Datenselektion geändert"

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   maxval = Max(<TxVarName>)
   PnSetValue("Page1.Max", maxval )
   IF maxval > 10
      ; Grenzwert überschritten (roter Hintergrund)
      PnSetProperty("Page1.Max", "FillColor", RGB( 255, 0, 0)))
   ELSE
      ; Standard (weißer Hintergrund)
      PnSetProperty("Page1.Max", "FillColor", RGB( 255, 255, 255)))
   END
END
```

An einem Versuchsaufbau wurden mehrere Messungen durchgeführt, deren Ergebnisse in separaten Verzeichnissen auf der Festplatte abgelegt worden sind. Zusätzlich enthält jedes Verzeichnis ein Foto des Aufbaus zum Zeitpunkt der Messung unter dem Namen "test.jpg". Die Daten werden nun nachträglich in einem einseitigen Panel visualisiert, welches zusätzlich zu einem Kurvenfenster auch ein Bildobjekt 'pic' enthält. Dieses soll zum aktuell angezeigten Kanal (1.Kanal/1.Messung) das zugehörige Foto anzeigen.

Ereignis-Sequenz 'Datenselektion geändert"

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   ; Name der zugehörigen Datei
   FileName = FileName?(<TxVarName>)
   IF FileName <> ""
      ; Dateinamen für Foto-Datei zusammenbauen
      PicFileName = FsGetParentDirectoryName(FileName)+ "\test.jpg"
      PnSetProperty( "pic", "Image", PicFileName)
   END
END
```

Siehe auch:

PnGetValue, PnGetText

PnSetText

Anwendungsbereich: Panele

Der Inhalt bzw. die Beschriftung des angegebenen Elements wird neu gesetzt.

Deklaration:

```
PnSetText ( TxElementName, TxText )
```

Parameter:

TxElementName	Name des zu ändernden Panel-Elements.
TxText	Text, auf den das Element gesetzt werden soll.

Beschreibung:

Die Funktion kann nur auf solche Elemente angewendet werden, die eine Beschriftung besitzen oder deren aktueller Zustand durch einen Text ausgedrückt werden kann. Die Interpretation des Textes ist abhängig vom Typ des Elementes, z.B.:

Element	Bedeutung
Knopf	Beschriftung des Knopfes.
Label	Inhalt des Labels.
Eingabefeld (einzeilig)	Inhalt des Eingabefeldes.
Klappliste	Setzt die Selektion auf den angegebenen Text (muss in der Auswahlliste vorhanden sein).
Kombinationsfeld	Inhalt des Eingabefeldes.

Die Funktion kann nicht auf Elemente angewendet werden, die mit einer Variablen verknüpft sind.

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

Auf einer Panelseite sind diverse Labels vorhanden, die einge Kennwerte einer gerade selektierten Variablen (1.Messung, 1. Kanal) anzeigen. Bei jeder Änderung der Selektion in Messungs- oder Kanalliste werden die Felder entsprechend aktualisiert.

Ereignis-Sequenz 'Datenselektion geändert''

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   PnSetText("Page1.Name", TxVarName)
   PnSetText("Page1.Comment", Comm?(<TxVarName>))
   PnSetText("Page1.YUnit", Unit?(<TxVarName>, 1))
   PnSetText("Page1.XUnit", Unit?(<TxVarName>, 0))
END
```

Siehe auch:

PnGetText, PnSetValue

PnSetTimer

Anwendungsbereich: Panele

Der mit einer Panelseite verknüpfte Zeitgeber wird gestartet/ umkonfiguriert/ gestoppt.

Deklaration:

```
PnSetTimer ( Seitennummer, Intervall )
```

Parameter:

Seitennummer	Gibt die Seitennummer (1..) des zu steuernden Zeitgebers an.
Intervall	Gibt das Intervall (in Sekunden) an, mit dem der Zeitgeber auslösen soll. 0, um den Zeitgeber zu stoppen.

Beschreibung:

Für jede Panelseite kann ein Zeitgeber aktiviert werden, der mit einem vorgegebenen Zeitintervall arbeitet. Nach jedem Ablauf des Intervalls wird das "Zeitgeber"-Ereignis generiert, wodurch die Abarbeitung einer entsprechend definierten Ereignis-Sequenz ausgelöst wird.

Um den Zeitgeber für eine Seite zu verwenden, gehen Sie folgendermaßen vor:

- Aktivieren des Zeitgebers beim Design der Seite (Eigenschaft 'Zeitgeber' auf 'aktiv' setzen)
- Im FAMOS-Sequenzeditor finden Sie dann die Ereignis-Sequenz 'Zeitgeber' für die entsprechende Seite. Sie tragen hier die zyklisch abzuarbeitende Befehlsfolge ein.
- Starten des Zeitgebers mit dem gewünschten Zeitintervall durch die Funktion PnSetTimer. Sie könnten beispielsweise den Zeitgeber schon beim Laden des Panels automatisch starten (Aufruf in der Ereignis-Sequenz ' 'Panel Initialisierung') oder durch den Anwender manuell beim Drücken eines Knopfes (Ereignis-Sequenz 'Knopf gedrückt') starten lassen.

FAMOS versucht, das vorgegebene Intervall bestmöglichst einzuhalten, kann aber das exakte Auslösen des Ereignisses im gegebenen Takt nicht garantieren. Wenn das Betriebssystem oder FAMOS gerade mit anderen Aufgaben ausgelastet ist, kann sich die Generierung des Zeitgeber-Ereignisses unvorhersehbar verzögern. Insbesondere ist es so, dass die Ereignis-Sequenz 'Zeitgeber' nur ausgeführt werden kann, wenn FAMOS gerade keine anderen Sequenzen (mehr) ausführt. Wenn während der Abarbeitung einer anderen Sequenz Zeitgeber-Ereignisse auftreten, wird die dem Zeitgeber zugeordnete Ereignis-Sequenz erst später genau 1x ausgeführt, sobald FAMOS die aktuelle Abarbeitung beendet hat.

Daraus leitet sich auch ab, dass innerhalb einer Zeitgeber-Ereignis-Sequenz keine zeitaufwändigen Abläufe programmiert werden sollten. Beispielsweise kann ein Zeitgeber mit 1s Takt praktisch nicht realisiert werden, wenn schon die Abarbeitung der zugehörigen Ereignis-Sequenz länger als 1s dauert.

Beispiele:

Auf einer Panelseite soll die Messung des vorhandenen Datensatzes 'speed' simuliert werden und der wachsende Datensatz in einem Kurvenfenster angezeigt werden. Neben dem Kurvenfenster enthält die Seite außerdem einen Knopf, mit dem die Simulation gestartet/gestoppt werden soll, die Beschriftung des Knopfes wechselt entsprechend zwischen 'Start' und 'Stop'.

Im Kurvenfenster wird ein temporärer Datensatz 'Speed_Sim' angezeigt, an den zyklisch (Zeitgeber-gesteuert) die entsprechenden Werte des Original-Datensatzes angehängt werden.

Ereignis-Sequenz 'Panel Initialisierung'

```
SimIsRunning = 0
Speed_Sim = EMPTY
```

Ereignis-Sequenz 'Gedrückt' des 'Start/Stop'-Knopfes

```
IF NOT(SimIsRunning)
   ; Start
   Speed_Sim = EMPTY
   PnSetTimer(1, 0.5) ; Update-Intervall 500ms
   SimIsRunning = 1
   PnSetText( "Button1", "Stop")
ELSE
   ; Stop
   PnSetTimer(1, 0)
   SimIsRunning = 0
   PnSetText( "Button1", "Start")
END
```

Ereignis-Sequenz 'Zeitgeber'

```
ori_len= leng?(Speed)
next_index= leng?(Speed_Sim)+1
IF next_index <= ori_len
   ; alle 0.5s 5 neue Werte anhängen...
   Speed_Sim = Join( Speed_Sim, CutIndex( Speed, next_index, LowerValue( next_index+5, ori_len)))
```

```
ELSE
   ; Fertig, Timer stoppen...
   PnSetTimer( 1, 0)
   SimIsRunning = 0
   PnSetText( "Button1", "Start")
END
```

PnSetValue

Anwendungsbereich: Panele

Der numerische Wert des angegebenen Elements wird neu gesetzt.

Deklaration:

```
PnSetValue ( TxElementName, Wert )
```

Parameter:

TxElementName	Name des zu setzenden Panel-Elements.
Wert	Zahlenwert, auf den das Panel-Element gesetzt werden soll.

Beschreibung:

Die Funktion kann nur auf solche Panel-Elemente angewendet werden, deren aktueller Zustand durch eine Zahl ausgedrückt werden kann. Die Interpretation des Wertes ist abhängig vom Typ des Elementes, z.B.:

Element	Bedeutung
Label	Der numerische Wert wird als Inhalt des Labels eingetragen.
Schalter	Mit einer 0 wird der Status des Schalters auf "Aus" gesetzt, ansonsten auf "Ein".
Optionsgruppe	Die Option mit dem angegeben Index wird selektiert.
Bereichswahl, Zeitspanne	Setzt die Untergrenze des eingestellten Bereichs.

Die Funktion kann nicht auf Elemente angewendet werden, die mit einer Variablen verknüpft sind.

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

Ein Panel hat eine Seite 'Filter', die zur Einstellung von Filterparametern für einen Tiefpass dient. Zwei Eingabefelder 'Order' und 'Input_Freq' dienen zur Vorgabe von Filterordnung und Grenzfrequenz. Ein Schalter 'Bessel' gibt vor, ob mit Butterworth- oder Bessel-Charakteristik (Schalter ein) gerechnet werden soll. Beim Laden des Panels werden die Felder mit Standardwerten initialisiert. Auf Knopfdruck wird die Filterung dann für die aktuelle Selektion in der Variablenliste (Messungen), hier: 1.Kanal/1.Messung, ausgeführt.

Ereignis-Sequenz 'Panel Initialisierung'

```
PnSetValue("Filter.Order", 4)
PnSetValue("Filter.Input_Freq", 100)
PnSetValue("Filter.Bessel", 1)
```

Ereignis-Sequenz 'Gedrückt' des 'Filtern!'-Knopfes

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   Order = PnGetValue( "Order")
   Freq  = PnGetValue("Input_Freq")
   Type  = PnGetValue("Bessel")
   IF Order > 1 AND Freq > 0
      IF Type = 1  ; Bessel
         filtrat = FiltLP( <TxVarName>, 1, 0, Order, Freq)
      ELSE          ; Butterworth
         filtrat = FiltLP( <TxVarName>, 0, 0, Order, Freq)
      END
   END
END
```

Siehe auch:

PnGetValue, PnGetText, PnSetText

PnSetValue2

Anwendungsbereich: Panele

Der 2. numerische Wert des angegebenen Elements wird gesetzt.

Deklaration:

```
PnSetValue2 ( TxElementName, Wert )
```

Parameter:

TxElementName	Name des zu setzenden Panel-Elements.
Wert	Zahlenwert, auf den das Panel-Element gesetzt werden soll.

Beschreibung:

Die Funktion kann nur auf solche Panel-Elemente angewendet werden, deren aktueller Zustand durch 2 Zahlen ausgedrückt werden kann.

Element	Bedeutung
Zeitspanne	Der Wert gibt die obere Grenze des selektierten Bereichs an. Der übergebene Zeitwert muss im imc FAMOS-Zeitformat vorliegen, wie ihn z. B. Funktionen wie Time?(), TimeSystem?() und TimeJoin() erzeugen.
Bereichswahl	Der Wert gibt die obere Grenze des selektierten Bereichs an.

Die Funktion kann nicht auf Elemente angewendet werden, die mit einer Variablen verknüpft sind.

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Zeitspanne, Bereichswahl

Beispiele:

Aus einem Datensatz 'channel1' soll ein Bereich ausgeschnitten werden. Die Indizes für Beginn und Ende des Ausschnitts werden durch ein Bereichswahl-Widget 'Range' festgelegt. Nach dem Laden des Datensatzes wird das Widget so initialisiert, dass der maximal einstellbare Wert der Datensatzlänge (= Index des letzten Wertes) entspricht. Als initiale Auswahl wird das mittlere Fünftel des Datensatzes eingestellt.

```
len = Leng?(channel1)
PnSetProperty("Range", "Minimum", 1)
PnSetProperty("Range", "Maximum", len)
PnSetValue("Range", floor(len*0.4))
PnSetValue2("Range", floor(len*0.6))
```

Siehe auch:

PnSetValue, PnGetValue2

PnShow

Anwendungsbereich: Panele

Steuert die Sichtbarkeit des angegebenen Elements.

Deklaration:

```
PnShow ( TxElementName, Auftrag )
```

Parameter:

TxElementName	Name des zu steuernden Panel-Elements.
Auftrag	Auftrag
	0 : Element verstecken
	1 : Element anzeigen

Beschreibung:

Die Auswahl des anzusprechenden Elements kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Beispiele:

Eine Panel-Seite mit dem Namen 'Filter' enthält einen Knopf 'Execute', mit dem ein aktuell selektierter Kanal (1.Messung/1.Kanal) gefiltert werden soll. Wenn durch die aktuelle Selektion in Messungs- und Kanalliste kein solcher Kanal ausgewählt ist, wird der Knopf versteckt.

Ereignis-Sequenz 'Datenselektion geändert''

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   PnShow("Filter.Execute", 1)
ELSE
   PnShow("Filter.Execute", 0)
END
```

Siehe auch:

PnEnable

PnShowPage

Anwendungsbereich: Panele

Die Sichtbarkeit einer Panelseite wird gesteuert.

Deklaration:

```
PnShowPage ( SeitennummerOderTitel, Auftrag )
```

Parameter:

SeitennummerOderTitel	Gibt die Seitennummer (1..) oder den Titel der zu steuernden Seite an.
Auftrag	Auftrag
	0 : Seite verstecken
	1 : Seite sichtbar, aber gesperrt
	2 : Seite normal anzeigen

Beschreibung:

Beispiele:

Nach dem Laden eines Panels mit vielen Seiten werden zunächst nur die ersten 3 Seiten angezeigt. Eine weitere Seite wird gesperrt.

Ereignis-Sequenz 'Panel Initialisierung'

```
PnShowPage( "Report", 1)
FOR iPage = 4 TO PnGetPageCount()
   PnShowPage( iPage, 0)
END
```

Siehe auch:

PnGetPageCount, PnSetActivePage

PnTableColumns?

Anwendungsbereich: Panele

Ermittelt die Zahl der Spalten einer Tabelle.

Deklaration:

```
PnTableColumns? ( TxTabellenName ) -> Ergebnis
```

Parameter:

TxTabellenName	Name der abzufragenden Panel-Tabelle.
Ergebnis	Anzahl der Spalten

Beschreibung:

Die Anzahl der Spalten der angegebenen Tabelle im aktiven Panel wird ermittelt.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Tabelle, Datengitter

Beispiele:

Die Spalten einer Tabelle werden mit den einzelnen Events eines eventierten Datensatzes belegt.

```
Anzahl    = PnTableColumns?( "Tab1")
AnzahlEvn = EventNum?( Daten)
IF AnzahlEvn < Anzahl
   Anzahl = AnzahlEvn
END
i = 1
WHILE i <= Anzahl
   PnTableSetColumn("Tab1",i,2, Daten[i])
   i = i + 1
END
```

Siehe auch:

PnTableRows?, PnTableSetCell, PnTableSetColumn, PnTableSetRow

PnTableGetCellText

Anwendungsbereich: Panele

Fragt den Inhalt einer Tabellenzelle ab.

Deklaration:

```
PnTableGetCellText ( TxTabellenName, Spalte, Zeile ) -> TxInhalt
```

Parameter:

TxTabellenName	Name der abzufragenden Panel-Tabelle.
Spalte	Spalte (1..)
Zeile	Zeile (1..)
TxInhalt	Inhalt der angegebenen Tabellenzelle

Beschreibung:

Der Inhalt einer Zelle der angegebenen Tabelle im aktiven Panel wird abgefragt.

Die Angaben fuer Zeile und Spalte bestimmen die Position der zu lesenden Zelle, links oben ist [1,1]

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Tabelle

Beispiele:

Der Inhalt einer Tabellenzelle (links oben) wird abgefragt. In dieser ist als Platzhalter ein "xxx" für den Namen abgelegt. Dieser Platzhalter wird ersetzt und das Tabellenzelle aktualisiert.

```
Tx$= PnTableGetCellText( "Tab1", 1, 1, 0)
Tx$= TReplace( Tx$,"xxx","Heinz Muster")
err= PnTableSetCell( "Tab1", 1, 1, Tx$, 0 )
```

Siehe auch:

PnTableSetColumn, PnTableSetRow, PnTableSetCell

PnTableGetCellValue

Anwendungsbereich: Panele

Fragt den numerischen Wert einer Tabellenzelle ab.

Deklaration:

```
PnTableGetCellValue ( TxTabellenName, Spalte, Zeile ) -> Wert
```

Parameter:

TxTabellenName	Name der abzufragenden Panel-Tabelle.
Spalte	Spalte (1..)
Zeile	Zeile (1..)
Wert	Wert der angegebenen Tabellenzelle

Beschreibung:

Der Inhalt einer Zelle der angegebenen Tabelle im aktiven Panel wird abgefragt.

Die Angaben fuer Zeile und Spalte bestimmen die Position der zu lesenden Zelle, links oben ist [1,1]

Falls der Inhalt nicht in eine Zahl konvertiert werden kann, wird eine 0 zurückgeliefert.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Tabelle

Beispiele:

Der Inhalt einer Tabellenzelle (links oben) wird abgefragt. In dieser ist als Platzhalter ein "xxx" für den Namen abgelegt. Dieser Platzhalter wird ersetzt und das Tabellenzelle aktualisiert.

Die Angaben fuer Zeile und Spalte bestimmen die Position der abzufragenden Zelle, links oben ist [1,1]

```
Tx$= PnTableGetCellText( "Tab1", 1, 1, 0)
Tx$= TReplace( Tx$,"xxx","Heinz Muster")
PnTableSetCell( "Tab1", 1, 1, Tx$, 0 )
```

Siehe auch:

PnTableSetColumn, PnTableSetRow, PnTableSetCell

PnTableGetSelectedRows

Anwendungsbereich: Panele

Ermittelt die aktuell selektierten Zeilen in einem Datengitter.

Deklaration:

```
PnTableGetSelectedRows ( TxElementName ) -> Ergebnis
```

Parameter:

TxElementName	Name des Panel-Elements.
Ergebnis	Datensatz, enthält die Indizes der selektierten Zeilen.

Beschreibung:

Der zurückgegebene Datensatz kann die Länge 0 (keine Zeile selektiert), 1 (eine Zeile selektiert) oder > 1 (bei Mehrfach-Selektion) haben.

Die ermittelten Indizes beziehen sich auf die absolute Position im angezeigten Datensatz. Der sichtbare Index der selektierten Zeile kann davon unter Umständen abweichen, beispielsweise wenn eine Sortierung auf die Tabelle angewendet wurde.

Die Reihenfolge der ermittelten Indizes entspricht der sichtbaren Reihenfolge der selektierten Einträge in der Tabelle von oben nach unten.

Die erste Zeile hat den Index 1.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Datengitter

Beispiele:

Ein Panelseite enthält eine 2-spaltige Datengitter 'tab'. In dieser werden die Datensätze 'channel_x' und 'channel_y' angezeigt. Auf Betätigung eines Knopfes 'Ausführen!' werden die aktuell selektierten Zeilen ausgelesen und daraus ein neuer XY-Datensatz gebaut.

Ereignis-Sequenz 'Panel Initialisierung'

```
PnTableSetColumn("tab1", 1, 1, channel_x)
PnTableSetColumn("tab1", 2, 1, channel_y)
```

Ereignis-Sequenz 'Gedrückt' des 'Ausführen!'-Knopfes

```
sel = PnTableGetSelectedRows("tab1")
IF leng?(sel) = 0
   BoxMessage("Fehler", "Bitte selektieren Sie zunächst die gewünschten Zeilen in der Tabelle!", "!1")
ELSE
   sel = Sort(sel, 1) ; falls Tabelle sortierbar ist
   x = EMPTY
   y = EMPTY
   FOR I = 1 TO leng?(sel)
      index = sel[I]
      x = Join(x, Channel_x[index])
      y = Join(y, Channel_y[index])
   END
   Result = xyof(x,y)
END
```

Siehe auch:

PnTableSetColumn

PnTableRows?

Anwendungsbereich: Panele

Ermittelt die Zahl der Zeilen einer Tabelle.

Deklaration:

```
PnTableRows? ( TxTabellenName ) -> Ergebnis
```

Parameter:

TxTabellenName	Name der abzufragenden Panel-Tabelle.
Ergebnis	Anzahl der Zeilen

Beschreibung:

Die Anzahl der Zeilen der angegebenen Tabelle im aktiven Panel wird ermittelt.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Tabelle, Datengitter

Beispiele:

Die Zeilen einer Tabelle werden mit den einzelnen Segmenten eines segmentierten Datensatzes belegt.

```
Anzahl    = PnTableRows?( "Tab1")
AnzahlSeg = leng?( Daten) / SegLen?( Daten)
IF AnzahlSeg < Anzahl
   Anzahl = AnzahlSeg
END
i = 1
WHILE i <= Anzahl
   PnTableSetRow ("Tab1",1,i, Daten[i])
   i = i + 1
END
```

Siehe auch:

PnTableColumns?, PnTableSetCell, PnTableSetColumn, PnTableSetRow

PnTableSetCell

Anwendungsbereich: Panele

Setzt den Inhalt einer Tabellenzelle.

Deklaration:

```
PnTableSetCell ( TxTabellenName, Spalte, Zeile, Inhalt )
```

Parameter:

TxTabellenName	Name der zu steuernden Panel-Tabelle.
Spalte	Spalte (1..)
Zeile	Zeile (1..) bzw. 0 für Spaltenkopf
Inhalt	Zu übertragende Zahl oder Text

Beschreibung:

Eine Zelle der angegebenen Tabelle im aktiven Panel wird mit einem Text oder einem einzelnen Wert belegt.

Die Angaben fuer Zeile und Spalte bestimmen die Position der zu beschreibenden Zelle, links oben ist [1,1]

Wenn bei [Inhalt] ein Datensatz angegeben wird, der eine Länge größer 1 besitzt, wird der letzte Wert des Datensatzes verwendet.

Die Funktion kann nicht auf Tabellenzellen angewendet werden, die mit einer Variablen verknüpft sind.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Die Funktion kann auch verwendet werden, um bei Elementen vom Typ [Datengitter] den Text im Spaltenkopf zu setzen. Für [Zeile] ist dann eine 0 anzugeben, für [Inhalt] ist dann nur Text erlaubt.

Anwendbar auf:

Tabelle, Datengitter (nur Spaltenkopf)

Beispiele:

Die zweite Spalte einer Tabelle wird mit einem gerade berechneten Datensatz belegt. Die erste Zeile enthält den Variablennamen, die zweite Zeile das Maximum des Datensatzes in einem festen Format. Ab der 3. Zeile beginnen die Zahlenwerte.

```
Kanal1= ...
PnTableSetCell( "Tab1", 2, 1, "Kanal1")
TxMax$ = TForm( Max( Kanal1), "f32")
PnTableSetCell(   "Tab1", 2, 2, TxMax$)
PnTableSetColumn( "Tab1", 2, 3, Kanal1)
```

Siehe auch:

PnTableSetColumn, PnTableSetRow, PnTableGetCellText

PnTableSetColumn

Anwendungsbereich: Panele

Setzt den Inhalt einer Tabellenspalte.

Deklaration:

```
PnTableSetColumn ( TxTabellenName, Spalte, Zeile, Inhalt )
```

Parameter:

TxTabellenName	Name der zu steuernden Panel-Tabelle.
Spalte	Spalte (1..)
Zeile	Zeile (1..)
Inhalt	Erlaubte Datentypen: Normaler Datensatz, Textfeld.

Beschreibung:

Eine Spalte der angegebenen Tabelle im aktiven Panel wird mit einem Datensatz belegt. Die einzelnen Zahlen/Texte werden entsprechend der in der Tabelle festgelegten Formatierung eingetragen. Die erste Zahl wird an der angegebenen Position eingetragen, alle weiteren darunter.

Die Angaben fuer Zeile und Spalte bestimmen die Position der ersten zu beschreibenden Zelle, links oben ist [1,1]

Es werden so viele Werte übertragen, bis entweder der letzte Wert des Datensatzes oder die letzte Tabellenzeile erreicht ist.

Zur Übertragung von Text in eine Tabellenzelle verwenden Sie die Funktion PnTableSetCell().

Die Funktion kann nicht auf Tabellenzellen angewendet werden, die mit einer Variablen verknüpft sind.

Wenn es sich um ein Datengitter handelt, muss der Parameter [Zeile] auf 1 gesetzt werden.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Tabelle, Datengitter

Beispiele:

Die Spalten einer Tabelle werden mit den einzelnen Events eines eventierten Datensatzes belegt.

```
Anzahl    = PnTableColumns?( "Tab1")
AnzahlEvn = EventNum?( Daten)
IF AnzahlEvn < Anzahl
   Anzahl = AnzahlEvn
END
i = 1
WHILE i <= Anzahl
   PnTableSetColumn( "Tab1",i,2, Daten[i])
   i = i + 1
END
```

Siehe auch:

PnTableSetRow, PnTableSetCell

PnTableSetDim

Anwendungsbereich: Panele

Setzt die Anzahl der Spalten und Zeilen einer Tabelle.

Deklaration:

```
PnTableSetDim ( TxElementName, Spalten, Zeilen, Option )
```

Parameter:

TxElementName	Name des Panel-Elements.
Spalten	Neue Anzahl der Spalten
Zeilen	Neue Anzahl der Zeilen
Option	Option zur Steuerung der Tabellengröße.
	0 : Die aktuelle Tabellengröße wird beibehalten.
	1 : Die Tabellengröße wird angepasst.

Beschreibung:

Mit dieser Funktion kann die Spalten- und Zeilenanzahl der Tabelle gesteuert werden.

Wenn [Spalten] oder [Zeilen] 0 sind, wird die aktuelle Spalten- bzw. Zeilenanzahl beibehalten.

Mit [Option] kann die Größe der Tabelle, wie folgt, gesteuert werden:

0: Die aktuelle Tabellenposition und -größe wird beibehalten. Die aktuellen Spaltenbreiten bzw. Zeilenhöhen werden proportional angepasst.

1: Die Tabellengröße wird angepasst (die Position links/oben bleibt unverändert):

- Beim Vergrößern der Spaltenzahl: Die Breite der Tabelle wird vergrößert. Die neuen Spalten erhalten die Breite und die weiteren Eigenschaften (außer Inhalt) der bisher letzten Spalte.
- Beim Vergrößern der Zeilenzahl: Die Höhe der Tabelle wird vergrößert. Die neuen Zeilen erhalten die die Höhe und die weiteren Eigenschaften (außer Inhalt) der bisher letzten Zeile.
- Beim Verkleinern der Zeilen- bzw. Spaltenzahl werden die nun überzähligen Zellen gelöscht, die Höhe/Breite der Tabelle verringert sich also entsprechend unter Beibehaltung der Größe der verbleibenden Zellen.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Tabelle

Beispiele:

Die Spalten einer Tabelle werden mit den einzelnen Events eines eventierten Datensatzes belegt. Die Spaltenzahl wird dabei auf die Anzahl der Events und die Zeilenanzahl auf die maximale Länge der Events + 1 gesetzt, sodass der gesamte Datensatz ab der 2. Zeile angezeigt wird.

```
eventCount = EventNum?(data)
maxEventLength = 0
FOREACH EVENT event IN data
   maxEventLength = UpperValue(maxEventLength, Leng?(event))
END
PnTableSetDim("Tab1", eventCount, maxEventLength + 1,  1)
i = 1
WHILE i <= eventCount
   PnTableSetColumn("Tab1", i, 2, data[i])
   i = i + 1
END
```

Siehe auch:

PnTableColumns?, PnTableRows?, PnTableSetCell, PnSetProperty

PnTableSetProperty

Anwendungsbereich: Panele

Eine Eigenschaft der gewählten Tabellenzelle wird neu gesetzt.

Deklaration:

```
PnTableSetProperty ( TxElementName, Spalte, Zeile, TxPropName, Wert )
```

Parameter:

TxElementName	Name des Panel-Elements.
Spalte	Spalte (1..)
Zeile	Zeile (1..)
TxPropName	Kennung der zu setzenden Eigenschaft.
	"FillColor" : Hintergrundfarbe
	"FrameColor" : Rahmenfarbe
	"TextColor" : Textfarbe
Wert	Zahlenwert oder Text, auf den die Eigenschaft gesetzt werden soll.

Beschreibung:

Die zu setzende Eigenschaft wird durch vordefinierte Bezeichner ausgewählt.

Die folgenden Eigenschaften sind bisher definiert:

Kennung	Bedeutung
"FillColor"	Hintergrundfarbe (RGB-Wert)
"FrameColor"	Rahmenfarbe (RGB-Wert)
"TextColor"	Textfarbe (RGB-Wert)

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Tabelle

Beispiele:

Ein Panelseite enthält eine Tabelle 'tab' zur Anzeige verschiedener Kennwerte der gerade selektierten Variablen (1.Messung, 1.Kanal). Bei jeder Änderung der Selektion in Messungs- oder Kanalliste wird die Tabelle entsprechend aktualisiert. Unter anderem wird das Maximum des Datensatzes bestimmt und in die 3. Zeile der 2. Spalte eingetragen. Wenn der Wert einen bestimmten Grenzwert überschreitet, wird die Hintergrundfarbe der Zelle auf Rot geändert.

Ereignis-Sequenz 'Datenselektion geändert"

```
TxVarName = SelBuildVarName(1, 1, 0)
IF TxVarName <> ""
   maxval = Max(<TxVarName>)
   PnTableSetCell("Page1.tab", 2, 3, maxval )
   IF maxval > 10
      ; Grenzwert überschritten (roter Hintergrund)
      PnTableSetProperty("Page1.tab", 2, 3, "FillColor", RGB( 255, 0, 0)))
   ELSE
      ; Standard (weißer Hintergrund)
      PnTableSetProperty("Page1.tab", 2, 3, "FillColor", RGB( 255, 255, 255)))
   END
   ; ... weitere Zellen der Tabelle füllen
END
```

Siehe auch:

PnSetValue, PnSetProperty

PnTableSetRow

Anwendungsbereich: Panele

Setzt den Inhalt einer Tabellenzeile.

Deklaration:

```
PnTableSetRow ( TxTabellenName, Spalte, Zeile, Daten )
```

Parameter:

TxTabellenName	Name der zu steuernden Panel-Tabelle.
Spalte	Spalte (1..)
Zeile	Zeile (1..)
Daten	Erlaubte Datentypen: Normaler Datensatz, Textfeld.

Beschreibung:

Eine Zeile der angegebenen Tabelle im aktiven Panel wird mit einem Datensatz belegt. Die einzelnen Zahlen/Texte werden entsprechend der in der Tabelle festgelegten Formatierung eingetragen. Die erste Zahl wird an der angegebenen Position eingetragen, alle weiteren rechts davon.

Die Angaben fuer Zeile und Spalte bestimmen die Position der ersten zu setzenden Zelle, links oben ist [1,1]

Es werden so viele Werte übertragen, bis entweder der letzte Wert des Datensatzes oder die letzte Tabellenspalte erreicht ist.

Zur Übertragung von Text in eine Tabellenzelle verwenden Sie die Funktion PnTableSetCell().

Die Funktion kann nicht auf Tabellenzellen angewendet werden, die mit einer Variablen verknüpft sind.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Tabelle

Beispiele:

Die Zeilen einer Tabelle werden mit den einzelnen Segmenten eines segmentierten Datensatzes belegt.

```
Anzahl    = PnTableRows?( "Tab1")
AnzahlSeg = leng?( Daten) / Seglen?( Daten)
IF AnzahlSeg < Anzahl
   Anzahl = AnzahlSeg
END
i = 1
WHILE i <= Anzahl
   PnTableSetRow ( "Tab1",1,i, Daten[i])
   i = i + 1
END
```

Siehe auch:

PnTableSetColumn, PnTableSetCell

PnTableShowColumn

Anwendungsbereich: Panele

Steuert die Sichtbarkeit einer Datengitterspalte.

Deklaration:

```
PnTableShowColumn ( TxTabellenName, Spalte, Auftrag )
```

Parameter:

TxTabellenName	Name der zu steuernden Panel-Tabelle.
Spalte	Spalte (1..)
Auftrag	Auftrag
	0 : Spalte verstecken
	1 : Spalte anzeigen

Beschreibung:

Die Sichtbarkeit einer Spalte der angegebenen Tabelle im aktiven Panel wird gesteuert.

Die Auswahl der anzusprechenden Tabelle kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird das Element wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Element mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Anwendbar auf:

Datengitter

Beispiele:

Ein Datengitter wurde zur Designzeit mit 10 Spalten definiert. Zur Laufzeit werden die Spalten mit den einzelnen Events eines eventierten Datensatzes belegt. Wenn der Datensatz weniger als 10 Events hat, werden die nicht benötigten Spalten versteckt.

```
AnzahlSpalten  = PnTableColumns?( "Grid1")
AnzahlEvn = EventNum?( Daten)
IF AnzahlEvn > AnzahlSpalten
   AnzahlEvn = AnzahlSpalten
END
i = 1
WHILE i <= AnzahlEvn
   PnTableSetColumn( "Grid1",i,1, Daten[i])
   i = i + 1
END
WHILE i <= AnzahlSpalten
   PnTableShowColumn( "Grid1",i, 0)
   i = i + 1
END
```

Siehe auch:

PnTableSetColumn, PnTableColumns?

PnTreeDeleteNode

Anwendungsbereich: Panele

Löschen von Knoten in einer Baumansicht.

Deklaration:

```
PnTreeDeleteNode ( TxElementName, Node, ChildPosition )
```

Parameter:

TxElementName	Name des Baumansicht-Elements.
Node	Absoluter Index oder eindeutiger Schlüssel des zu löschenden Knotens bzw. des zugehörigen Elternknotens. Geben Sie eine -1 an, um alle Knoten zu löschen. Mit einer 0 adressieren Sie einen Knoten der Wurzelebene, der nächste Parameter gibt dann die Position innerhalb der Wurzelebene an.
ChildPosition	Kind-Position
	0 : [Node] identifiziert den/die zu löschenden Knoten direkt.
	-1 : Alle Kindknoten von [Node] werden gelöscht. Wenn [Node] = 0, werden alle Knoten außer den Wurzelknoten gelöscht.
	>=1 : Position des gewünschten Knotens unter den unmittelbaren Kindknoten von [Node] bzw. innerhalb der Wurzelebene, wenn für [Node] eine 0 angegeben wurde.

Beschreibung:

Beispiele:

```
; Löscht den gesamten Inhalt der Baumansicht
PnTreeDeleteNode("TreeView1", -1, 0)

; Löscht alle Knoten außer der Wurzelebene
PnTreeDeleteNode("TreeView1", 0, -1)

; Löscht den 2. Knoten der Wurzelebene komplett
PnTreeDeleteNode("TreeView1", 0, 2)

; Löscht im 2. Knoten der Wurzelebene den 3. Kindknoten
index = PnTreeGetNodeState("TreeView1", 0, 2, "AbsoluteIndex")
PnTreeDeleteNode("TreeView1", index, 3)

; Löscht im 2. Knoten der Wurzelebene alle Kindknoten
index = PnTreeGetNodeState("TreeView1", 0, 2, "AbsoluteIndex")
PnTreeDeleteNode("TreeView1", index, -1)

; Löscht den Knoten mit dem Schlüssel "#node11"
PnTreeDeleteNode("TreeView1", "#node_11", 0)

; Löscht alle Kinder des Knotens mit dem Schlüssel "#node11"
PnTreeDeleteNode("TreeView1", "#node_11", -1)

; Sucht nach einem Knoten mit der Beschriftung "Mr. Smith" und löscht den gesamten Zweig,
; in dem sich der Knoten befindet
index = PnTreeFindNodes("TreeView1", 0, 0, "Mr. Smith", "Caption")
IF leng?(index) = 1
   rootIndex = PnTreeGetNodeState("TreeView1", index, 0, "RootNode")
   PnTreeDeleteNode("TreeView1", rootIndex, 0)
END
```

Siehe auch:

PnTreeInsertNode, PnTreeSetNodeProp

Unterstützt ab:

Version 2023

PnTreeFindNodes

Anwendungsbereich: Panele

Suche nach Knoten in einer Baumansicht, die eine gegebene Bedingung erfüllen.

Deklaration:

```
PnTreeFindNodes ( TxElementName, ParentNode, Option, TxPattern, TxCondition ) -> NodeIndizes
```

Parameter:

TxElementName	Name des Baumansicht-Elements.
ParentNode	Absoluter Index oder eindeutiger Schlüssel des Knotens, unter dessen Kindern die Suche ausgeführt werden soll. Geben Sie eine 0 an, um über alle Knoten bzw. innerhalb der Wurzelknoten zu suchen.
Option	Suchbereich
	0 : Unter allen Kindknoten suchen. Für [ParentNode] = 0 wird die Suche über alle Knoten des Baumes ausgeführt.
	1 : Nur unter direkten Kindknoten suchen. Für [ParentNode] = 0 wird die Suche über die Wurzelknoten des Baumes ausgeführt.
TxPattern	Muster, nach dem gesucht werden soll.
TxCondition	Welche Bedingung muss erfüllt sein?
	"Caption" : Die Beschriftung des Knotens entspricht genau dem Suchmuster. Bezüglich Groß-/Kleinschreibung wird nicht unterschieden.
	"NodeKey" : Der Knoten-Schlüssel entspricht genau dem Suchmuster. Bezüglich Groß-/Kleinschreibung wird nicht unterschieden.
	"Caption*" : Die Beschriftung des Knotens entspricht dem Suchmuster. Das Suchmuster die kann die Jokerzeichen "*" (beliebig viele beliebige Zeichen) und "?" (ein beliebiges Zeichen) enthalten. Wenn das erste Zeichen des Suchmusters ein "!" ist, wird die Bedingung negiert, es werden also alle Knoten gefunden, die NICHT dem Suchmuster entsprechen. Bezüglich Groß-/Kleinschreibung wird nicht unterschieden.
	"NodeKey*" : Der Knoten-Schlüssel entspricht dem Suchmuster. Das Suchmuster die kann die Jokerzeichen "*" (beliebig viele beliebige Zeichen) und "?" (ein beliebiges Zeichen) enthalten. Wenn das erste Zeichen des Suchmusters ein "!" ist, wird die Bedingung negiert, es werden also alle Knoten gefunden, die NICHT dem Suchmuster entsprechen. Bezüglich Groß-/Kleinschreibung wird nicht unterschieden.
NodeIndizes	Datensatz, enthält die absoluten Indizes der gefundenen Knoten. Der erste Knoten im Baum hat den Index 1. Leerer Datensatz (Länge 0), wenn kein Knoten der Bedingung entspricht.

Beschreibung:

Beispiele:

Sucht nach einem Knoten mit der Beschriftung "Mr. Smith" und löscht den gesamten Zweig, in dem sich der Knoten befindet:

```
index = PnTreeFindNodes("TreeView1", 0, 0, "Mr. Smith", "Caption")
rootIndex = PnTreeGetNodeState("TreeView1", index, 0, "RootNode")
PnTreeDeleteNode("TreeView1", rootIndex, 0)
```

In einem Baum-Widget werden alle Einträge gesucht, die direkte Kinder des Knotens mit dem Schlüssel "#datafiles" sind und deren Beschriftung auf ".dat" endet. Die gefundenen Beschriftungen werden in einem Textfeld gespeichert.

```
TxDatFiles = TxArrayCreate(0)
fileNodeIndex = PnTreeFindNodes("TreeView1", 0, 0, "#datafiles", "NodeKey")
nodeIndizes = PnTreeFindNodes("TreeView1", fileNodesIndex, 1, "*.dat", "Caption*")
count = leng?(nodeIndizes)
FOR i = 1 TO count
   TxDatFiles[i] = PnTreeGetNodeProp("TreeView1", nodeIndizes[i], 0, "Caption")
END
```

Siehe auch:

PnTreeGetNodeProp, PnTreeGetNodeState, PnTreeGetNodes

Unterstützt ab:

Version 2023

PnTreeGetNodeCount

Anwendungsbereich: Panele

Die Anzahl der Knoten in einer Baumansicht wird ermittelt.

Deklaration:

```
PnTreeGetNodeCount ( TxElementName, ParentNode, Option ) -> Count
```

Parameter:

TxElementName	Name des Baumansicht-Elements.
ParentNode	Absoluter Index oder eindeutiger Schlüssel des Knotens, von dem die Anzahl der Kindknoten bestimmt werden soll. Geben Sie eine 0 an, um die Anzahl aller Knoten bzw. die Anzahl der Wurzelknoten zu bestimmen.
Option	Option
	0 : Alle Kindknoten berücksichtigen. Für [ParentNode] = 0 wird die Anzahl aller Knoten im Baum bestimmt.
	1 : Nur direkte Kindknoten berücksichtigen. Für [ParentNode] = 0 wird die Anzahl aller Wurzelknoten bestimmt.
Count	Anzahl der Knoten

Beschreibung:

Beispiele:

```
; Gesamtzahl aller Knoten im Baum bestimmen:
count = PnTreeGetNodeCount("TreeView1", 0, 0)

; Anzahl der Wurzelnoten bestimmen:
root_count = PnTreeGetNodeCount("TreeView1", 0, 1)

; Anzahl der direkten Kinder des letzten Wurzelknotens bestimmen:
lastroot_index = PnTreeGetNodeState("TreeView1", 0, root_count, "AbsoluteIndex")
lastroot_children_count = PnTreeGetNodeCount("TreeView1", lastroot_index, 1)

; Die Anzahl der direkten Kindes des Knotens mit der Beschriftung "Source Files" bestimmen:
file_group_index = PnTreeFindNodes("TreeView1", 0, 0, "Source Files", "Caption")
file_count = PnTreeGetNodeCount("TreeView1", file_group_index, 1)
```

Siehe auch:

PnTreeGetNodeProp, PnTreeGetNodeState, PnTreeGetNodes

Unterstützt ab:

Version 2023

PnTreeGetNodeProp

Anwendungsbereich: Panele

Eine Eigenschaft des angegebenen Knotens wird abgefragt.

Deklaration:

```
PnTreeGetNodeProp ( TxElementName, Node, ChildPosition, TxPropertyID ) -> TxText
```

Parameter:

TxElementName	Name des Baumansicht-Elements.
Node	Absoluter Index oder eindeutiger Schlüssel des zu verwendenden Knotens bzw. des zugehörigen Elternknotens. Mit einer 0 adressieren Sie einen Knoten der Wurzelebene, der nächste Parameter gibt dann die Position innerhalb der Wurzelebene an.
ChildPosition	Kind-Position
	0 : [Node] identifiziert den zu verwendenden Knoten direkt.
	>=1 : Position des gewünschten Knotens unter den unmittelbaren Kindknoten von [Node] bzw. innerhalb der Wurzelebene, wenn für [Node] eine 0 angegeben wurde.
TxPropertyID	Auswahl der abzufragenden Eigenschaft
	"Caption" : Beschriftung des Knotens.
	"NodeKey" : Schlüssel/Zusatztext. Der hier angegebene Text kann als Schlüssel verwendet werden, um diesen Knoten später leicht in anderen Funktionen zur Steuerung der Baumansicht spezifizieren zu können. In dieser Anwendung sollte der Schlüssel natürlich eindeutig sein. Bei Schlüsseln wird bezüglich Groß-/Kleinschreibung nicht unterschieden. Alternativ kann der Text auch verwendet werden, um zusätzliche Informationen am Knoten zu speichern. Beispielsweise könnte beim Füllen des Baumes mit Dateinamen als Beschriftung nur der "kurze" Dateiname angezeigt werden und im Schlüssel der komplette Dateipfad abgelegt werden.
	"ImageKey" : Bild-Schlüssel. Legt das anzuzeigende Miniaturbild für diesen Knoten fest. Die verfügbaren Bilder müssen zusammen mit einem eindeutigen identifizierenden Schlüssel zur Design-Zeit in der Eigenschaft "Bildliste" der Baumansicht definiert werden. Leerer Text "", wenn kein Bild angezeigt werden soll.
TxText	Aktueller Wert der abgefragten Eigenschaft.

Beschreibung:

Um den gewünschten Knoten zu adressieren, stehen verschiedene Möglichkeiten zur Verfügung:

- **Absoluter Index:** Der absolute Index des Knotens unter allen vorhandenen Knoten. Entspricht der Zeilennummer, in der sich der Knoten befindet, wenn alle Knoten aufgeklappt sind. Der erste Knoten der Wurzelebene hat den Index 1.
- **Schlüssel:** Beim Anlegen eines Knotens kann vom Anwender ein optionaler textueller Schlüssel definiert werden, durch den der Knoten dann später wieder angesprochen werden kann. Der vergebene Schlüssel sollte darum unbedingt eindeutig sein. Bezüglich Groß-/Kleinschreibung wird dabei nicht unterschieden.
- **Kind-Position:** Gibt die Position des Knotens unter allen unmittelbaren Kindknoten eines gegebenen Elternknotens an. Bei Knoten der Wurzelebene die Position innerhalb der Wurzelebene (entspricht der Zeilennummer, wenn alle Knoten zugeklappt sind). Der erste Kindknoten hat die Position 1.

Beispiele:

Die Beschriftungen aller markierten Knoten einer Baumansicht werden in ein Textfeld geschrieben.

```
indizes = PnTreeGetNodes("TreeView1", 0, 0, "Selected")
count = leng?(indizes)
TxaCheckedNodes = TxArrayCreate(count)
FOR i = 1 TO count
   caption = PnTreeGetNodeProp("TreeView1", indizes[i], 0, "Caption")
   TxaCheckedNodes[i] = caption
END
```

Für alle Knoten der Wurzelebene, die bisher kein Miniaturbild haben, wird ein Standardbild zugewiesen. Dieses wurde beim Design der Baumansicht in der Eigenschaft "Bildliste" mit dem Schlüssel "GenericFileIcon" definiert.

```
count = PnTreeGetNodeCount("TreeView1", 0, 1)
FOR i = 1 TO count
   imagekey = PnTreeGetNodeProp("TreeView1", 0, i, "ImageKey")
   IF imageKey = ""
      PnTreeSetNodeProp("TreeView1", 0, i, "ImageKey", "GenericFileIcon")
   END
```

```
END
```

Die Beschriftung aller Knoten, die auf ".dat" enden, wird in Großschreibung gewandelt.

```
indizes = PnTreeFindNodes("TreeView1", 0, 0, "*.dat", "Caption*")
FOREACH SAMPLE index in indizes
   caption = PnTreeGetNodeProp("TreeView1", index, 0, "Caption")
   PnTreeSetNodeProp("TreeView1", index, 0, "Caption", TConv(caption, 2))
END
```

Siehe auch:

PnTreeInsertNode, PnTreeSetNodeProp

Unterstützt ab:

Version 2023

PnTreeGetNodes

Anwendungsbereich: Panele

Ermittelt die aktuell selektierten/markierten Knoten, die Wurzelknoten oder die Kindknoten zu einem gegebenen Elternknoten in einer Baumansicht.

Deklaration:

```
PnTreeGetNodes ( TxElementName, ParentNode, Option, TxCondition ) -> NodeIndizes
```

Parameter:

TxElementName	Name des Baumansicht-Elements.
ParentNode	Absoluter Index oder eindeutiger Schlüssel des Knotens, unter dessen Kindern die Suche ausgeführt werden soll. Geben Sie eine 0 an, um über alle Knoten bzw. innerhalb der Wurzelknoten zu suchen.
Option	Bestimmt die Suchtiefe.
	0 : Alle Kindknoten berücksichtigen
	1 : Nur direkte Kindknoten berücksichtigen
TxCondition	Welche Bedingung muss erfüllt sein?
	"Selected" : Es werden alle selektierten Knoten geliefert.
	"Checked" : Es werden alle markierten Knoten geliefert.
	"All" : Zu einem gegebenen Elternknoten werden alle Kinderknoten geliefert. Für [ParentNode]=0 und [Option]=1 werden alle Knoten der Wurzelebene geliefert.
NodeIndizes	Datensatz, enthält die absoluten Indizes der gefundenen Knoten. Der erste Knoten im Baum hat den Index 1. Leerer Datensatz (Länge 0), wenn kein Knoten der Bedingung entspricht.

Beispiele:

```
; Ermittelt die absoluten Indizes aller selektierten Knoten in der Wurzelebene
sel_nodes_root = PnTreeGetNodes("TreeView1", 0, 1, "Selected")

; Ermittelt die absoluten Indizes aller markierten Knoten im gesamten Baum:
checked_nodes_global = PnTreeGetNodes("TreeView1", 0, 0, "Checked")

; Ermittelt die absoluten Indizes aller Wurzelknoten
root_nodes = PnTreeGetNodes("TreeView1", 0, 1, "all")

; Ermittelt alle markierten, direkten Kinder des des Knotens mit der Beschriftung "Source Files"
files_group_index = PnTreeFindNodes("TreeView1", 0, 0, "Source Files", "Caption")
files_checked = PnTreeGetNodes("TreeView1", file_group_index, 1, "Checked")

; Legal, aber sinnfrei: Ermittelt die absoluten Indizes aller Knoten
Das Ergebnis enthält die ganzen Zahlen von 1 bis zur Gesamtzahl der Knoten.
all_nodes = PnTreeGetNodes("TreeView1", 0, 0, "all")
```

Siehe auch:

PnTreeGetNodeProp, PnTreeGetNodeState, PnTreeFindNodes

Unterstützt ab:

Version 2023

PnTreeGetNodeState

Anwendungsbereich: Panele

Der aktuelle Zustand eines Baumknotens wird abgefragt.

Deklaration:

```
PnTreeGetNodeState ( TxElementName, Node, ChildPosition, TxStateID ) -> State
```

Parameter:

TxElementName	Name des Baumansicht-Elements.
Node	Absoluter Index oder eindeutiger Schlüssel des zu verwendenden Knotens bzw. des zugehörigen Elternknotens. Mit einer 0 adressieren Sie einen Knoten der Wurzelebene, der nächste Parameter gibt dann die Position innerhalb der Wurzelebene an.
ChildPosition	Kind-Position
	0 : [Node] identifiziert den zu verwendenden Knoten direkt.
	>=1 : Position des gewünschten Knotens unter den unmittelbaren Kindknoten von [Node] bzw. innerhalb der Wurzelebene, wenn für [Node] eine 0 angegeben wurde.
TxStateID	Welche Zustands-Eigenschaft soll ermittelt werden?
	"Selected" : 1, wenn der Knoten selektiert ist. 0 sonst.
	"Checked" : 1, wenn der Knoten markiert ist. 0 wenn nicht. -1 bei unbestimmter Markierung (möglich für Elternknoten bei "Markierungs-Modus"= "rekursiv").
	"Expanded" : 1, wenn der Knoten aufgeklappt ist. 0 sonst.
	"Level" : Ebene (Einzug) des Knotens. Wurzelknoten haben die Ebene 0.
	"ParentNode" : Absoluter Index des Elternknotens oder 0, falls sich der Knoten in der Wurzelebene findet.
	"RootNode" : Absoluter Index des zugehörigen Wurzelknotens.
	"Position" : Position des Knotens unter den direkten Kindern des Elternknotens. Bei Knoten der Wurzelebene die Position innerhalb der Wurzelebene.
	"AbsoluteIndex" : Absoluter Index des Knotens im Baum.
	"IsLeaf" : 1, wenn der Knoten ein Blatt ist, also ein Endpunkt in der Hierarchie ohne eigene Kinder ist. 0 sonst. Invers zu "HasChildren".
	"HasChildren" : 1, wenn der Knoten Kindknoten besitzt, 0 sonst. Invers zu "IsLeaf".
State	Ermittelter Zustand, ganzzahlig und >-1. Interpretation abhängig von [TxStateID].

Beschreibung:

Um den gewünschten Knoten zu adressieren, stehen verschiedene Möglichkeiten zur Verfügung:

- **Absoluter Index:** Der absolute Index des Knotens unter allen vorhandenen Knoten. Entspricht der Zeilennummer, in der sich der Knoten befindet, wenn alle Knoten aufgeklappt sind. Der erste Knoten der Wurzelebene hat den Index 1.
- **Schlüssel:** Beim Anlegen eines Knotens kann vom Anwender ein optionaler textueller Schlüssel definiert werden, durch den der Knoten dann später wieder angesprochen werden kann. Der vergebene Schlüssel sollte darum unbedingt eindeutig sein. Bezüglich Groß-/Kleinschreibung wird dabei nicht unterschieden.
- **Kind-Position:** Gibt die Position des Knotens unter allen unmittelbaren Kindknoten eines gegebenen Elternknotens an. Bei Knoten der Wurzelebene die Position innerhalb der Wurzelebene (entspricht der Zeilennummer, wenn alle Knoten zugeklappt sind). Der erste Kindknoten hat die Position 1.

Beispiele:

Die Anzahl der direkten Kinder des 2. Wurzelknotens wird bestimmt.

```
index = PnTreeGetNodeState("TreeView1", 0, 2, "AbsoluteIndex")
children_count = PnTreeGetNodeCount("TreeView1", index, 1)
```

Die Beschriftungen aller selektierten Knoten, die direkte Kinder eines Wurzelknotens sind, werden in ein Textfeld geschrieben.

```
selected_entries = TxArrayCreate(0)
all_selected_nodes = PnTreeGetNodes("TreeView1", 0, 0, "Selected")
FOREACH SAMPLE node IN all_selected_nodes
IF PnTreeGetNodeState("Treeview1", node , 0, "Level") = 1
```

```
   selected_entries = TxArrayInsert( selected_entries, PnTreeGetNodeProp("TreeView1", node, 0, "Caption"), -1)
END
```

Ein Baum-Widget enthält unter anderem einen Zweig "Data files", in dem in diversen Unter-Verzweigungen Messwertdateien gelistet sind. Beim Doppelklick auf einen Blatt-Knoten wird die zugehörige Datei geladen.

Ereignis-Sequenz 'Doppelgeklickt' für das Baum-Widget:

```
selected_index = PA2
root_index = PnTreeGetNodeState(   PA1, selected_index, 0, "RootNode")
IF PnTreeGetNodeProp(PA1, root_index, 0, "Caption") = "Data files"
   ; OK, der Doppelklick erfolgte innerhalb des gewünschten Zweiges
   filename = PnTreeGetNodeProp(PA1, selected_index, 0, "Caption")
   FileLoad(filename, "", 0)
END
```

Siehe auch:

PnTreeGetNodeProp, PnTreeGetNodeState

Unterstützt ab:

Version 2023

PnTreeInsertNode

Anwendungsbereich: Panele

Fügt einer Baumansicht einen neuen Knoten hinzu.

Deklaration:

```
PnTreeInsertNode ( TxElementName, ParentNode, ChildPosition, TxCaption [, TxKeyOrData] [, TxImageKey] ) -> Index
```

Parameter:

TxElementName	Name des Baumansicht-Widgets.
ParentNode	Absoluter Index oder Schlüssel des Elternknotens, an den der neue Knoten angehängt werden soll. Geben Sie eine 0 an, um den Knoten in der Wurzelebene zu erzeugen.
ChildPosition	Position des neuen Knotens unter den direkten Kindern des angegebenen Elternknotens. Bei Knoten der Wurzelebene die Position innerhalb der Wurzelebene (entspricht der Zeilennummer, wenn alle Knoten zugeklappt sind). Das erste Kind hat den Index 1. Geben Sie eine 0 an, um den neuen Knoten als letztes Kind anzuhängen.
TxCaption	Beschriftung des neuen Knotens.
TxKeyOrData	Zusätzlich am Knoten gespeicherter Text, z.B. als Schlüssel zur späteren Identifikation des Knotens oder zusätzlich vermerkte Daten in Textform. (optional , Standardwert: "")
TxImageKey	Legt das anzuzeigende Miniaturbild für diesen Knoten fest. Das Bild wird links von der Beschriftung angezeigt. Die verfügbaren Bilder müssen zusammen mit einem eindeutigen identifizierenden Schlüssel zur Design-Zeit in der Eigenschaft "Bildliste" der Baumansicht definiert werden. Geben Sie einen leeren Text "" an, wenn kein Bild angezeigt werden soll. (optional , Standardwert: "")
Index	Absoluter Index des neu angelegten Knotens.

Beschreibung:

Um den gewünschten Elternknoten zu adressieren, stehen für den Parameter [ParentNode] zwei verschiedene Möglichkeiten zur Verfügung:

- **Absoluter Index:** Der absolute Index des Knotens unter allen vorhandenen Knoten. Entspricht der Zeilennummer, in der sich der Knoten befindet, wenn alle Knoten aufgeklappt sind. Der erste Knoten der Wurzelebene hat den Index 1.
- **Schlüssel:** Beim Anlegen eines Knotens kann vom Anwender ein optionaler textueller Schlüssel definiert werden, durch den der Knoten dann später wieder angesprochen werden kann. Der vergebene Schlüssel sollte darum unbedingt eindeutig sein. Bezüglich Groß-/Kleinschreibung wird dabei nicht unterschieden.

Parameter [TxKeyOrData]: Der in diesem Parameter angegebene Text kann als Schlüssel verwendet werden, um diesen Knoten später leicht in anderen Funktionen zur Steuerung der Baumansicht spezifizieren zu können. In dieser Anwendung sollte der Schlüssel natürlich eindeutig sein. Bei Schlüsseln wird bezüglich Groß-/Kleinschreibung nicht unterschieden. Alternativ kann der Parameter auch verwendet werden, um zusätzliche Informationen am Knoten zu speichern. Beispielsweise könnte beim Füllen des Baumes mit Dateinamen als Beschriftung nur der "kurze" Dateiname angezeigt werden und im Schlüssel der komplette Dateipfad abgelegt werden.

Beispiele:

Einfügen eines neuen Wurzelknotens an letzter Position in der Wurzelebene mit zwei Kindknoten. Der Wurzelknoten wird aufgeklappt und der zweite Kindknoten selektiert.

```
index = PnTreeInsertNode("TreeView1", 0, 0, "Result data")
PnTreeInsertNode("TreeView1", index, 0, "Highpass10Hz")
PnTreeInsertNode("TreeView1", index, 0, "Highpass20Hz")
PnTreeSetNodeState("TreeView1", index, 0, "Expanded", 1)
PnTreeSetNodeState("TreeView1", index, 2, "Selected", 1)
```

Die Namen der mitgelieferten FAMOS-Beispieldateien werden in einer Baumansicht angezeigt. Alle Dateien mit der Erweiterung "dat" werden unter dem Wurzelknoten "imc files" einsortiert, alle anderen Dateien unter dem Wurzelknoten "Other files". Angezeigt wird der Dateiname, der komplette Pfadname wird als Zusatzinformation gespeichert. Die beiden Wurzelnoten zeigen Miniaturbilder (die beim Design des Panels in der Eigenschaft "Bildliste" der Baumansicht definiert wurden). Der "imc files"-Knoten wird aufgeklappt. Der Knoten "Other files" wurde zusätzlich mit einem Schlüssel versehen, um diesen beim Einfügen seiner Kindknoten einfach adressieren zu können (die Adressierung über seinen absoluten Index wäre aufwendiger, da dieser nicht konstant ist und bei jedem Aufruf erst neu bestimmt werden müsste.).

```
PnTreeDeleteNode("Treeview1", -1, 0); clear content
PnTreeInsertNode("TreeView1", 0, 0, "imc files", "", "ImcFileIcon")
PnTreeInsertNode("TreeView1", 0, 0, "Other files", "#other", "GenericFileIcon")
files = FsGetFileNames("C:\Users\Public\Documents\imc\imc FAMOS\_Demo Projects\_Demo Files\dat", "*.*", 0, 0, 1)
FOREACH ELEMENT path IN files
   ext = FsSplitPath(path, 3)
   file = FsSplitPath(path, 4)
   IF ext = ".dat"
```

```
      PnTreeInsertNode("TreeView1", 1, 0, file, path)
   ELSE
      PnTreeInsertNode("TreeView1", "#other", 0, file, path)
   END
END
PnTreeSetNodeState("TreeView1", 1, 0, "Expanded", 1)
```

Siehe auch:

PnTreeDeleteNode, PnTreeSetNodeProp, PnTreeSetNodeState

Unterstützt ab:

Version 2023

PnTreeSetNodeProp

Anwendungsbereich: Panele

Eine Eigenschaft des angegebenen Knotens wird gesetzt.

Deklaration:

```
PnTreeSetNodeProp ( TxElementName, Node, ChildPosition, TxPropertyID, TxText )
```

Parameter:

TxElementName	Name des Baumansicht-Elements.
Node	Absoluter Index oder eindeutiger Schlüssel des zu verwendenden Knotens bzw. des zugehörigen Elternknotens. Mit einer 0 adressieren Sie einen Knoten der Wurzelebene, der nächste Parameter gibt dann die Position innerhalb der Wurzelebene an.
ChildPosition	Kind-Position
	0 : [Node] identifiziert den zu verwendenden Knoten direkt.
	>=1 : Position des gewünschten Knotens unter den unmittelbaren Kindknoten von [Node] bzw. innerhalb der Wurzelebene, wenn für [Node] eine 0 angegeben wurde.
TxPropertyID	Auswahl der zu setzenden Eigenschaft
	"Caption" : Beschriftung
	"NodeKey" : Knoten-Schlüssel
	"ImageKey" : Bild-Schlüssel
	"ChildCheckStyle" : Markierungs-Stil der Kindknoten
TxText	Neuer Wert. Die Bedeutung ist abhängig von [TxPropertyID]:
	"ChildCheckStyle" : Markierungs-Stil der Kindknoten. Für [Node]=0 und [ChildPosition]=0 wird der Markierungsstil für alle Wurzelknoten gesetzt.
	"Standard" – Die entsprechende globale Einstellung am Widget wird verwendet. **"None"** – Kein Markierungselement anzeigen. **"Check"** – Kontrollkästchen (Checkbox). **"Radio"** – Optionsgruppe (Radiogroup).

Beschreibung:

Beispiele:

Für alle Knoten der Wurzelebene, die bisher kein Miniaturbild haben, wird ein Standardbild zugewiesen. Dieses wurde beim Design der Baumansicht in der Eigenschaft "Bildliste" mit dem Schlüssel "GenericFileIcon" definiert.

```
count = PnTreeGetNodeCount("TreeView1", 0, 1)
FOR i = 1 TO count
   imagekey = PnTreeGetNodeProp("TreeView1", 0, i, "ImageKey")
   IF imageKey = ""
      PnTreeSetNodeProp("TreeView1", 0, i, "ImageKey", "GenericFileIcon")
   END
END
```

Die Beschriftung aller Knoten, die auf ".dat" enden, wird in Großschreibung gewandelt.

```
indizes = PnTreeFindNodes("TreeView1", 0, 0, "*.dat", "Caption*")
FOREACH SAMPLE index IN indizes
   caption = PnTreeGetNodeProp("TreeView1", index, 0, "Caption")
   PnTreeSetNodeProp("TreeView1", index, 0, "Caption", TConv(caption, 2))
END
```

Siehe auch:

PnTreeInsertNode, PnTreeGetNodeProp, PnTreeSetNodeState

Unterstützt ab:

Version 2023

PnTreeSetNodeState

Anwendungsbereich: Panele

Der aktuelle Zustand eines Baumknotens wird geändert.

Deklaration:

```
PnTreeSetNodeState ( TxElementName, Node, ChildPosition, TxStateID, StateValue )
```

Parameter:

TxElementName	Name des Baumansicht-Elements.
Node	Absoluter Index oder Schlüssel des gewünschten Knotens oder des zugehörigen Elternknotens. Die Angabe von 0 bedeutet, dass ein Knoten der Wurzelebene angesprochen werden soll, der nächste Parameter gibt dann die gewünschte Position innerhalb der Wurzelebene an. Die Angabe von -1 bedeutet, dass die Operation auf alle Baumknoten abgewendet werden soll. Der nächste Parameter ist dann auf 0 zu setzen.
ChildPosition	Kind-Position
	0 : Nicht verwendet. [Node] identifiziert den zu verwendenden Knoten direkt. Wenn für [Node] eine 0 angegeben wurde, wird die Operation auf alle Wurzelknoten angewendet.
	-1 : Alle Kindknoten. Die Operation wird auf alle Kindknoten von [Node] angewendet. Wenn für [Node] eine 0 angegeben wurde, wird die Operation auf alle direkten Kinder der Wurzelnknoten angewendet.
	>=1 : Position des gewünschten Knotens unter den unmittelbaren Kindknoten von [Node] bzw. innerhalb der Wurzelebene, wenn für [Node] eine 0 angegeben wurde.
TxStateID	Welche Zustands-Eigenschaft soll geändert werden?
	"Selected" : Die Selektion für den gewählten Knoten wird ein- oder ausgeschaltet. Das explizite Ausschalten ist nur bei Bäumen mit Multiselektion erlaubt.
	"Checked" : Der Markiert-Zustand des Knotens wird ein- oder ausgeschaltet.
	"Expanded" : Der Knoten wird auf- oder zugeklappt.
	"Selected>" : Die Selektion für den gewählten Knoten und alle seine Kindknoten wird ein- oder ausgeschaltet. Nur für Bäume mit Multiselektion zulässig.
	"Checked>" : Der Markiert-Zustand des Knotens und aller seiner Kindknoten wird ein- oder ausgeschaltet.
	"Expanded>" : Der Knoten und alle seine Kindknoten werden auf- oder zugeklappt.
	"Visible" : Es wird sichergestellt, dass der Knoten für den Anwender sichtbar ist. Dazu werden ggf. alle Elternknoten aufgeklappt und der Knoten in den sichtbaren Bereich gerollt. [StateValue] muss den Wert 1 haben.
StateValue	Neuer Zustand
	0 : Die gewählte Eigenschaft wird ausgeschaltet.
	1 : Die gewählte Eigenschaft wird eingeschaltet.

Beschreibung:

Beispiele:

Alle Knoten in einer Baumansicht zusammenklappen:

```
PnTreeSetNodeState("TreeView1", 0, 0, "Expanded>", 0)
```

Markieren aller direkten Kindknoten des Knotens mit dem Schlüssel "#files". Der Knoten wird dann für den Anwender sichtbar gemacht (ggf. durch Aufklappen aller Elternknoten und Rollen in den sichtbaren Bereich).

```
PnTreeSetNodeState("TreeView1", "#files", -1, "Checked", 1)
PnTreeSetNodeState("TreeView1", "#files", 0, "Visible", 1)
```

Einfügen eines neuen Wurzelknotens an letzter Position in der Wurzelebene mit 2 Kindknoten. Der Wurzelknoten wird aufgeklappt und der 2. Kindknoten selektiert.

```
index = PnTreeInsertNode("TreeView1", 0, 0, "Result data")
PnTreeInsertNode("TreeView1", index, 0, "Highpass10Hz")
PnTreeInsertNode("TreeView1", index, 0, "Highpass20Hz")
PnTreeSetNodeState("TreeView1", index, 0, "Expanded", 1)
PnTreeSetNodeState("TreeView1", index, 2, "Selected", 1)
```

Siehe auch:

PnTreeSetNodeProp, PnTreeGetNodeState

Unterstützt ab:

Version 2023

Pol

Transformation eines komplexen Datensatzes in Polarkoordinaten (Betrag/Phase).

Deklaration:

```
Pol ( KomplexeDaten ) -> KomplexAlsBP
```

Parameter:

KomplexeDaten	Zu transformierender komplexer Datensatz. [BP], [DP] oder [RI].
KomplexAlsBP	Resultierender komplexer Datensatz in Polarkoordinaten. [BP]

Beschreibung:

Ein komplexer Datensatz wird in Polarkoordinaten transformiert, d. h. eine Darstellung mit Betrag und Phase. Liegt der Datensatz bereits in dieser Form vor, bleibt er unverändert. Liegt der Datensatz in der Form Betrag in dB und Phase vor, wird der Betrag wieder linear berechnet.

- Falls der übergebene Datensatz in kartesischen Koordinaten vorliegt, wird die Phase in Grad berechnet.
- Liegt der zu transformierende Datensatz als DP vor, hat die Funktion die gleiche Wirkung wie idB().
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Ein Spektrum wird in die meist anschaulichere Darstellungsart BP transformiert:

```
MPspectrum = Pol(RIspectrum)
```

Ein Spektrum, das nach mehreren Berechnungen Phasen außerhalb von -180° ... +180° hat oder Beträge kleiner Null, wird bereinigt. Vom Inhalt her wird das Spektrum nicht verändert:

```
MPcorr = Pol(Rect(BPspectrum))
```

Siehe auch:

Rect, dB, idB, Compl

Poly

Polynom-Approximation der Werte eines Datensatzes

Deklaration:

```
Poly ( Daten, EwOrdnung, EwOption ) -> ErgebnisPolynom
```

Parameter:

Daten	Datensatz, der durch ein Polynom approximiert werden soll. Erlaubte Datentypen: [ND], [XY]
EwOrdnung	Ordnung des approximierenden Polynoms
EwOption	Definiert die Art der Berechnung
	1 : Ausgabe der Funktionswerte
	2 : Ausgabe der Koeffizienten
ErgebnisPolynom	Funktionswerte bzw. Koeffizienten des approximierenden Polynoms

Beschreibung:

Es wird eine Polynom-Approximation des angegebenen Datensatzes durchgeführt. Dabei wird der Datensatz durch ein Polynom der eingegebenen Ordnung approximiert, wobei die Methode der kleinsten Quadrate angewendet wird. Das approximierende Polynom kann eine Ordnung m von 1 bis 9 (bei XY-Daten von 1 bis 7) annehmen. Die Ordnung m wird durch den Parameter [EwOrdnung] eingegeben.

Die Vorgehensweise bei der Funktion Poly wird folgend erläutert: Sei ein Messdatensatz mit den Werten g(x[i]) (i = 0, ..., Datensatzlänge - 1) gegeben. Der Messdatensatz soll durch ein Polynom m-ter Ordnung approximiert werden. Mit Hilfe der Funktion Poly() werden die Koeffizienten k[0], ..., k[m] nach der Methode der kleinsten Quadrate so bestimmt, dass die Bedingung

$$g(x[i]) \approx k[0] + k[1] \cdot x[i] + k[2] \cdot x[i]^2 + \ldots + k[m] \cdot x[i]^m$$

für alle i möglichst gut erfüllt wird.

Durch geeignete Eingabe des Parameters [EwOption] werden entweder der Datensatz des approximierenden Polynoms oder der Datensatz der Koeffizienten des approximierenden Polynoms ausgegeben.

Bei Eingabe von 1 für den Parameter [EwOption] werden die Funktionswerte des approximierenden Polynoms ausgegeben. Für den Datentyp [ND] hat die zurückgegebene Kurve die gleiche Auflösung wie der Quelldatensatz. Beim Datentyp [XY] hat die zurückgegebene Kurve 1000 Punkte und ist über denselben X-Bereich wie der Quelldatensatz definiert.

Bei Eingabe von 2 für den Parameter [EwOption] werden die Koeffizienten des approximierenden Polynoms ausgegeben. Die den aufsteigenden Potenzen des Polynoms zugehörigen Koeffizienten werden nacheinander ausgegeben. Die Konstante wird zuerst ausgegeben. Der zur größten Potenz des Polynoms gehörige Koeffizient wird zuletzt ausgegeben.

- Der Parameter [EwOrdnung] ist als ganze Zahl einzugeben. Die Ordnung muss mindestens 1 und darf höchstens 9 (Datentyp ND) bzw. 7 (XY) betragen.
- Die Länge des zu approximierenden Datensatzes muss mindestens 2 betragen.
- Die Funktion Poly() ist ein Spezialfall der Funktion Appro().

Abhängigkeit der Numerik von den x-Werten:

Für die Numerik ist es am besten, wenn alle x-Werte möglichst dicht um 0 verteilt sind. Das Ergebnis wird numerisch um so schlechter, je weiter die Mehrzahl der x-Werte von 0 entfernt sind.

Für eine grobe Abschätzung kann der folgende Ausdruck bestimmt werden:

```
A1 = X0 ;bzw. der kleinste x-Wert bei XY-Daten
A2 = X0 + DeltaX * (Datensatzlänge-1) ;bzw. der größte x-Wert bei XY-Daten
A  = |A1 + A2| / (2*(A2-A1))
```

Je größer A ist, um so ungenauer wird das Ergebnis.

Als Faustregel gelten für A die folgenden groben Grenzen (abhängig von der Ordnung, bei XY-Daten unter der Annahme von etwa gleichverteilten x-Koordinaten):

Ordnung	A
1	A <= 100000
2	A <= 100
3	A <= 10
4	A <= 5
5	A <= 3
6	A <= 2

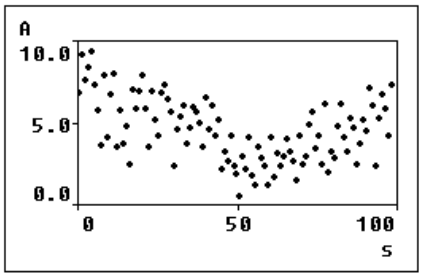
Genauigkeit

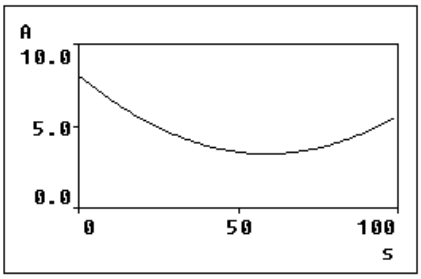
Bei der Anwendung der Methode der kleinsten Quadrate werden die Produkte (k* x) gegeneinander aufgewogen und die k so bestimmt, dass der quadratische Fehler der Gleichung minimiert wird. Dabei kann es vorkommen, dass einige dieser Produkte keinen nennenswerten Beitrag zur Summe über alle Produkte liefern. Dann bedeutet das oft ein sehr kleines k, was allerdings auch nur ungenau bestimmt werden kann. Intern wird zwar mit 15 Stellen hinter dem Komma gerechnet, aber es liegt am Wesen der numerischen Methode, dass nicht relevante Größen mit stärkeren Ungenauigkeiten behaftet sind. Auch wenn Parameter k dieselbe Größenordnung haben, ist nicht der Wert von k, sondern das Produkt (k*x) für diese Betrachtung entscheidend. Beispielsweise sind k1 = k2 = 1, aber (k1*x) liegt um 10, während (k2* x *x) um 1000 liegt. Damit kann dann angenommen werden, dass bei der Summation der beiden Produkte das erste eine um den Faktor 100 geringere Rolle spielt, weshalb der Fehler von k1 dann auch etwa 100mal größer sein kann als der von k2.

Beispiele:

```
NDfktwerte = Poly(NDdaten, 2, 1)
NDkoeff = Poly(NDdaten, 2, 2)
```

Der zur linken Abbildung gehörige Datensatz NDdaten soll durch ein Polynom 2-ter Ordnung approximiert werden. Der Datensatz NDdaten besteht aus 100 Werten. Bei Eingabe von 1 für den Parameter [EwOption] (Ausgabe der Funktionswerte des approximierenden Polynoms) wird in imc FAMOS als Ergebnis der Polynomapproximation der auf der rechten Abbildung dargestellte Datensatz NDfktwerte (Funktionswerte) ausgegeben:





Bei Eingabe von 2 für den Parameter [EwOption] (Ausgabe der Koeffizienten des approximierenden Polynoms) wird in imc FAMOS als Ergebnis der Datensatz NDkoeff (Koeffizienten) ausgegeben. Im Datensatz NDkoeff befinden sich die 3 Werte

```
koeff0 = 8.0577 , koeff1 = -162.87E-3 und koeff2 = 1.3804E-3
```

So erhalten Sie die einzelnen Koeffizienten:

```
EWkoeff0 = NDkoeff[1]
EWkoeff1 = NDkoeff[2]
EWkoeff2 = NDkoeff[3]
```

Siehe auch:

Appro, ApproNonLin, eFit, LFit, Value

PolynomRoots

Verfügbar ab: Professional Edition

Nullstellen bzw. Wurzeln eines Polynoms

Deklaration:

```
PolynomRoots ( Polynom ) -> Nullstellen
```

Parameter:

Polynom	Polynom
Nullstellen	Nullstellen

Beschreibung:

Das Ergebnis ist die Liste der Nullstellen. Das Polynom hat soviele Nullstellen wie sein Grad angibt.

Die Nullstellen werden als komplexe Zahlen ermittelt und in der Form Realteil/Imaginärteil zurückgegeben. Bei reeller Nullstelle ist der Imaginärteil 0.

Deutung eines Polynoms mit den Koeffizienten p[i], wobei die Koeffizienten im Datensatz nach aufsteigender Potenz sortiert sind:

```
Polynom = p[1] + p[2] * x + p[3] * x^2 + p[4] * x^3 +...
```

Der maximal unterstützte Grad ist 20.

Das Polynom muss reell (nicht komplex) sein.

Die Funktion findet die Nullstellen über die Eigenwerte der Begleitmatrix.

Beispiele:

Nullstellen einer quadratischen Gleichung bzw. eines Polynoms 2. Grades

```
P = [ 24, -14, 2]  ; test data, 24-14x+2x^2=0
z = PolynomRoots ( P )
z_r = z.r
; z = [3, 4]
```

komplexe Nullstellen einer quadratischen Gleichung bzw. eines Polynoms 2. Grades

```
P = [ 58, -8, 2]  ; test data, 58-8x+2x^2=0
z = PolynomRoots ( P )
; z.r = [2, 2]
; z.i = [5, -5]
; 2+5i, 2-5i
```

Siehe auch:

Alle0

Pos

Liefert die erste Position (X-Koordinate) eines gegebenen Y-Wertes
Alternativer Name: **Posi**

Deklaration:

```
Pos ( Daten, EwLevel ) -> EwPosition
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND],[XY].
EwLevel	Y-Wert, dessen X-Koordinate bestimmt werden sollen.
EwPosition	Die ermittelte X-Koordinate

Beschreibung:

In einem Datensatz wird zu einem vorgegebenen y-Wert die zugehörige x-Koordinate (x-Position) bestimmt. Der Datensatz wird zwischen seinen Abtastwerten linear interpoliert gedacht. Damit braucht der übergebene y-Wert keinen Abtastwert zu treffen. Die ermittelte x-Koordinate liegt deshalb i. a. auch nicht genau bei einem Abtastwert.

Die Funktion ermittelt also die Position eines Wertes präziser als die Abtastschritte sind.

Tritt ein y-Wert mehrfach in einem Datensatz auf, wird nur das erste Auftreten bestimmt. Um jedes Auftreten eines y-Wertes zu finden, können Sie die Funktion PosiEx2() verwenden.

Die Einheit des zurückgelieferten Wertes ist die x-Einheit des Datensatzes.

Die Einheit des übergebenen y-Wertes sollte sinnvollerweise die y-Einheit des Datensatzes haben.

Liegt der übergebene y-Wert nicht im Wertebereich des Datensatzes, d. h. es kann keine x-Position zugeordnet werden, wird eine Warnung erzeugt und der zurückgegebene Wert auf -10^20 gesetzt. Zur Ermittlung des zulässigen Bereichs können die Funktionen Min und Max benutzt werden.

Beispiele:

Die erste Nullstelle eines Datensatzes wird bestimmt:

```
pos1stZero = Pos(NDdata, 0)
```

Ein Spannungsanstieg von 0 auf SVmax wurde gemessen. Die Anstiegszeit ist definiert als die Zeitspanne zwischen dem Auftreten der Werte 0.9 * SVmax und 0.1 * SVmax. Die Anstiegszeit kann mit Hilfe der Funktion Pos() ermittelt werden:

```
slopeTime = Pos(NDvoltage, 0.9 * SVmax) - Pos(NDvoltage, 0.1 * SVmax)
```

Siehe auch:

PosiEx2, PosiEx, Value2, SearchLevel

PosiEx

Liefert die Positionen (X-Koordinaten) zu einem gegebenen Y-Wert

Deklaration:

```
PosiEx ( Daten, EwLevel ) -> Positionen
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND],[XY].
EwLevel	Y-Wert, dessen X-Koordinaten bestimmt werden sollen.
Positionen	Die ermittelten X-Koordinaten

Beschreibung:

In einem Datensatz werden zu einem vorgegebenen y-Wert die zugehörigen x-Koordinaten (x-Positionen) bestimmt. Der Datensatz wird zwischen seinen Abtastwerten linear interpoliert gedacht. Damit braucht der übergebene y-Wert keinen Abtastwert zu treffen. Die ermittelten x-Koordinaten liegen deshalb i. Allg. auch nicht genau bei einem Abtastwert.

Die Funktion ermittelt also die Positionen eines Wertes präziser als die Abtastschritte sind.

Die Y-Einheit des zurückgelieferten Datensatzes ist die x-Einheit des Parameter-Datensatzes.

Die Einheit des übergebenen y-Wertes sollte sinnvollerweise die y-Einheit des Datensatzes haben.

Liegt der übergebene y-Wert nicht im Wertebereich des Datensatzes, d. h. es kann keine x-Position zugeordnet werden, wird eine Warnung erzeugt und ein leerer Datensatz (Länge 0) zurückgegeben.

Die Funktion ist veraltet, die leistungsfähigere Funktion PosiEx2() ist im Allgemeinen vorzuziehen.

Beispiele:

Es werden alle Nullstellen eines Datensatzes bestimmt und dann die Position der letzten Nullstelle ermittelt:

```
zeroes = PosiEx(NDdata, 0)
lastZero = zeroes[Leng?(zeroes)]
```

Siehe auch:

PosiEx2, Pos, Value2, SearchLevel

PosiEx2

Liefert die Positionen zu einem gegebenen Y-Wert

Deklaration:

```
PosiEx2 ( Daten, EwLevel, EwFlanke, EwRückgabe ) -> Positionen
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND],[XY].
EwLevel	Y-Wert, dessen x-Koordinaten bestimmt werden sollen.
EwFlanke	Zusätzliche Bedingung für die Richtung, mit der der gesuchte Pegel durchlaufen wird.
	0 : Keine Bedingung.
	1 : Positive Flanke (vom Punkt weg). Ein Wert EWLevel liegt auf der Flanke, wenn gilt: y[k] <= EWLevel < y[k+1]
	2 : Positive Flanke (zum Punkt hin). y[k] < EWLevel <= y[k+1]
	3 : Negative Flanke (vom Punkt weg). y[k] >= EWLevel > y[k+1]
	4 : Negative Flanke (zum Punkt hin). y[k] > EWLevel >= y[k+1]
EwRückgabe	Steuert, in welcher Form die gefundene Position zurückgegeben wird. Das Eingangs-Signal wird linear interpoliert und daraus zunächst die entsprechenden interpolierten x-Positionen bestimmt. Zurückgegeben wird dann:
	0 : x, Interpoliert: Der linear interpolierte x-Wert wird verwendet.
	1 : x, Aktuell: Es wird der x-Wert derjenigen Stützstelle des Signals verwendet, die direkt VOR dem interpolierten x-Wert liegt.
	2 : x, Nächstliegend: Es wird der x-Wert derjenigen Stützstelle des Signals verwendet, die am NÄCHSTEN zum interpolierten x-Wert liegt.
	3 : Index, Interpoliert: Der (interpolierte) Index des gefundenen Wertes.
	4 : Index, Aktuell: Es wird der Index derjenigen Stützstelle des Signals verwendet, die direkt VOR dem interpolierten x-Wert liegt.
	5 : Index, Nächstliegend: Es wird der Index derjenigen Stützstelle des Signals verwendet, die am NÄCHSTEN zum interpolierten x-Wert liegt.
Positionen	Die ermittelten X-Koordinaten bzw. Indizes.

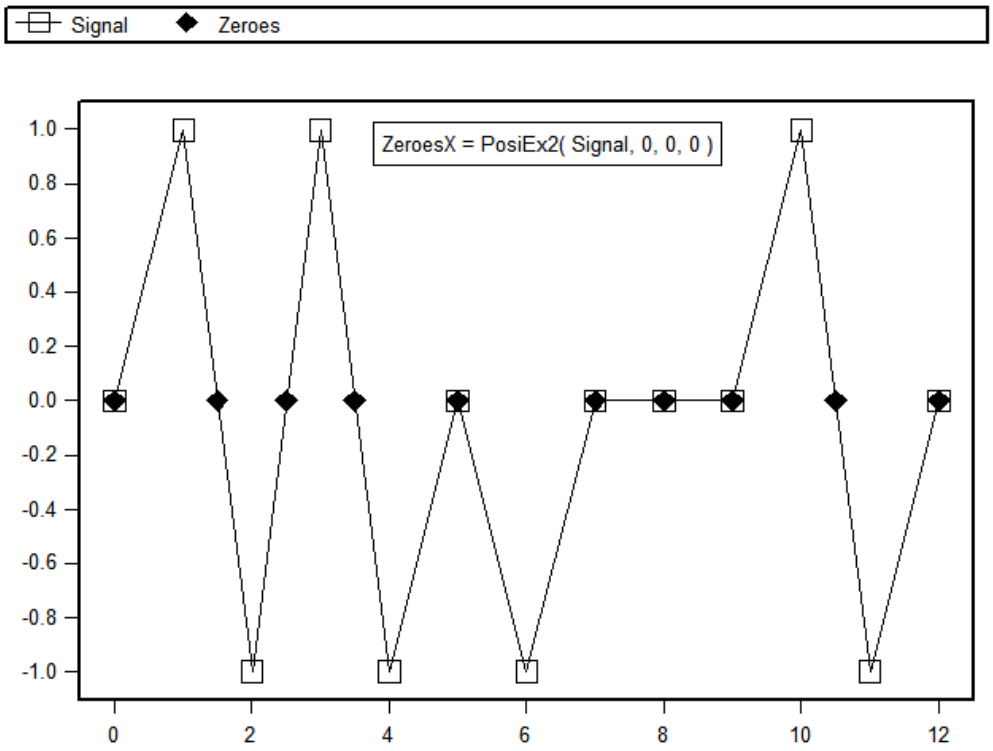
Beschreibung:

In einem Datensatz werden zu einem vorgegebenen y-Wert die zugehörigen x-Koordinaten bzw. Indizes bestimmt. Der Datensatz wird zwischen seinen Abtastwerten linear interpoliert gedacht. Damit braucht der übergebene y-Wert keinen Abtastwert zu treffen.

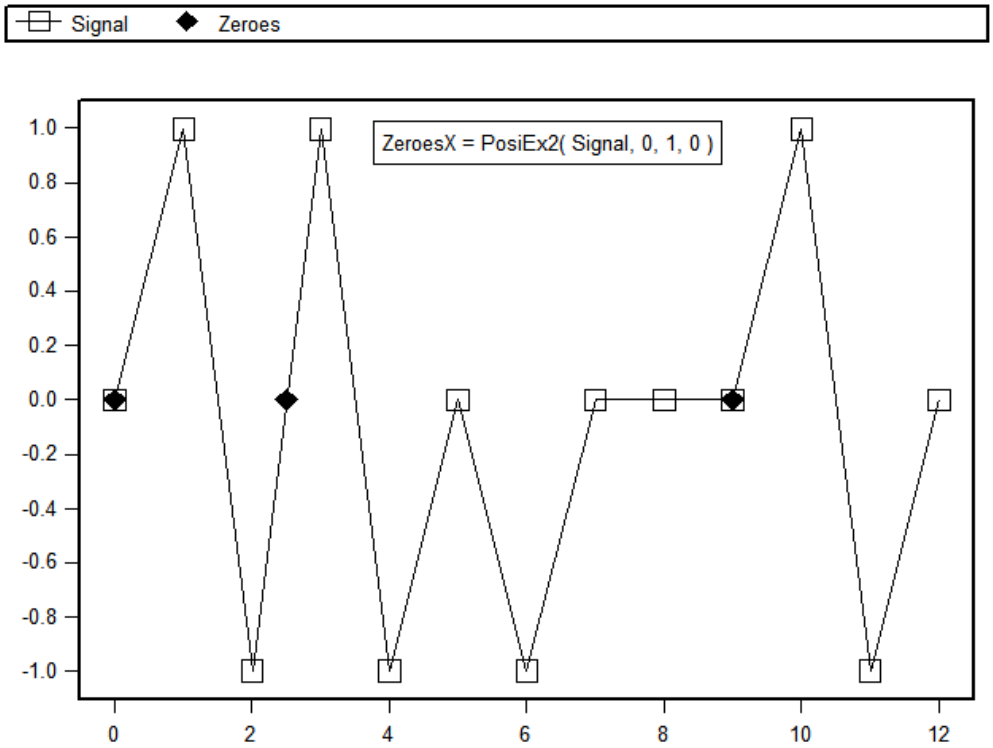
Liegt der übergebene y-Wert nicht im Wertebereich des Datensatzes, d. h. es kann keine Position zugeordnet werden, wird eine Warnung erzeugt und ein leerer Datensatz (Länge 0) zurückgegeben.

Flankenbedingung:
Der Unterschied zwischen den Optionen 1 und 2 bzw. 3 und 4 liegt in der Definition der Flanke, wenn ein Punkt im Signal genau den gesuchten y-Wert trifft und keine eindeutige Flankenzuordnung möglich ist. Dies betrifft den ersten und letzten Punkt des Signals sowie alle Punkte, in denen der Anstieg des Signals wechselt.
Die Optionen 1/3 bedeuten, dass der Anstieg zwischen diesem Punkt und dem nächsten Punkt positiv/negativ sein muss. Der letzte Punkt des Signals ist damit niemals Bestandteil des Ergebnisses.
Die Optionen 2/4 bedeuten, dass der Anstieg vom vorherigen Punkt zu diesem Punkt positiv/negativ sein muss. Der erste Punkt des Signals ist damit niemals Bestandteil des Ergebnisses.

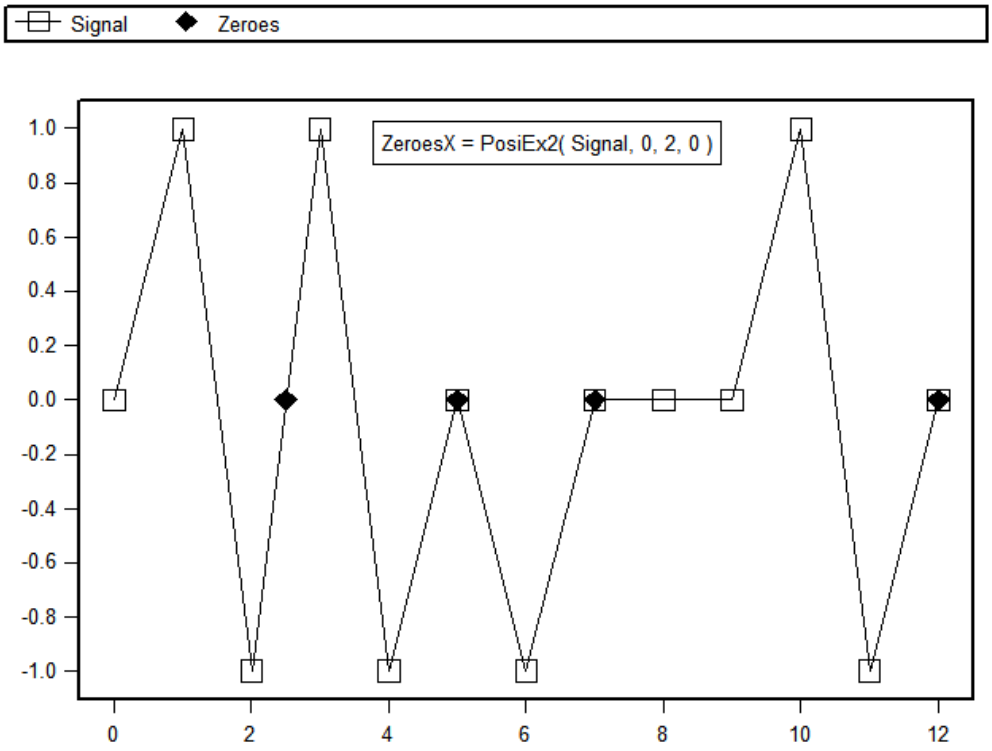
Im folgenden Beispiel werden zunächst alle Nullstellen im Signal bestimmt (EwFlanke = 0):



Nun sollen alle Nulldurchgänge an positiven Flanken gefunden werden. Mit EwFlanke = 1 wird der Anstieg von der Nullstelle 'weg' betrachtet. Die Nullstelle am Ende des Signals wird somit nicht berücksichtigt.



Mit EwFlanke = 2 wird der Anstieg zur Nullstelle 'hin' betrachtet. Die Nullstelle am Anfang des Signals wird damit nicht berücksichtigt.



Beispiele:

Bestimmung aller Nullstellen eines Signals. Rückgabe der interpolierten x-Werte:

```
xZeroes = PosiEx2(Signal, 0, 0, 0)
```

Um die Nullstellen zusammen mit dem Signal darzustellen, wird zunächst ein XY-Datensatz für die Nullstellen erzeugt und dieser dann mit Symbol=Kreis im selben Kurvenfenster wie das Originalsignal angezeigt.

```
Zeroes = XYof(xZeroes, Value2(Signal, xZeroes, 0))
CwNewWindow(Signal, "show")
CwNewChannel("append new axis", Signal)
CwNewChannel("append last axis", Zeroes)
CwSelectByIndex("line", 2)
CwLineSet("type", 0)
CwLineSet("symbol", 2)
```

Aus einem Signal soll der Bereich zwischen den ersten beiden Nullstellen ausgeschnitten werden. Die beiden Nullstellen sollen sicher enthalten sein.

```
zeroes = PosiEx2(signal, 0, 0, 4)
IF Leng?(zeroes) > 1
   front = zeroes[1]
   back = zeroes[2]
   IF signal[back] <> 0
      back = back + 1 ; Index hinter die Nullstelle verschieben
   END
   signalPart = CutIndex(signal, front, back)
END
```

Bestimmung aller Positionen in einem Signal, bei dem der Pegel 10 bei positiver Flanke von unten kommend erreicht wird. Rückgabe des Index des am nächsten liegenden Punktes.

```
xZeroes = PosiEx2(Signal, 10, 2, 5)
```

Bestimmung aller Positionen in einem Signal, bei dem der Pegel 10 bei positiver Flanke nach oben hin verlassen wird. Rückgabe des Index des jeweils davor liegenden Punktes.

```
xZeroes = PosiEx2(Signal, 10, 1, 4)
```

Siehe auch:

Pos, PosiEx, Value2, SearchLevel

Power1

Verfügbar ab: Professional Edition

Berechnung der Leistung an einer Phase

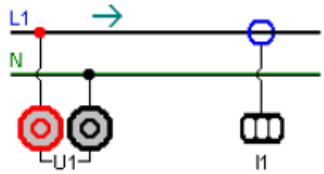
Deklaration:

```
Power1 ( U1, I1 [, Intervalldaten] ) -> Ergebnisgruppe
```

Parameter:

U1	U1 = UL1 - N, Spannung der Phase gegen den Neutralleiter
I1	I1, auch IL1, Strom durch den Leiter, Stromrichtung (Pfeil) von der Quelle zum Verbraucher
Intervalldaten	Intervalldaten, die Intervalle stellen die Perioden des Signals dar. Möglichst eine Folge von lückenlos aneinander grenzenden Intervallen. Die Berechnung erfolgt über diese Perioden. Vor allem bei unbekannter oder sich verändernder Signalfrequenz. (optional)
Ergebnisgruppe	Gruppe, die alle Rechenergebnisse enthält: P Wirkleistung, PF Leistungsfaktor, U* Effektivwerte der Spannungen, I* Effektivwerte der Ströme, p_t Momentanleistung, f Frequenz

Beschreibung:



Die Funktionen PowerSelect() und PowerParameter() müssen vorher aufgerufen werden, um die Funktion komplett zu parametrieren.

Wird der Parameter Intervalldaten nicht angegeben, erfolgt die Berechnung mit der Frequenz, die in PowerParameter() angegeben wurde. Die Anzahl von Signalperioden, über die die Berechnung erfolgen soll, muss multipliziert mit der Periodendauer auf ein ganzes Vielfaches der Abtastzeit führen. So werden bei 50Hz oft 1 oder 10 Perioden angegeben, bei 60Hz dann 12 Perioden, Abtastfrequenz typisch 10kHz.

Wird der Parameter Intervalldaten angegeben, geben die Längen der Intervalle die Periodenlängen vor. Gibt es Lücken vor/zwischen/hinter den Intervallen, werden zur Berechnung kontinuierlich aneinander grenzende Perioden angenommen, die Intervallgrenzen werden dann mitunter nicht mehr benutzt, nur noch ihre Breiten. Start der Berechnung mit dem ersten Intervall oder auch einer ganzen Anzahl von Intervallbreiten vorher. Falls kein Intervall vorhanden, wird die Frequenz aus PowerParameter() benutzt.

Der Intervalldatensatz kann z.B. mit Hilfe von IntervalFromLevel() ermittelt und mit anderen Intervallfunktionen nachbearbeitet werden. Als Grundlage zur Bestimmung der Nulldurchgänge ist ein besonders sinusförmiges und gleichmäßiges Signal wie etwa eine Spannung empfohlen.

Intervalformat: Äquidistant "equi" oder XY "xy", d.h. Intervall-Code mit Position

Die Frequenz wird nur bestimmt, wenn ein Intervalldatensatz angegeben ist. Dann zeigt sie stets die der Berechnung zugrunde gelegte Frequenz an, auch an Stellen fehlender Intervalle.

Wenn mit Intervalldaten gearbeitet wird, ist eine möglichst präzise Bestimmung der Nullstellen erforderlich, um genaue Ergebnisse zu erhalten.

Die Frequenzbestimmung und auch die anderen Ergebnisse werden (deutlich) genauer, wenn die Analyse über mehrere Perioden durchgeführt wird.

Für eine genaue Analyse ist bei einer Netzfrequenz von 50Hz eine Abtastfrequenz von 10kHz sinnvoll.

Stromrichtung entsprechend Verbrauchersystem. War ein Amperemeter anderes herum geschaltet, wird z.B. anstelle von I1 als Parameter -I1 übergeben.

Anstelle des Paares U1, I1 können auch U2, I2 oder U3, I3 eines 3-Phasennetzes einzeln analysiert werden.

Strom und Spannung müssen dieselbe Zeitbasis haben. Sie dürfen Events haben. Segmente sind auch möglich, aber nicht in Zusammenhang mit Intervalldaten.

Die Berechnung liefert nur dann präzise Ergebnisse, wenn die Intervallbreite der Analyse eine ganze Anzahl von Samples beträgt.

Beispiele:

1-phasige Leistungsmessung. Standardberechnung bei 50Hz über 10 Perioden, Abtastfrequenz 10kHz

```
PowerSelect()
PowerParameter(50, 10)
g = Power1 ( U1, I1 )
P = g:P
Q = g:Q
```

1-phasige Leistungsmessung. Frequenz schwankt

```
PowerSelect()
PowerParameter ( 50, 1)
ivl = IntervalFromLevel ( U1, 0, 0, 0, 1, 0.0, 1e35, 0, "")
```

```
g = Power1 ( U1, I1, ivl )
P = g:P
Q = g:Q
```

Alle 3 Phasen des Drehstromnetzes einzeln und zusammen analysieren, die Effektivwerte der Ströme und Spannungen werden nur einmal berechnet.

```
PowerSelect()
PowerParameter(50, 10)
g = Power3 ( U1, I1, U2, I2, U3, I3, ivl )
PowerSelect(1, 0, 1, 0, 0, 1)
g1 = Power1 ( U1, I1, ivl )
g2 = Power1 ( U2, I2, ivl )
g3 = Power1 ( U3, I3, ivl )
P = g:P
Q = g:Q
P1 = g1:P1
```

Siehe auch:

PowerSelect, PowerParameter, Power2, Power3, IntervalFromLevel

Power2

Verfügbar ab: Professional Edition

Berechnung der Leistung am Dreiphasennetz mit Dreileitersystem

Deklaration:

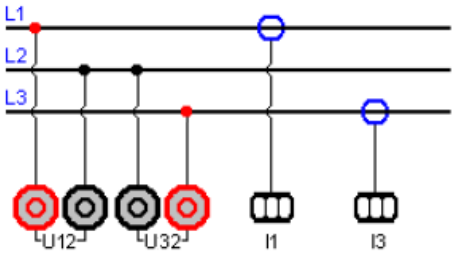
```
Power2 ( U12, I1, U32, I3 [, Intervalldaten] ) -> Ergebnisgruppe
```

Parameter:

U12	U12 = UL1 - UL2, Dreieckspannung, Spannung zwischen den Außenleitern
I1	I1, auch IL1, Strom durch die Phase, Stromrichtung (Pfeil) von der Quelle zum Verbraucher
U32	U32 = UL3 - UL2, Dreieckspannung, Spannung zwischen den Außenleitern
I3	I3, auch IL3, Strom durch die Phase, Stromrichtung (Pfeil) von der Quelle zum Verbraucher
Intervalldaten	Intervalldaten, die Intervalle stellen die Perioden des Signals dar. Möglichst eine Folge von lückenlos aneinander grenzenden Intervallen. Die Berechnung erfolgt über diese Perioden. Vor allem bei unbekannter oder sich verändernder Signalfrequenz. (optional)
Ergebnisgruppe	Gruppe, die alle Rechenergebnisse enthält: P Wirkleistung, PF Leistungsfaktor, U* Effektivwerte der Spannungen, I* Effektivwerte der Ströme, p_t Momentanleistung, f Frequenz

Beschreibung:

Anstelle von [U12, I1, U32, I3] können auch die Paare [U13, I1, U23, I2] oder [U21, I2, U31, I3] benutzt werden.



Der Neutralleister N ist ncht vorhanden oder führt keinen Strom. Typisch ist die Dreieckschaltung. Es gilt I1 + I2 + I3 = 0.

Ferner gilt U12 = -U21, U32 = -U32, U31 = -U13 und U12 + U23 + U31 = 0.

Die Funktionen PowerSelect() und PowerParameter() müssen vorher aufgerufen werden, um die Funktion komplett zu parametrieren.

Wird der Parameter Intervalldaten nicht angegeben, erfolgt die Berechnung mit der Frequenz, die in PowerParameter() angegeben wurde. Die Anzahl von Signalperioden, über die die Berechnung erfolgen soll, muss multipliziert mit der Periodendauer auf ein ganzes Vielfaches der Abtastzeit führen. So werden bei 50Hz oft 1 oder 10 Perioden angegeben, bei 60Hz dann 12 Perioden, Abtastfrequenz typisch 10kHz.

Wird der Parameter Intervalldaten angegeben, geben die Längen der Intervalle die Periodenlängen vor. Gibt es Lücken vor/zwischen/hinter den Intervallen, werden zur Berechnung kontinuierlich aneinander grenzende Perioden angenommen, die Intervallgrenzen werden dann mitunter nicht mehr benutzt, nur noch ihre Breiten. Start der Berechnung mit dem ersten Intervall oder auch einer ganzen Anzahl von Intervallbreiten vorher. Falls kein Intervall vorhanden, wird die Frequenz aus PowerParameter() benutzt.

Der Intervalldatensatz kann z.B. mit Hilfe von IntervalFromLevel() ermittelt und mit anderen Intervallfunktionen nachbearbeitet werden. Als Grundlage zur Bestimmung der Nulldurchgänge ist ein besonders sinusförmiges und gleichmäßiges Signal wie etwa eine Spannung empfohlen.

Intervalformat: Äquidistant "equi" oder XY "xy", d.h. Intervall-Code mit Position

Die Frequenz wird nur bestimmt, wenn ein Intervalldatensatz angegeben ist. Dann zeigt sie stets die der Berechnung zugrunde gelegte Frequenz an, auch an Stellen fehlender Intervalle.

Wenn mit Intervalldaten gearbeitet wird, ist eine möglichst präzise Bestimmung der Nullstellen erforderlich, um genaue Ergebnisse zu erhalten.

Die Frequenzbestimmung und auch die anderen Ergebnisse werden (deutlich) genauer, wenn die Analyse über mehrere Perioden durchgeführt wird.

Für eine genaue Analyse ist bei einer Netzfrequenz von 50Hz eine Abtastfrequenz von 10kHz sinnvoll.

Stromrichtung entsprechend Verbrauchersystem. War ein Amperemeter anderes herum geschaltet, wird z.B. anstelle von I1 als Parameter -I1 übergeben.

Strom und Spannung müssen dieselbe Zeitbasis haben. Sie dürfen Events haben. Segmente sind auch möglich, aber nicht in Zusammenhang mit Intervalldaten.

Die Berechnung liefert nur dann präzise Ergebnisse, wenn die Intervallbreite der Analyse eine ganze Anzahl von Samples beträgt.

Beispiele:

Leistungsmessung am Drehstromnetz mit Aron-Schaltung.

```
PowerSelect()
```

```
PowerParameter(50, 10)
g = Power2 ( U12, I1, U32, I3 )
P = g:P
Q = g:Q
```

Leistungsmessung am Drehstromnetz mit Aron-Schaltung. Frequenz schwankt

```
PowerSelect ()
PowerParameter ( 50, 1)
ivl = IntervalFromLevel ( U12, 0, 0, 0, 1, 0.0, 1e35, 0, "")
g = Power2 ( U12, I1, U32, I3, ivl )
P = g:P
Q = g:Q
```

Siehe auch:

PowerSelect, PowerParameter, Power1, Power3, IntervalFromLevel

Power3

Verfügbar ab: Professional Edition

Berechnung der Leistung am Dreiphasennetz mit Vierleitersystem

Deklaration:

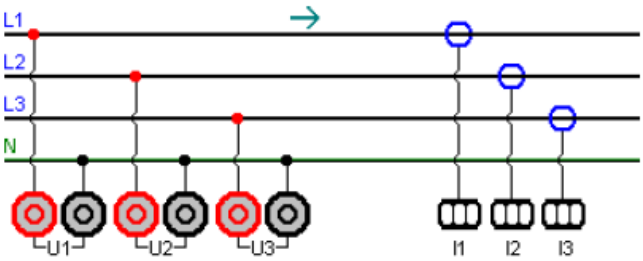
```
Power3 ( U1, I1, U2, I2, U3, I3 [, Intervalldaten] ) -> Ergebnisgruppe
```

Parameter:

U1	U1 = UL1 - N, Sternspannung, Strangspannung, Spannung der Phase gegen den Neutralleiter
I1	I1, auch IL1, Strom durch den Leiter, Stromrichtung (Pfeil) von der Quelle zum Verbraucher
U2	U2 = UL2 - N, Sternspannung, Strangspannung, Spannung der Phase gegen den Neutralleiter
I2	I2, auch IL2, Strom durch den Leiter, Stromrichtung (Pfeil) von der Quelle zum Verbraucher
U3	U3 = UL3 - N, Sternspannung, Strangspannung, Spannung der Phase gegen den Neutralleiter
I3	I3, auch IL3, Strom durch die Phase, Stromrichtung (Pfeil) von der Quelle zum Verbraucher
Intervalldaten	Intervalldaten, die Intervalle stellen die Perioden des Signals dar. Möglichst eine Folge von lückenlos aneinander grenzenden Intervallen. Die Berechnung erfolgt über diese Perioden. Vor allem bei unbekannter oder sich verändernder Signalfrequenz. (optional)
Ergebnisgruppe	Gruppe, die alle Rechenergebnisse enthält: P Wirkleistung, PF Leistungsfaktor, U* Effektivwerte der Spannungen, I* Effektivwerte der Ströme, p_t Momentanleistung, f Frequenz

Beschreibung:

Der Neutralleister N ist vorhanden und kann Strom führen. Typisch ist die Sternschaltung.



Um weitere Analysen auf den einzelnen Phasen durchzuführen, kann Power1() für jede der Phasen benutzt werden. Power3() liefert im Wesentlichen die summarischen Werte.

Die Funktionen PowerSelect() und PowerParameter() müssen vorher aufgerufen werden, um die Funktion komplett zu parametrieren.

Wird der Parameter Intervalldaten nicht angegeben, erfolgt die Berechnung mit der Frequenz, die in PowerParameter() angegeben wurde. Die Anzahl von Signalperioden, über die die Berechnung erfolgen soll, muss multipliziert mit der Periodendauer auf ein ganzes Vielfaches der Abtastzeit führen. So werden bei 50Hz oft 1 oder 10 Perioden angegeben, bei 60Hz dann 12 Perioden, Abtastfrequenz typisch 10kHz.

Wird der Parameter Intervalldaten angegeben, geben die Längen der Intervalle die Periodenlängen vor. Gibt es Lücken vor/zwischen/hinter den Intervallen, werden zur Berechnung kontinuierlich aneinander grenzende Perioden angenommen, die Intervallgrenzen werden dann mitunter nicht mehr benutzt, nur noch ihre Breiten. Start der Berechnung mit dem ersten Intervall oder auch einer ganzen Anzahl von Intervallbreiten vorher. Falls kein Intervall vorhanden, wird die Frequenz aus PowerParameter() benutzt.

Der Intervalldatensatz kann z.B. mit Hilfe von IntervalFromLevel() ermittelt und mit anderen Intervallfunktionen nachbearbeitet werden. Als Grundlage zur Bestimmung der Nulldurchgänge ist ein besonders sinusförmiges und gleichmäßiges Signal wie etwa eine Spannung empfohlen.

Intervalformat: Äquidistant "equi" oder XY "xy", d.h. Intervall-Code mit Position

Die Frequenz wird nur bestimmt, wenn ein Intervalldatensatz angegeben ist. Dann zeigt sie stets die der Berechnung zugrunde gelegte Frequenz an, auch an Stellen fehlender Intervalle.

Wenn mit Intervalldaten gearbeitet wird, ist eine möglichst präzise Bestimmung der Nullstellen erforderlich, um genaue Ergebnisse zu erhalten.

Die Frequenzbestimmung und auch die anderen Ergebnisse werden (deutlich) genauer, wenn die Analyse über mehrere Perioden durchgeführt wird.

Für eine genaue Analyse ist bei einer Netzfrequenz von 50Hz eine Abtastfrequenz von 10kHz sinnvoll.

Stromrichtung entsprechend Verbrauchersystem. War ein Amperemeter anderes herum geschaltet, wird z.B. anstelle von I1 als Parameter -I1 übergeben.

Strom und Spannung müssen dieselbe Zeitbasis haben. Sie dürfen Events haben. Segmente sind auch möglich, aber nicht in Zusammenhang mit Intervalldaten.

Die Berechnung liefert nur dann präzise Ergebnisse, wenn die Intervallbreite der Analyse eine ganze Anzahl von Samples beträgt.

Beispiele:

Leistungsmessung am Drehstromnetz mit 4 Leitern.

```
PowerSelect()
PowerParameter(50, 10)
g = Power3 ( U1, I1, U2, I2, U3, I3 )
P = g:P
Q = g:Q
```

Leistungsmessung am Drehstromnetz mit 4 Leitern. Frequenz schwankt

```
PowerSelect ()
PowerParameter ( 50, 1)
ivl = IntervalFromLevel ( U1, 0, 0, 0, 1, 0.0, 1e35, 0, "")
g = Power3 ( U1, I1, U2, I2, U3, I3 , ivl )
P = g:P
Q = g:Q
```

Siehe auch:

PowerSelect, PowerParameter, Power1, Power2, IntervalFromLevel

PowerCepstrum

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Power Cepstrum. Das Cepstrum wird mit gleitendem Fenster und linearer Mittelung bestimmt.

Deklaration:

```
PowerCepstrum ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Cepstrum wird ausgegeben.
Mittelungsart	Wie werden mehrere Cepstren gemittelt?
	0 : Keine Mittelung
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Cepstren). Der Mittelwert wird über so viele Cepstren gebildet, wie der Parameter Reduktion angibt.
	2 : Peak Hold Max ab Beginn, Maximalwerte, berechnet von allen bislang berechneten Cepstren
	3 : Peak Hold Max pro Intervall, Maximalwerte, über je so viele Cepstren gebildet, wie der Parameter Reduktion angibt.
	4 : Peak Hold Min ab Beginn, Minimalwerte, berechnet von allen bislang berechneten Cepstren
	5 : Peak Hold Min pro Intervall, Minimalwerte, über je so viele Cepstren gebildet, wie der Parameter Reduktion angibt.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von Cepstren, segmentierter Datensatz. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Cepstrum ist.

Beschreibung:

RMS-Spektrum des zweiseitigen logarithmierten (ln) Leistungsspektrums.

Dynamik auf 8 Zehnerpotenzen begrenzt.

Algorithmus:

- Den Ausschnitt aus dem Zeitdatensatz bilden, z.B. 1000 Punkte
- Ggf. auf 2er Potenz von Messwerten interpolieren, z.B. 1024
- Ggf. mit der Fensterfunktion multiplizieren
- Leistungsspektrum ermitteln. Nur Betrag weiterverwenden. Gibt 513 Punkte
- Dieses Spektrum in seiner Dynamik auf 8 Zehnerpotenzen begrenzen
- Den natürlichen Logarithmus vom Betrag ermitteln

- Das so gewonnene Spektrum auch für die entsprechenden negativen Frequenzen erweitern. Gibt 1024 Punkte
- Von diesem zweiseitigen Spektrum das RMS-Amplitudenspektrum bestimmen. Gibt 513 Punkte.
- Das so erhaltene Cepstrum wird anschließend der Mittelung zugeführt.

Beispiele:

```
PCepstren = PowerCepstrum ( Kanal, 1000, 0, 50, 1, 0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Cepstren, die sich um je 50% überlappen.

```
PCepstren = PowerCepstrum ( Kanal, 2048, 1, 0, 10, 1, 0 )
```

Das ist die Berechnung einer Folge von 2048 Punkte-Cepstren mit Hamming-Fenster. Über je 10 Cepstren wird der Mittelwert bestimmt und im Ergebnis vermerkt.

Siehe auch:

PowerCepstrum_exp, PowerCepstrum_1

PowerCepstrum_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Power Cepstrum. Ein gemitteltes Cepstrum wird bestimmt. Die Mittelung erfolgt über viele Cepstren, die aus einem gleitenden Fenster ermittelt wurden.

Deklaration:

```
PowerCepstrum_1 ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Mittelungsart	Wie werden mehrere Cepstren gemittelt?
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Cepstren). Der Mittelwert wird über alle berechneten Cepstren gebildet.
	2 : Peak Hold Max, Maximalwerte, berechnet von allen berechneten Cepstren
	4 : Peak Hold Min, Minimalwerte, berechnet von allen berechneten Cepstren
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ein gemitteltes Cepstrum als Ergebnis.

Beschreibung:

RMS-Spektrum des zweiseitigen logarithmierten (ln) Leistungsspektrums.

Dynamik auf 8 Zehnerpotenzen begrenzt.

Algorithmus:

- Den Ausschnitt aus dem Zeitdatensatz bilden, z.B. 1000 Punkte
- Ggf. auf 2er Potenz von Messwerten interpolieren, z.B. 1024
- Ggf. mit der Fensterfunktion multiplizieren
- Leistungsspektrum ermitteln. Nur Betrag weiterverwenden. Gibt 513 Punkte
- Dieses Spektrum in seiner Dynamik auf 8 Zehnerpotenzen begrenzen
- Den natürlichen Logarithmus vom Betrag ermitteln
- Das so gewonnene Spektrum auch für die entsprechenden negativen Frequenzen erweitern. Gibt 1024 Punkte
- Von diesem zweiseitigen Spektrum das RMS-Amplitudenspektrum bestimmen. Gibt 513 Punkte.
- Das so erhaltene Cepstrum wird anschließend der Mittelung zugeführt.

Beispiele:

```
Cepstrum = PowerCepstrum_1 ( Kanal, 1000, 0, 50, 1, 0 )
```

Das ist die Berechnung eines gemittelten Cepstrums. Die Mittelung geschieht über eine Folge von 1000 Punkte-Cepstren, die sich um je 50% überlappen. Der Eingangskanal hat etwa 20000 Messwerte.

Siehe auch:

PowerCepstrum, PowerCepstrum_exp

PowerCepstrum_exp

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Power Cepstrum, exponentielle Mittelung

Deklaration:

```
PowerCepstrum_exp ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Zeitkonstante [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Cepstrum wird ausgegeben.
Zeitkonstante	Die Zeitkonstante für die exponentielle Mittelung. In Sekunden angegeben.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von Cepstren, segmentierter Datensatz. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Cepstrum ist.

Beschreibung:

RMS-Spektrum des zweiseitigen logarithmierten (ln) Leistungsspektrums.

Dynamik auf 8 Zehnerpotenzen begrenzt.

Algorithmus:

- Den Ausschnitt aus dem Zeitdatensatz bilden, z.B. 1000 Punkte
- Ggf. auf 2er Potenz von Messwerten interpolieren, z.B. 1024
- Ggf. mit der Fensterfunktion multiplizieren
- Leistungsspektrum ermitteln. Nur Betrag weiterverwenden. Gibt 513 Punkte
- Dieses Spektrum in seiner Dynamik auf 8 Zehnerpotenzen begrenzen
- Den natürlichen Logarithmus vom Betrag ermitteln
- Das so gewonnene Spektrum auch für die entsprechenden negativen Frequenzen erweitern. Gibt 1024 Punkte
- Von diesem zweiseitigen Spektrum das RMS-Amplitudenspektrum bestimmen. Gibt 513 Punkte.
- Das so erhaltene Cepstrum wird anschließend der Mittelung zugeführt.

Beispiele:

```
Cepstren = PowerCepstrum_exp ( Kanal, 1000, 0, 50, 2, 40.0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Cepstren, die sich um je 50% überlappen. Der Kanal hat eine Abtastzeit von 10ms. Also wird alle 5s ein Cepstrum gebildet. Diese werden mit einer Zeitkonstante von 40.0s geglättet. Jedes 2. Cepstrum wird ausgegeben.

Siehe auch:

PowerCepstrum, PowerCepstrum_1

PowerDS

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Leistungsdichtespektrum (Power Density Spectrum) mit gleitendem Fenster und linearer Mittelung. Quadrat des RMS-Spektrums, geteilt durch Frequenzlinienabstand. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Deklaration:

```
PowerDS ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	0 : Keine Mittelung
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Quadrate der Betragsspektren). Der Mittelwert wird über so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	2 : Peak Hold Max ab Beginn, Maximalwerte, berechnet von allen bislang berechneten Spektren
	3 : Peak Hold Max pro Intervall, Maximalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	4 : Peak Hold Min ab Beginn, Minimalwerte, berechnet von allen bislang berechneten Spektren
	5 : Peak Hold Min pro Intervall, Minimalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von Leistungsdichte-Spektren, segmentierter Datensatz.

Beschreibung:

Das Ergebnis wird durch die ENBW (Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch 1.5 im Fall des Hanning-Fensters.

Beispiele:

```
LeistungsdichteSpektren = PowerDS ( Kanal, 1000, 0, 50, 1, 0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen.

```
LeistungsdichteSpektren = PowerDS ( Kanal, 2048, 1, 0, 10, 1, 0 )
```

Das ist die Berechnung einer Folge von 2048 Punkte-Spektren mit Hamming-Fenster. Über je 10 Spektren wird der Mittelwert bestimmt und im

Ergebnis vermerkt.

Siehe auch:

PowerDS_exp, PowerDS_1, PowerSpectrum

PowerDS_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein gemitteltes Leistungsdichtespektrum (Power Density Spectrum) wird bestimmt. Quadrat des RMS-Spektrums, geteilt durch Frequenzlinienabstand. Die Mittelung erfolgt über viele Spektren, die aus einem gleitenden Fenster ermittelt wurden. Berechnung mittels FFT.

Deklaration:

```
PowerDS_1 ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Quadrate der Betragsspektren). Der Mittelwert wird über alle berechneten Spektren gebildet.
	2 : Peak Hold Max, Maximalwerte, berechnet von allen berechneten Spektren
	4 : Peak Hold Min, Minimalwerte, berechnet von allen berechneten Spektren
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ein gemitteltes Spektrum als Ergebnis.

Beschreibung:

Das Ergebnis wird durch die ENBW (Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch 1.5 im Fall des Hanning-Fensters.

Intern verwendete Multiplikationsfaktoren bei Bewertung mit ENBW: Rechteck: 1.0; Hamming: 1.0 / 1.3628257; Hanning: 1.0 / 1.5000000; Blackman: 1.0 / 1.7267573; Blackman-Harris: 1.0 / 2.0043528; Flat top: 1.0 / 3.7702901

Beispiele:

```
LeistungsdichteSpektrum = PowerDS_1 ( Kanal, 1000, 0, 50, 1, 0 )
```

Das ist die Berechnung eines gemittelten Spektrums. Die Mittelung geschieht über eine Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Eingangskanal hat etwa 20000 Messwerte.

Siehe auch:

PowerDS, PowerDS_exp, PowerSpectrum_1

PowerDS_exp

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Leistungsdichtespektrum (Power Density Spectrum) mit gleitendem Fenster und exponentieller Mittelung. Quadrat des RMS-Spektrums, geteilt durch Frequenzlinienabstand. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Deklaration:

```
PowerDS_exp ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Zeitkonstante [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Zeitkonstante	Die Zeitkonstante für die exponentielle Mittelung. In Sekunden angegeben.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz.

Beschreibung:

Das Ergebnis wird durch die ENBW (Equivalent noise bandwidth) entsprechend der verwendeten Fensterfunktion geteilt. Z.B. durch 1.5 im Fall des Hanning-Fensters.

Beispiele:

```
LeistungsdichteSpektren = PowerDS_exp ( Kanal, 1000, 0, 50, 2, 40.0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Kanal hat eine Abtastzeit von 10ms. Also wird alle 5s ein Spektrum gebildet. Diese werden mit einer Zeitkonstante von 40.0s geglättet. Jedes 2. Spektrum wird ausgegeben.

Siehe auch:

PowerDS, PowerDS_1, PowerSpectrum _exp

PowerParameter

Verfügbar ab: Professional Edition

Legt fest, mit welchen Parametern und Verfahren die Leistungsberechnung erfolgen soll

Deklaration:

```
PowerParameter ( f, Periods [, S_calc] [, Q_calc] [, Q_sign] [, PF_calc] )
```

Parameter:

f	Nominalfrequenz in Hz, z.B. 50 oder 60
Periods	Anzahl der Perioden, über die eine Berechnung stattfindet und nach der ein Ergebnis ausgegeben wird, z.B. 1 oder 10 oder 12
S_calc	Scheinleistungsberechnung (optional , Standardwert: 0)
	0 : nach DIN 40110, S = Ueff*Ieff, Ieff=sqrt(I1^2+I2^2+I3^2), 4-Leitersystem: Ueff=sqrt(U1^2+U2^2+U3^2), 3-Leitersystem:Ueff= sqrt((U12^2+U32^2+U13^2)/3)
	1 : 4-Leitersystem: S = S1 + S2 + S3, 3-Leitersystem S = (sqrt(3)/2)*(U12*I1+U32*I3)
Q_calc	Blindleistungsberechnung (optional , Standardwert: 0)
	0 : Blindleistung = sqrt (Gesamte Scheinleistung ^2 - Gesamte Wirkleistung ^2)
	1 : Blindleistung = Q1+Q2+Q3 (nicht bei Aron möglich)
Q_sign	Blindleistungsvorzeichen (optional , Standardwert: 0)
	0 : Blindleistung stets positiv
	1 : Blindleistung mit Vorzeichen. positiv bei induktiv, negativ bei kapazitiv (nicht bei Aron möglich)
PF_calc	Leistungsfaktorberechnung (optional , Standardwert: 0)
	0 : Vorzeichen der Wirkleistung, positiv für Leistungsbezug, negativ für Leistungsabgabe
	1 : induktiv: positiv bei Bezug, negativ bei Abgabe; kapazitiv: negativ bei Bezug, positiv bei Abgabe (nicht bei Aron möglich)
	2 : stets positiv

Beschreibung:

Die Bestimmung des induktiven/kapazitiven Betriebs wird anhand der Analyse der Grundschwingungsphasenblindleistung U1*I1*sin(Phi1) durchgeführt. Bei 3 Phasen werden dazu die Grundschwingungsphasenblindleistungen aller Phasen addiert.

Für den induktiven/kapazitiven Betrieb gilt mit Phi als Winkel, um den der Strom nachläuft: Quadrant 1: Phase Phi = 0 bis 90 Grad motorisch induktiv, Quadrant 2: Phase Phi = 90 bis 180 Grad generatorisch induktiv, Quadrant 3: Phase Phi = 180 bis 270 Grad generatorisch kapazitiv, Quadrant 4: Phase Phi = 270 bis 360 Grad motorisch kapazitiv.

Die Wirkleistung ist in den Quadranten 1 und 4 positiv, sonst negativ.

Die Blindleistung ist in den Quadranten 1 und 2 positiv, sonst negativ, falls mit Vorzeichen ermittelt.

Beispiele:

1-phasige Leistungsmessung. Standardberechnung bei 50Hz über 10 Perioden, Abtastfrequenz 10kHz

```
PowerSelect()
PowerParameter(50, 10)
g = Power1 ( U1, I1 )
```

1-phasige Leistungsmessung. Standardberechnung bei 60Hz über 12 Perioden, Abtastfrequenz 10kHz

```
PowerSelect()
PowerParameter(60, 12)
g = Power1 ( U1, I1 )
```

1-phasige Leistungsmessung. Blindleistung und Leistungsfaktor mit Vorzeichen

```
PowerSelect()
PowerParameter(50, 1, 0, 0, 1, 1)
g = Power1 ( U1, I1 )
```

Siehe auch:

PowerSelect, Power1, Power2, Power3

PowerSelect

Verfügbar ab: Professional Edition

Legt fest, welche Ergebnisse eine Leistungsberechnung liefern soll.

Deklaration:

```
PowerSelect ( [QS] [, RMS] [, PF] [, f] [, p_t] [, P] )
```

Parameter:

QS	Blind- und Scheinleistung gewünscht (optional , Standardwert: 1)
	0 : nein
	1 : ja
RMS	Effektivwerte der Ströme und Spannungen gewünscht (optional , Standardwert: 1)
	0 : nein
	1 : ja
PF	Leistungsfaktor gewünscht (optional , Standardwert: 1)
	0 : nein
	1 : ja
f	Signalfrequenz gewünscht (optional , Standardwert: 1)
	0 : nein
	1 : ja
p_t	Momentanleistung gewünscht (optional , Standardwert: 0)
	0 : nein
	1 : ja
P	Wirkleistung gewünscht (optional , Standardwert: 1)
	0 : nein
	1 : ja

Beschreibung:

Die Funktion muss einmal vor den Funktionen Power1(), Power2(), Power3() aufgerufen werden.

Wenn die Frequenz gewünscht ist, wird sie aber dennoch nur berechnet, wenn Power1(), Power2() oder Power3() mit einem Intervalldatensatz aufgerufen werden, weil die Frequenz nicht fest ist.

Beispiele:

1-phasige Leistungsmessung. Standardberechnung

```
PowerSelect()
PowerParameter(50, 10)
g = Power1 ( U1, I1 )
```

1-phasige Leistungsmessung. Explizite Auswahl der Berechnung der Wirkleistung und des Leistungsfaktors

```
PowerSelect(0, 0, 1, 0, 0, 1)
PowerParameter(50, 10)
g = Power1 ( U1, I1 )
```

Siehe auch:

PowerParameter, Power1, Power2, Power3

PowerSpectrum

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Leistungsspektrum (Power Spectrum) mit gleitendem Fenster und linearer Mittelung. Quadrat des RMS-Spektrums. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Deklaration:

```
PowerSpectrum ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	0 : Keine Mittelung
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Quadrate der Betragsspektren). Der Mittelwert wird über so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	2 : Peak Hold Max ab Beginn, Maximalwerte, berechnet von allen bislang berechneten Spektren
	3 : Peak Hold Max pro Intervall, Maximalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
	4 : Peak Hold Min ab Beginn, Minimalwerte, berechnet von allen bislang berechneten Spektren
	5 : Peak Hold Min pro Intervall, Minimalwerte, über je so viele Spektren gebildet, wie der Parameter Reduktion angibt.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz.

Beschreibung:

Beispiele:

```
Leistungsspektren = PowerSpectrum ( Kanal, 1000, 0, 50, 1, 0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen.

```
Leistungsspektren = PowerSpectrum ( Kanal, 2048, 1, 0, 10, 1, 0 )
```

Das ist die Berechnung einer Folge von 2048 Punkte-Spektren mit Hamming-Fenster. Über je 10 Spektren wird der Mittelwert bestimmt und im Ergebnis vermerkt.

Siehe auch:

PowerSpectrum_exp, PowerSpectrum_1, PowerDS

PowerSpectrum_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Ein gemitteltes Leistungsspektrum (Power Spectrum) wird bestimmt. Quadrat des RMS-Spektrums. Die Mittelung erfolgt über viele Spektren, die aus einem gleitenden Fenster ermittelt wurden.

Deklaration:

```
PowerSpectrum_1 ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Mittelungsart [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Mittelungsart	Wie werden mehrere Spektren gemittelt?
	1 : Mittelung (arithmetisches Mittel bzw. lineare Mittelung der Quadrate der Betragsspektren). Der Mittelwert wird über alle berechneten Spektren gebildet.
	2 : Peak Hold Max, Maximalwerte, berechnet von allen berechneten Spektren
	4 : Peak Hold Min, Minimalwerte, berechnet von allen berechneten Spektren
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Ein gemitteltes Spektrum als Ergebnis.

Beschreibung:

Beispiele:

```
Leistungsspektrum = PowerSpectrum_1 ( Kanal, 1000, 0, 50, 1, 0 )
```

Das ist die Berechnung eines gemittelten Spektrums. Die Mittelung geschieht über eine Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Eingangskanal hat etwa 20000 Messwerte.

Siehe auch:

PowerSpectrum, PowerSpectrum_exp, PowerDS_1

PowerSpectrum_exp

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Leistungsspektrum (Power Spectrum) mit gleitendem Fenster und exponentieller Mittelung. Quadrat des RMS-Spektrums. Das Ergebnis ist ein Datensatz mit Segmenten, wobei jedes Segment ein Spektrum ist.

Deklaration:

```
PowerSpectrum_exp ( Eingangsdaten, Fensterbreite, Fenstertyp, Überlappung, Reduktion, Zeitkonstante [, Basis2] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Der Zeitdatensatz, die Zeit in Sekunden skaliert
Fensterbreite	Breite des Zeitfensters in Punkten, >= 4. Falls keine 2er-Potenz, wird je nach Parameter 'Basis2' auf eine etwas kleinere Abtastzeit interpoliert.
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Überlappung	Um so viele Prozent überlappen sich die Zeitfenster.
	0 : Keine Überlappung
	> 0 : Überlappung. Wert > 0, aber < 100. Je größer, desto mehr Rechenzeit.
	< 0 : Wert < 0, Abstand negativ angegeben. So viel Prozent der Fensterbreite wird Abstand gelassen beim Vorrücken der Fenster. Damit werden diese Werte nicht bei der Berechnung berücksichtigt.
Reduktion	>= 1: Nur jedes soundsovielte Spektrum wird ausgegeben.
Zeitkonstante	Die Zeitkonstante für die exponentielle Mittelung. In Sekunden angegeben.
Basis2	Soll die interne Berechnung der FFT nur mit 2er-Potenzen (Basis 2) erfolgen oder auch mit anderen Fensterbreiten? Empfohlen ist der Wert 3. Wenn nicht angegeben, wird 2 verwendet. (optional)
	2 : Wenn die Fensterbreite keine 2er-Potenz ist, werden die Daten für eine FFT-Berechnung auf eine 2er-Potenz interpoliert.
	3 : FFT mit allen Fensterbreiten, die ein Produkt aus Potenzen von 2, 3, 5 sind; keine Interpolation der Zeitdaten
Ergebnis	Folge von FFT-Spektren, segmentierter Datensatz.

Beschreibung:

Beispiele:

```
Leistungsspektren = PowerSpectrum_exp ( Kanal, 1000, 0, 50, 2, 40.0, 0 )
```

Das ist die Berechnung einer Folge von 1000 Punkte-Spektren, die sich um je 50% überlappen. Der Kanal hat eine Abtastzeit von 10ms. Also wird alle 5s ein Spektrum gebildet. Diese werden mit einer Zeitkonstante von 40.0s geglättet. Jedes 2. Spektrum wird ausgegeben.

Siehe auch:

PowerSpectrum_1, PowerSpectrum, PowerDS_exp

PptAddSlides

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Folien aus einer PowerPoint-Datei werden in die Präsentation eingefügt.

Deklaration:

```
PptAddSlides ( SlideIndex, TxFilename [, SlideStartIndex] [, SlideEndIndex] ) -> Result
```

Parameter:

SlideIndex	Index der Folie hinter der die Folien aus der Datei eingefügt werden. Geben Sie eine 0 an, wenn Sie die Folien vor der ersten Folie einfügen wollen.
TxFilename	Kompletter Pfadname einer PowerPoint- Datei aus der die Folien eingefügt werden.
SlideStartIndex	Index der ersten Folie, die aus der Datei eingefügt werden soll. (1...) (optional) (optional)
SlideEndIndex	Index der letzten Folie, die aus der Datei eingefügt werden soll. (>= SlideStartIndex oder -1) (optional) (optional)
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Folien werden aus einer PowerPoint-Datei in die Präsentation eingefügt.

Sind die Parameter 'SlideStartIndex' und 'SlideEndIndex' nicht angegeben, so werden alle Folien aus der PowerPoint-Datei eingefügt.

Ist nur der Parameter 'SlideStartIndex' angegeben, so wird nur die Folie mit dem angegebenen Index aus der PowerPoint-Datei eingefügt.

Sind beide Parameter angegeben, so werden die Folien zwischen den angegebenen Indizes aus der PowerPoint-Datei eingefügt.

Wird der Parameter 'SlideEndIndex'= -1 angegeben, so werden die Folien von 'SlideStartIndex' bis zur letzten Folie eingefügt.

Beispiele:

Die PowerPoint-Datei 'Presentation1' wird geöffnet. Anschließend werden an den Anfang alle Folien aus der Datei 'Presentation2' eingefügt.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
count = PptGetSlidesCount( )
result= PptAddSlides( 0,"c:\imc\ppt\Presentation2.pptx")
count = PptGetSlidesCount( )
result= PptClosePresentation( )
```

Die PowerPoint-Datei 'Presentation1' wird geöffnet. Anschließend wird hinter der letzten Folie die 2. Folie aus der Datei 'Presentation2' eingefügt.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
count = PptGetSlidesCount( )
result= PptAddSlides( count,"c:\imc\ppt\Presentation2.pptx",2)
result= PptClosePresentation( )
```

Siehe auch:

PptGetSlidesCount, PptDuplicateSlide, PptDeleteSlide, PptMoveSlide

PptClosePresentation

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Die Präsentation wird geschlossen.

Deklaration:

```
PptClosePresentation ( ) -> Result
```

Parameter:

Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Mit dieser Funktion wird gleichzeitig die PowerPoint- Instanz beendet.

Eventuell noch offene Dateien werden geschlossen. Dabei gehen eventuelle Änderungen verloren.

Diese Funktion sollte immer am Ende einer Sequenz aufgerufen werden.

Siehe auch:

PptOpenPresentation, PptSavePresentation

PptDeleteSlide

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Eine Folie wird aus der Präsentation entfernt.

Deklaration:

```
PptDeleteSlide ( SlideIndex ) -> Result
```

Parameter:

SlideIndex	Index der zu löschenden Folie. Die erste Folie hat den Index 1.
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Eine Folie wird aus der Präsentation entfernt.

Beispiele:

Die erste und letzte Folie werden aus der Präsentation gelöscht.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
result= PptDeleteSlide( 1)
count = PptGetSlidesCount( )
result= PptDeleteSlide(count)
result= PptSavePresentation("c:\imc\ppt\Presentation3.pptx")
result= PptClosePresentation( )
```

Siehe auch:

PptAddSlides, PptGetSlidesCount, PptDuplicateSlide

PptDuplicateSlide

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Dupliziert eine Folie der Präsentation.

Deklaration:

```
PptDuplicateSlide ( SlideIndex ) -> Result
```

Parameter:

SlideIndex	Index der Folie, die dupliziert werden soll. (1...)
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Eine Folie der Präsentation wird dupliziert.

Die duplizierte Folie wird direkt nach der ursprünglichen Folie in die Präsentation eingefügt.

Beispiele:

Die letzte Folie der Präsentation wird dupliziert.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
count = PptGetSlidesCount( ) ; Index der letzten Folie bestimmen
result= PptDuplicateSlide(count) ; Folie duplizieren
result= PptSavePresentation("c:\imc\ppt\Presentation3.pptx")
result= PptClosePresentation( )
```

Siehe auch:

PptAddSlides, PptDeleteSlide, PptMoveSlide, PptGetSlidesCount

PptFindSlideByAlternativeText

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Sucht eine Folie mit einem Formobjekt mit dem angegebenen Alternativtext.

Deklaration:

```
PptFindSlideByAlternativeText ( StartIndex, TxAlternativeText ) -> Result
```

Parameter:

StartIndex	Index der Folie, bei der die Suche beginnen soll. (1...)
TxAlternativeText	Der Alternativtext des Formobjektes.
Result	Ergebnis:
	>0 : Index der Folie, die das Formobjekt enthält
	=0 : Es gibt keine Folie mit einem Formobjekt mit dem angegebenen Alternativtext.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Ab dem 'startindex' wird die Präsentation nach Folien durchsucht, die ein Formobjekt mit dem angegebenen Alternativtext enthalten.

Beim ersten Auftreten des Formobjekts wird der Index der Folie zurückgegeben.

Siehe auch:

PptAddSlides, PptDeleteSlide, PptGetSlidesCount, PptDuplicateSlide, PptMoveSlide, PptGetSlidesCount

PptGetSlidesCount

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Die Anzahl der Folien in der Präsentation wird ermittelt.

Deklaration:

```
PptGetSlidesCount ( ) -> Result
```

Parameter:

Result	Ergebnis:
	>=0 : Anzahl der Folien
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Es wird die Anzahl der Folien ermittelt.

Im Fehlerfall wird eine -1 zurückgegeben. Die Ursache kann dann mit der Funktion GetLastError() ermittelt werden.

Siehe auch:

PptAddSlides, PptDeleteSlide, PptDuplicateSlide, PptMoveSlide

PptMoveSlide

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Verschiebt die angegebene Folie an eine bestimmte Position in der Präsentation.

Deklaration:

```
PptMoveSlide ( SlideIndex, ToPosition ) -> Result
```

Parameter:

SlideIndex	Index der Folie, die verschoben werden soll. (1...)
ToPosition	Index der neuen Position. (1...)
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Eine Folie der Präsentation wird verschoben.

Alle anderen Folien in der Präsentation werden entsprechend neu nummeriert.

Beispiele:

Die 1. Folie der Präsentation wird dupliziert und an die letzte Position der Präsentation verschoben.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
slideindex=1
result= PptDuplicateSlide(slideindex) ; Folie duplizieren
slideindex= slideindex+1; Index der duplizierten Folie
count = PptGetSlidesCount( ) ; Index der letzten Folie bestimmen
result= PptMoveSlide(slideindex,count); Duplizierte Folie an die letzte Position verschieben
result= PptSavePresentation("c:\imc\ppt\Presentation3.pptx")
result= PptClosePresentation( )
```

Siehe auch:

PptAddSlides, PptDeleteSlide, PptGetSlidesCount, PptDuplicateSlide, PptGetSlidesCount

PptOpenPresentation

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Die angegebene Datei im PowerPoint- Format wird geöffnet.

Deklaration:

```
PptOpenPresentation ( TxFilename, Visible ) -> Result
```

Parameter:

TxFilename	Kompletter Pfadname der zu öffnenden Datei
Visible	Sichtbarkeit des Arbeitsfensters
	0 : Das Arbeitsfensters von PowerPoint ist nicht sichtbar.
	1 : Das Arbeitsfensters von PowerPoint ist sichtbar.
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die angegebene PowerPoint- Datei wird geöffnet. Dazu wird eine PowerPoint- Instanz gestartet, falls noch nicht geschehen.

War vor diesem Aufruf eine andere Präsentation vom selben Ausführungs-Thread geöffnet, so wird diese geschlossen.

Multithreading: Jeder Ausführungs-Thread verwendet eine eigene Powerpoint-Instanz. Wenn also beispielsweise in einer parallel ausgeführten Sequenzfunktion (BEGIN_PARALLEL) mittels PptOpenPresentation() Powerpoint gestartet und ein Dokument geladen wird, sind weitere Zugriffe auf dieses Dokument nur innerhalb der selben Sequenzfunktion erlaubt. Wenn die Powerpoint-Instanz nicht explizit mit PptClosePresentation() geschlossen wurde, erfolgt das Schließen automatisch am Ende der Sequenzfunktion.

Beispiele:

Eine PowerPoint- Datei wird geöffnet. Es werden Folien aus einer anderen PowerPoint- Datei zugefügt. Die Präsentation wird unter einem anderen Namen gespeichert.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1);
count = PptGetSlidesCount( )
result= PptAddSlides(count,"c:\imc\ppt\Presentation2.pptx")
result= PptSavePresentation("c:\imc\ppt\Result.pptx")
result= PptClosePresentation( )
```

Siehe auch:

PptSavePresentation, PptClosePresentation

PptPrintPresentation

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Die Präsentation wird gedruckt.

Deklaration:

```
PptPrintPresentation ( [FromSlideIndex] [, ToSlideIndex] ) -> Result
```

Parameter:

FromSlideIndex	Index der ersten Folie, die gedruckt werden soll. (1...) (optional) (optional)
ToSlideIndex	Index der letzten Folie, die gedruckt werden soll. (optional) (optional)
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die komplette Präsentation bzw. Folien aus der Präsentation werden gedruckt.

Sind die Parameter 'FromSlideIndex' und 'ToSlideIndex' nicht angegeben, so wird die ganze Präsentation gedruckt.

Ist nur der Parameter 'FromSlideIndex' angegeben, so wird nur die Folie mit dem angegebenen Index gedruckt.

Sind beide Parameter angegeben, so werden die Folien zwischen den angegebenen Indizes gedruckt.

Beispiele:

Die Präsentation Presentation1.pptx wird geöffnet. Es werden alle Folien gedruckt.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
result= PptPrintPresentation( )
result= PptClosePresentation( )
```

Die Präsentation Presentation1.pptx wird geöffnet. Es wird die 2. Folie gedruckt.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
result= PptPrintPresentation(2)
result= PptClosePresentation( )
```

Siehe auch:

PptOpenPresentation, PptSavePresentation

PptSavePresentation

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Die Präsentation wird gespeichert.

Deklaration:

```
PptSavePresentation ( TxFilename [, FileFormat] ) -> Result
```

Parameter:

TxFilename	Kompletter Pfadname, unter dem die Präsentation gespeichert werden soll.
FileFormat	Legt das Dateiformat für das Speichern fest. Ist der Parameter nicht angegeben, so wird im Standardformat gespeichert. (optional) (optional)
	0 : Präsentation im Standardformat speichern.
	1 : als PDF- Datei speichern.
	2 : als JPG- Datei speichern.
	3 : als PNG- Datei speichern.
	4 : als BMP- Datei speichern.
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die Präsentation wird unter dem angegebenen Dateinamen gespeichert.

Fehlt der Parameter 'FileFormat', so wird die Präsentation im PowerPoint- Standardformat gespeichert.

Ebenso wird im Standardformat gespeichert, wenn Parameter =0 ist.

Mit 'FileFormat' = 1 wird die Präsentation als PDF- Datei gespeichert.

Bei den Formaten 2, 3 und 4 wird aus dem Dateinamen ein Verzeichnisname gebildet. In diesem Verzeichnis ist jede Folie als JPG, PNG oder BMP-Datei gespeichert.

Beispiele:

Eine Präsentation wird geöffnet und in verschiedenen Format gespeichert.

```
result=PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1);
result=PptSavePresentation("c:\imc\ppt\test.pptx"); Standardformat
result=PptSavePresentation("c:\imc\ppt\test1.pptx",0); Standardformat
result=PptSavePresentation("c:\imc\ppt\test.pdf",1)  ; PDF-Format
result=PptSavePresentation("c:\imc\ppt\testa.jpg",2) ; JPG-Format
result=PptSavePresentation("c:\imc\ppt\testb.png",3) ; PNG-Format
result=PptSavePresentation("c:\imc\ppt\testc.bmp",4) ; BMP-Format
result=PptClosePresentation( )
```

Siehe auch:

PptOpenPresentation, PptClosePresentation

PptSetCellText

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Setzt den Text in einer Zelle einer Tabelle in der angegebenen Folie.

Deklaration:

```
PptSetCellText ( SlideIndex, TxAlternativeText, Row, Column, TxContent ) -> Result
```

Parameter:

SlideIndex	Index der Folie. Die erste Folie hat den Index 1.
TxAlternativeText	Der Alternativtext der Tabelle.
Row	Zeilennummer der Zelle (1...).
Column	Spaltennummer der Zelle (1...).
TxContent	Dieser Text wird in die Zelle übertragen.
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Über den 'SlideIndex' wird die Folie in der Präsentation ermittelt. Der Index entspricht der Position der Folie in der Präsentation.

In der angegebenen Folie wird nach der Tabelle mit dem Alternativtext gesucht.

Wurde die Tabelle gefunden, so wird der übergebene Text in die Zelle übertragen.

Die Zelle wird durch die Angabe der Zeilen- und Spaltennummer adressiert.

Eine Zelle kann wie ein Textfeld mehrere Absätze haben.In diesem Fall wird der übergebene Text in Teilzeichenketten aufgespaltet.

Die Aufspaltung erfolgt an den Zeichen CR ("~013") , LF ("~010") oder NL ("~013"+ "~010").

Jede Teilzeichenkette wird dem korrespondierenden Absatz der Zelle zugewiesen.

So definieren Sie in PowerPoint den Alternativtext für eine Tabelle:

1. Selektieren Sie auf der Folie das Tabellen-Objekt.
2. Klicken Sie mit der rechten Maustaste in das Element und wählen Sie "Form formatieren" aus.
3. Klicken Sie auf "Größe und Eigenschaften" und dann auf "Alternativtext".
4. Geben Sie im Feld "Beschreibung" einen Text für die Kennzeichnung ein, z.B. FAMOS_Table1. Dieser Kennzeichnungstext wird in den Kit-Funktionen zum Suchen des Formobjektes benutzt.
5. **Wichtig!** Geben Sie den Kennzeichnungstext nicht im Feld "Titel" ein.
6. Wollen Sie den Alternativtext für seine ursprüngliche Funktion benutzen (barrierefreies PowerPoint), so geben Sie im Feld "Beschreibung" zuerst den Text für die Kennzeichnung ein, gefolgt von einem Semikolon (;). Dann kann der Text folgen, der für ein barrierefreies PowerPoint gedacht ist. Das Semikolon ist nicht Teil des Kennzeichnungstextes.

Beispiele:

Die Präsentation enthält eine Tabelle mit 3 Spalten und 6 Zeilen. Die Zellen der zweiten bis 4. Zeile werden gesetzt.

```
result=pptOpenPresentation( "c:\imc\ppt\PresentationTable1.pptx",1)
result=PptSetCellText( 1,"Table1",2,1,"Channel 1"); Channel name -> column 1
result=PptSetCellText( 1,"Table1",2,2,"34.06"); Maximum -> column 2
result=PptSetCellText( 1,"Table1",2,3,"-3.02"); Minimum -> Column 3
result=PptSetCellText( 1,"Table1",3,1,"Channel 2")
result=PptSetCellText( 1,"Table1",3,2,"10.4555")
result=PptSetCellText( 1,"Table1",3,3,"1.02")
result=PptSetCellText( 1,"Table1",4,1,"Channel 3")
result=PptSetCellText( 1,"Table1",4,2,"210.4555")
result=PptSetCellText( 1,"Table1",4,3,"-23.02")
result=pptsavePresentation("c:\imc\ppt\test.pptx")
result=pptClosePresentation( )
```

Siehe auch:

PptSetText, PptSetPicture, PptSetCurve

PptSetCurve

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Überträgt das ausgewählte Kurvenfenster als Bild in die angegebene Folie.

Deklaration:

```
PptSetCurve ( SlideIndex, TxAlternativeText, SizeOption, Position, Screen ) -> Result
```

Parameter:

SlideIndex	Index der Folie. Die erste Folie hat den Index 1.
TxAlternativeText	Der Alternativtext des Bildes.
SizeOption	Anpassung der Bildgröße.
	0 : Die Orginalgröße des Bildes wird verwendet.
	1 : Die Größe des Platzhalters wird verwendet.
	2 : Die Größe des Platzhalters wird verwendet. Dabei werden die Seitenverhältnisse des Orginalbildes beibehalten.
Position	Der Teil der Form (Platzhalter des Bildes), der bei der Skalierung der Form seine Position behält.
	0 : Oben links.
	1 : Unten rechts.
	2 : Zentriert.
Screen	Bildschirm oder Drucker.
	"screen" : Zur Erzeugung des Bildes werden die Bildschirmeinstellungen des Kurvenfensters verwendet.
	"printer" : Zur Erzeugung des Bildes werden die Druckereinstellungen des Kurvenfensters verwendet.
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Ein Kurvenfenster wird als Bild in die Präsentation übertragen. Dazu muss die Folie ein Bild oder einen Bildplatzhalter haben.

Das Kurvenfenster muss vor dem Aufruf dieser Funktion erzeugt und selektiert werden. Dazu können die Funktionen des Kurvenfenster- Kits (Cw...- Funktionen) benutzen.

Über den 'SlideIndex' wird die Folie in der Präsentation ermittelt. Der Index entspricht der Position der Folie in der Präsentation.

In der angegebenen Folie wird nach dem Bildplatzhalter mit dem angegebenen Alternativtext gesucht.

Wurde das Bild gefunden, so wird das Kurvenfenster in das Bild der Präsentation übertragen.

Mit 'SizeOption' = 0 wird das Kurvenfenster mit seiner Orginalgröße in den Platzhalter übertragen.
Über den Parameter 'Position' wird festgelegt, an welcher Stelle des Platzhalters das Bild verankert wird.
Wenn das Kurvenfenster größer ist als der Platzhalter, kann es vorkommen, dass andere Formobjekte überdeckt werden.

Mit 'SizeOption' = 1 wird das Kurvenfenster in die Fläche des Bildplatzhalters eingepasst.
Es füllt den Platzhalterbereich vollständig aus. Der Parameter 'Position' ist nicht relevant.
Bei dieser Option kann es zu Verzerrungen des Kurvenfensters kommen.

Mit 'SizeOption' = 2 wird das Kurvenfenster unter Beibehaltung der Seitenverhältnisse in die Fläche des Bildplatzhalters eingepasst.
Über den Parameter 'Position' wird festgelegt, an welcher Stelle des Platzhalters das Kurvenfenster verankert wird.

So definieren Sie in PowerPoint den Alternativtext für ein Bild:

1. Selektieren Sie auf der Folie das Bild-Objekt.
2. Klicken Sie mit der rechten Maustaste in das Element und wählen Sie "Grafik formatieren" aus.
3. Klicken Sie auf "Größe und Eigenschaften" und dann auf "Alternativtext".
4. Geben Sie im Feld "Beschreibung" einen Text für die Kennzeichnung ein, z.B. FAMOS_Cw1. Dieser Kennzeichnungstext wird in den Kit-Funktionen zum Suchen des Formobjektes benutzt.
5. **Wichtig!** Geben Sie den Kennzeichnungstext nicht im Feld "Titel" ein.
6. Wollen Sie den Alternativtext für seine ursprüngliche Funktion benutzen (barrierefreies PowerPoint), so geben Sie im Feld "Beschreibung" zuerst den Text für die Kennzeichnung ein, gefolgt von einem Semikolon (;). Dann kann der Text folgen, der für ein barrierefreies PowerPoint gedacht ist. Das Semikolon ist nicht Teil des Kennzeichnungstextes.

Beispiele:

In der Sequenz werden 2 Kurvenfenster erzeugt. Dann wird die Präsentation 'PräsentationCW3' geöffnet.
Die Präsentation hat auf der Folie 1 zwei Bildplatzhalter mit den Alternativtexten 'CW1' und 'CW2'.
Zuerst wird das Kurvenfenster 'Bogen' selektiert und in das Bild mit dem Alternativtext 'CW1' übertragen.
Dann wird das Kurvenfenster 'Gewicht' selektiert und in das Bild mit dem Alternativtext 'CW2' übertragen.

```
FileLoad("C:\imc\Dat\bogen.dat","",0)
CwNewWindow( bogen,"show")
CwNewChannel( "append new axis", bogen)
FileLoad("C:\imc\Dat\gewicht.dat","",0)
CwNewWindow( gewicht,"show")
CwNewChannel( "append new axis", gewicht)
screen= "screen"
result= PptOpenPresentation( "c:\imc\ppt\PräsentationCW3.pptx",1);
CwSelectWindow(bogen)
result= PptSetcurve(1,"CW1",2,0,screen); Kurvenfenster "Bogen" -> CW1
CwSelectWindow(gewicht)
result= PptSetcurve(1,"CW2",2,0,screen); Kurvenfenster "Gewicht" -> CW2
result= PptSavePresentation("c:\imc\ppt\test.pptx")
result= PptClosePresentation( )
```

Siehe auch:

PptSetCellText, PptSetText, PptSetPicture

PptSetPicture

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Überträgt das Bild in eine Form in der angegebenen Folie.

Deklaration:

```
PptSetPicture ( SlideIndex, TxAlternativeText, TxPictureFilename, SizeOption, Position ) -> Result
```

Parameter:

SlideIndex	Index der Folie. Die erste Folie hat den Index 1.
TxAlternativeText	Der Alternativtext des Bildes.
TxPictureFilename	Kompletter Pfadname der Bilddatei
SizeOption	Anpassung der Bildgröße.
	0 : Die Orginalgröße des Bildes wird verwendet.
	1 : Die Größe des Platzhalters wird verwendet.
	2 : Die Größe des Platzhalters wird verwendet. Dabei werden die Seitenverhältnisse des Orginalbildes beibehalten.
Position	Der Teil der Form (Platzhalter des Bildes), der bei der Skalierung der Form seine Position behält.
	0 : Oben links.
	1 : Unten rechts.
	2 : Zentriert.
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Über den 'SlideIndex' wird die Folie in der Präsentation ermittelt. Der Index entspricht der Position der Folie in der Präsentation.

In der angegebenen Folie wird nach dem Bild mit dem Alternativtext gesucht. Es muss ein Bildobjekt oder ein Bildplatzhalter sein.

Wurde das Bild gefunden, so wird das Bild aus der Datei in das Bild der Präsentation übertragen.

Mit 'SizeOption' = 0 wird das Bild aus der Datei mit seiner Orginalgröße in den Platzhalter übertragen.
Über den Parameter 'Position' wird festgelegt, an welcher Stelle des Platzhalters das Bild verankert wird.
Wenn das Orginalbild größer ist als der Platzhalter, kann es vorkommen, dass andere Formobjekte überdeckt werden.

Mit 'SizeOption' = 1 wird das Orginalbild in die Fläche des Bildplatzhalters eingepasst.
Es füllt den Platzhalterbereich vollständig aus. Der Parameter 'Position' ist nicht relevant.
Bei dieser Option kann es zu Verzerrungen des Bildes kommen.

Mit 'SizeOption' = 2 wird das Orginalbild unter Beibehaltung der Seitenverhältnisse in die Fläche des Bildplatzhalters eingepasst.
Über den Parameter 'Position' wird festgelegt, an welcher Stelle des Platzhalters das Bild verankert wird.

So definieren Sie in PowerPoint den Alternativtext für ein Bild:

1. Selektieren Sie auf der Folie das Bild-Objekt.
2. Klicken Sie mit der rechten Maustaste in das Element und wählen Sie "Grafik formatieren" aus.
3. Klicken Sie auf "Größe und Eigenschaften" und dann auf "Alternativtext".
4. Geben Sie im Feld "Beschreibung" einen Text für die Kennzeichnung ein, z.B. FAMOS_Pic1. Dieser Kennzeichnungstext wird in den Kit-Funktionen zum Suchen des Formobjektes benutzt.
5. **Wichtig!** Geben Sie den Kennzeichnungstext nicht im Feld "Titel" ein.
6. Wollen Sie den Alternativtext für seine ursprüngliche Funktion benutzen (barrierefreies PowerPoint), so geben Sie im Feld "Beschreibung" zuerst den Text für die Kennzeichnung ein, gefolgt von einem Semikolon (;). Dann kann der Text folgen, der für ein barrierefreies PowerPoint gedacht ist. Das Semikolon ist nicht Teil des Kennzeichnungstextes.

Beispiele:

Das Bild 'Tulips.jpg' wird in die Präsentation auf den Platzhalter mit dem Alternativtext 'Picture1' übertragen.
Dabei bleiben die Seitenverhältnisse des Orginalbildes erhalten. Es wird auf dem Platzhalter zentriert.

```
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
result= PptSetPicture( 1,"Picture1","c:\imc\ppt\Tulips.jpg",2,2)
result= PptSavePresentation("c:\imc\ppt\Presentation4.pptx")
result= PptClosePresentation( )
```

Siehe auch:

PptSetCellText, PptSetText, PptSetCurve

PptSetText

Anwendungsbereich: PowerPoint-Fernsteuerung

Verfügbar ab: Professional Edition (PowerPoint-Kit)

Der Text eines Textfeldes in der angegebenen Folie wird gesetzt.

Deklaration:

```
PptSetText ( SlideIndex, TxAlternativeText, TxContent ) -> Result
```

Parameter:

SlideIndex	Index der Folie. Die erste Folie hat den Index 1.
TxAlternativeText	Der Alternativtext des Textfeldes.
TxContent	Dieser Text wird in das Textfeld übertragen.
Result	Ergebnis:
	0 : erfolgreich.
	-1 : Es ist ein Fehler aufgetreten. Die Ursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Über den 'SlideIndex' wird die Folie in der Präsentation ermittelt. Der Index entspricht der Position der Folie in der Präsentation.

In der angegebenen Folie wird nach dem Textfeld mit dem Alternativtext gesucht.
Alle Textfelder, die durch das Kit gesetzt werden sollen, benötigen einen Alternativtext.

Wurde das Textfeld gefunden, so wird der übergebene Text in das Textfeld übertragen.

Die Formatierung des Textfeldes bleibt erhalten.

Hat das Textfeld mehrere Absätze, so wird der übergebene Text in Teilzeichenketten aufgespaltet.

Die Aufspaltung erfolgt an den Zeichen CR ("~013") , LF ("~010") oder NL ("~013"+ "~010").

Jede Teilzeichenkette wird dem korrespondierenden Absatz zugewiesen.

So definieren Sie in PowerPoint den Alternativtext für ein Textfeld:

1. Selektieren Sie auf der Folie das Form-Objekt.
2. Klicken Sie mit der rechten Maustaste in das Element und wählen Sie "Form formatieren" aus.
3. Klicken Sie auf "Größe und Eigenschaften" und dann auf "Alternativtext".
4. Geben Sie im Feld "Beschreibung" einen Text für die Kennzeichnung ein, z.B. FAMOS_Text1. Dieser Kennzeichnungstext wird in den Kit-Funktionen zum Suchen des Formobjektes benutzt.
5. **Wichtig!** Geben Sie den Kennzeichnungstext nicht im Feld "Titel" ein.
6. Wollen Sie den Alternativtext für seine ursprüngliche Funktion benutzen (barrierefreies PowerPoint), so geben Sie im Feld "Beschreibung" zuerst den Text für die Kennzeichnung ein, gefolgt von einem Semikolon (;). Dann kann der Text folgen, der für ein barrierefreies PowerPoint gedacht ist. Das Semikolon ist nicht Teil des Kennzeichnungstextes.

Beispiele:

Die Folie 1 enthält die Textfelder mit den Alternativtexten 'Marker1' und 'Marker2'. Das Textfeld 'Marker2' hat 3 Absätze.

```
cr="~013"
result= PptOpenPresentation( "c:\imc\ppt\Presentation1.pptx",1)
result= PptSetText(1,"Marker1","FAMOS Presentation")
measurementname ="Measurement 01"
measurementbegin="2017-09-25 14:30:00"
measurementend  ="2017-09-25 14:36:00"
textcontent=measurementname+cr+measurementbegin+cr+measurementend
result= PptSetText(1,"Marker2",textcontent)
result= PptSavePresentation("c:\imc\ppt\Presentation3.pptx")
result= PptClosePresentation( )
```

Siehe auch:

PptSetCellText, PptSetPicture, PptSetCurve

PulseDuration

Verfügbar ab: Professional Edition

Dauer bzw. Breite oder auch Frequenz von Pulsen wird über der Zeit bestimmt.

Deklaration:

```
PulseDuration ( Datensatz, Fenster [, Flanke] [, Berechnung] [, Randfaktor] ) -> Ergebnis
```

Parameter:

Datensatz	Datensatz
Fenster	Breite des Mittelungsintervalls in x-Einheiten, wird auf ganze Vielfache der Abtastzeit gerundet
Flanke	Erkennung der Flanke, die eine Puls beginnen lässt (optional , Standardwert: "0-1")
	"0-1" : Übergang von 0 auf 1 (ungleich 0). Die 1 ist die genaue Position des Pulsbeginns. Positive Flanke.
	"1-0" : Übergang von 1 (ungleich 0) auf 0 . Die 0 ist die genaue Position des Pulsbeginns. Negative Flanke.
	"+" : linear interpolierter Nulldurchgang bei positiver Flanke. Beim Übergang von <= 0 auf > 0 liegt die genaue Position zwischen diesen Werten.
	"-" : linear interpolierter Nulldurchgang bei negativer Flanke. Beim Übergang von > 0 auf <= 0 liegt die genaue Position zwischen diesen Werten.
	"ifa" : kompatibel zu imc Inline FAMOS: Alle Übergänge von 0 nach 1 (ungleich 0) und umgekehrt werden beachtet. Die durchschnittliche Pulsdauer wird gebildet aus allen vollständigen Impulsen im Fenster. Wird im Fenster kein Impuls abgeschlossen, so wird die letzte Pulsdauer zurückgeliefert oder, wenn die Fensterbreite mal der Anzahl der Fenster ohne Impuls größer als die letzte Pulsdauer ist, das Produkt aus Fensterbreite und -anzahl. Zeitlich danach liegende Pulse bleiben stets unberücksichtigt. Der Randfaktor wird ignoriert.
Berechnung	Was wird berechnet (optional , Standardwert: "t")
	"t" : Pulsdauer, angegeben in der x-Einheit der Eingangsdaten
	"n" : Pulsbreite, angegeben als Anzahl von Messpunkten, ggf. auch mit Nachkommastellen
	"f" : Pulsfrequenz, berechnet als Kehrwert der gemittelten Pulsdauer
	"rpm" : Drehzahl, berechnet als Kehrwert der gemittelten Pulsdauer, angegeben in Drehzahleinheiten für einen Geber mit einem Puls pro Umdrehung. Die x-Einheit der Eingangsdaten muss s sein.
Randfaktor	Am Beginn und Ende des Signals sind oft keine Flanken, das Signal befindet sich mitten in einem Puls. Ist die letzte angrenzende Pulsdauer multipliziert mit diesem Faktor größer gleich dem flankenlosen Randbereich, wird die letzte Pulsdauer weiterhin angenommen. Sonst wird null ausgegeben. (optional , Standardwert: 1.5)
Ergebnis	Ergebnis

Beschreibung:

Die Funktion arbeitet in einem Fenster von fester Breite und erzeugt je einen Ergebniswert am Ende des Fensters. Die durchschnittliche Pulsdauer wird gebildet aus allen Impulsen im Fenster. Sind Pulse nicht vollständig enthalten, werden sie anteilig nach enthaltener Zeit gezählt. Die Dauer eines jedern Pulses, dessen vordere oder hintere Flanke innerhalb des Intervalles liegt, wird gewichtet gemittelt, wobei der Anteil der Dauer des Pulses innerhalb des Fensters als Gewicht benutzt wird. Sowohl zeitlich davor wie auch zeitlich danach folgende Pulse werden berücksichtigt.

Die Ergebniswerte sind jeweils der Fensterbeginn. Wenn das Ergenis in Treppen dargestellt wird, dann umfasst jede Treppenstufe den Mittelungszeitraum.

Wenn Flanke = "imc Inline FAMOS", dann abweichender Algorithmus, s.o.

Ein Randfaktor liegt in typischen Anwendungen deutlich > 1.0. Bei kontinuierlicher Pulsfolge bewirkt er, dass das Ergebnis am Rand keine Sprünge auf null enthält.

Mit Randfaktor = 0.0 wird das Fortsetzen am Rand unterdrückt. Liegen am Rand innerhalb einer Fensterbreite keine Pulse vor, wird das Ergebnis an diesen Stellen zu null.

Die Eingangsdaten sind äquidistant und können Events und Segmente haben.

Beispiele:

Periodendauer aus nicht geformtem Pulssignal

```
pulses = ( stri ( signal, 3, 2 ) > 0 )
duration = pulseDuration ( pulses, 0.1)
```

Drehzahl aus Pulsfolge

```
rpm = pulseDuration ( pulses, 0.1, "0-1", "rpm")
```

Pulsfrequenz aus sinusförmigem Signal

```
f = pulseDuration ( sine, 0.1, "+", "f")
```

Siehe auch:

OtrTachoToSpeed, Pulse

PyCallFunction

Anwendungsbereich: Python-Fernsteuerung

Verfügbar ab: Professional Edition (Python-Kit)

Eine Python-Funktion wird ausgeführt.

Deklaration:

```
PyCallFunction ( TxModulName, TxFunktionsName [, Arg1] [, Arg2] [, Arg3] [, Arg4] [, Arg5] [, Arg6] [, Arg7] [, Arg8] [, Arg9] [, Arg10] [, Arg11] [, Arg12] [, Arg13] ) -> Ergebnis
```

Parameter:

TxModulName	Name des Python-Moduls, in dem die auszuführende Funktion definiert ist.
TxFunktionsName	Name der auszuführenden Funktion
Arg1	Argument #1 (optional) (optional)
Arg2	Argument #2 (optional) (optional)
Arg3	Argument #3 (optional) (optional)
Arg4	Argument #4 (optional) (optional)
Arg5	Argument #5 (optional) (optional)
Arg6	Argument #6 (optional) (optional)
Arg7	Argument #7 (optional) (optional)
Arg8	Argument #8 (optional) (optional)
Arg9	Argument #9 (optional) (optional)
Arg10	Argument #10 (optional) (optional)
Arg11	Argument #11 (optional) (optional)
Arg12	Argument #12 (optional) (optional)
Arg13	Argument #13 (optional) (optional)
Ergebnis	Von der Funktion zurückgeliefertes Ergebnis

Beschreibung:

Zunächst wird geprüft, ob ein Modul mit dem angegebenen Namen bereits in den Interpreter importiert wurde. Falls nicht, wird der Import durchgeführt.

- Bei Untermodulen eines Pakets muss die übliche "Punkt-Notation", also z.B. "Paketname.Untermodulname" verwendet werden.
- Für in Python fest eingebaute Funktionen ("built-in functions") ist ein leerer Text oder "builtins" anzugeben.
- Falls das angegebene Modul sich nicht im Python-Standard-Suchpfad befindet, muss vorher die Funktion PyConfig("Sys.Path.Append", ...) aufgerufen werden.

Anschließend wird die Funktion mit dem angegebenen Namen gesucht und, wenn vorhanden, die übergebenen Parameter in Python-Variablen konvertiert und die Funktion damit ausgeführt.

- Für die Parameter-Konvertierung gelten die selben Regeln wie bei der Funktion PySetVar() beschrieben.

Wenn die Funktion eine Rückgabe liefert, wird das Ergebnis in eine FAMOS-Variable konvertiert.

- Für die Rückkonvertierung gelten die selben Regeln wie bei der Funktion PyGetVar() beschrieben.

Anmerkungen:

- Wenn eine Python-Funktion nicht direkt aufgerufen werden kann, weil z.B. die Parametertypen nicht von/nach FAMOS konvertierbar sind, kann ein kleines Python-Skript Abhilfe schaffen, in dem die gewünschte Funktion gekapselt wird. Zuerst werden dann mit PySetVar() die Parameter übertragen, dann das Skript mit PyRunFile() oder wiederholte PyRun()-Befehle aufgerufen und anschließend die Ergebnisse mit PyGetVar() die Ergebnisse abgeholt. Das Skript kümmert sich dann um die Aufbereitung der Parameter, ruft die Funktion auf und konvertiert die Ergebnisse in eine Form, die mit PyGetVar() abgeholt werden kann.
- Falls die Python-Laufzeitumgebung noch nicht geladen ist, wird diese zunächst durch einen impliziten Aufruf der Funktion PyInit()

initialisiert.
- Im Fehlerfall (z.B. Modul oder Funktion unbekannt, falsche Parametertypen) wird ein Laufzeitfehler erzeugt.

Beispiele:

Die eingebaute Python-Funktion 'divmod' liefert Quotient und Rest der ganzzahligen Division der beiden Argumente als Tupel mit 2 Elementen. Das Ergebnis-Tupel wird in einen FAMOS-Datensatz der Länge 2 konvertiert.

```
result = PyCallFunction("", "divmod", 13, 3)
; identisch zu PyCallFunction("builtins", "divmod", 13, 3)
quotient_= result[1]  ; => 4
remainder= result[2]  ; => 1
```

Vorstehender Code ist funktionsgleich zu:

```
PySetVar("", "a", 13)
PySetVar("", "b", 3)
PyRun("result = divmod(a,b)")
result = PyGetVar("", "result")
quotient_= result[1]
remainder= result[2]
```

Aufruf von Funktionen aus externen Bibliotheken, hier am Beispiel von 'NumPy'. Das nachfolgende Beispiel verwendet die FFT-Funktion von 'NumPy' und vergleicht das Ergebnis mit der in FAMOS eingebauten FFT()-Funktion.

```
; Create test signal
n = 1024
signal = sin(Ramp(0,0.01,n))

; Apply FFT from 'NumPy'
PyConfig("Converter.PyArrayType", "ndarray")
res = PyCallFunction("numpy.fft", "rfft", signal )
; Convert Cartesian to Polar coordinates
res = Pol(res)
; Undo normalization
res.b = res.b/n
; Calculate frequency line distance
freq = PyCallFunction("numpy.fft", "rfftfreq", ToInt(n))
res = xdel(res, (freq[2]-freq[1])/xdel?(signal))

; Compare with FAMOS FFFT
FFTOption 0 0
resFAMOS = FFT(signal)
Verify(Equal(resFAMOS, res,  1e-10, "absolute", 1e-10))
```

Berechnung des unteren und oberen Quartils sowie des Medians mit 'NumPy':

```
signal = ...
PyConfig("Converter.PyArrayType", "ndarray")
percs = PyCallFunction("numpy", "percentile", signal, [25, 50, 75])
lower_q = percs[1]
median = percs[2]
upper_q = percs[3]
```

Wenn zusätzlich noch die Interpolationsart für die "percentile"-Funktion gesetzt werden soll (über den benannten optionalen Parameter 'interpolation'), ist ein direkter Aufruf mit PyCallFunction() nicht mehr möglich, sondern muss mit PySetVar()/PyRun()/PyGetVar() nachgebildet werden:

```
...
PySetVar("", "signal", signal)
PyRun("import numpy")
PyRun("percs = numpy.percentile(signal, [25, 50, 75], interpolation='nearest')")
percs = PyGetVar("", "percs")
...
```

Siehe auch:

PyGetVar, PySetVar, PyRun, PyRunFile, PyConfig

PyConfig

Anwendungsbereich: Python-Fernsteuerung

Verfügbar ab: Professional Edition (Python-Kit)

Steuerung des Verhaltens des eingebetteten Python-Interpreters

Deklaration:

```
PyConfig ( TxOption, TxValue ) -> Erfolg
```

Parameter:

TxOption	Name der zu setzenden Option.
	"Converter.PyArrayType" : Gibt an, in welchen Python-Datentyp ein FAMOS-Datensatz bei der Übertragung nach Python konvertiert wird.
	"Sys.Path.Append" : Fügt dem Modul-Suchpfad des Python-Interpreters ein weiteres Verzeichnis hinzu.
TxValue	Wert der zu setzenden Option.
	"Converter.PyArrayType" : Feld-Datentyp in Python
	"list" [Standard] Python Standard-Datentyp 'list'. **"ndarray"** Datentyp 'numpy.ndarray' (definiert in der 'NumPy'-Erweiterungsbibliothek). Wenn die 'NumPy'-Bibliothek nicht vorhanden ist, gibt die Funktion eine 0 zurück.
Erfolg	1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Python-Datentyp für FAMOS-Datensätze ("Converter.PyArrayType")

Diese Option bestimmt, welcher Python-Datentyp bei der Übertragung von FAMOS-Datensätzen nach Python verwendet wird. Dies betrifft die Funktionen PySetVar() und PyCallFunction(). Standardmäßig wird der in Python eingebaute Standard-Datentyp 'list' verwendet. Dieser ist jedoch für die Speicherung und Verarbeitung größerer Datenmengen nicht effektiv.

Alternativ kann der Datentyp 'numpy.ndarray' verwendet werden, der in der weit verbreiteten 'NumPy'- Erweiterungsbibliothek definiert ist. Dieser Typ ist optimiert für die effiziente Speicherung und schnelle Verarbeitung von mehrdimensionalen Feldern. Der Datentyp kann nur eingestellt werden, wenn das 'NumPy'-Paket auch installiert ist, ansonsten liefert die Funktion einen Fehler.

Eine genaue Auflistung, wie die verschiedenen FAMOS-Datentypen in Python umgesetzt werden, finden Sie in der Beschreibung der Funktion PySetVar().

Modul-Suchpfad ("Sys.Path.Append")

Der Modul-Suchpfad enthält eine Liste von Verzeichnissen, in denen der Python-Interpreter nach zu importierenden Modulen sucht. Der Modul-Suchpfad ist in der Python-Variablen "sys.path" abgelegt. Nach dem Start des Interpreters enthält diese Variable die durch die Python-Installation festgelegten Standardpfade sowie das aktuelle Arbeitsverzeichnis.

Der Aufruf

```
PyConfig("Sys.Path.Append", "c:\my\folder")
```

ist äquivalent zum folgenden Python-Code:

```
>>> import sys
>>> sys.path.append('c:\\my\\folder')
```

Anmerkungen:

- Die Funktion wird benötigt, wenn nachfolgend Python-Module verwendet werden sollen, die sich nicht in den Python-Standardverzeichnissen befinden.
- Falls die Python-Laufzeitumgebung noch nicht geladen ist, wird diese zunächst durch einen impliziten Aufruf der Funktion PyInit() initialisiert.

Multithreading: Alle Funktionen des Python-Kits dürfen in beliebigen Ausführungs-Threads (BEGIN_PARALLEL) aufgerufen werden, wirken aber Thread-übergreifend und global. Alle Aufrufe werden intern zunächst in den FAMOS-Haupt-Thread verschoben und von dort ausgeführt, da die Python-Laufzeitumgebung keine parallelen Aufrufe aus unterschiedlichen Threads unterstützt.

Beispiele:

Eine Auswertung verwendet die Zusatzbibliothek 'NumPy'. Der Standard-Feld-Typ wird entsprechend zu Beginn auf 'ndarray' gesetzt.

```
IF 0 = PyConfig("Converter.PyArrayType", "ndarray")
   BoxMessage("Error", "NumPy-Package not found. Please install first (see https://numpy.org)."
   EXITSEQUENCE
END
signal = ...
; Apply FFT from 'NumPy'
res = PyCallFunction("numpy.fft", "rfft", signal )
```

Ein FAMOS-Projekt enthält ein selbst geschriebenes Python-Modul mit den Namen "myfunctions.py", aus dem die Funktion "sum3" aufgerufen werden soll, die aus 3 Parametern die Summe bildet. Vor Verwendung des Moduls muss das FAMOS-Projektverzeichnis dem Suchpfad hinzugefügt werden.

```
projectPath = GetOption("Dir.CurrentProject")
PyConfig("Sys.Path.Append", projectPath)
sum = PyCallFunction("myfunctions", "sum3", 1, 2.2, 3)
; sum = 6.2
```

Siehe auch:

PyInit, PySetVar, PyRun, PyCallFunction

PyGetVar

Anwendungsbereich: Python-Fernsteuerung

Verfügbar ab: Professional Edition (Python-Kit)

Der Inhalt einer Python-Variablen wird abgefragt.

Deklaration:

```
PyGetVar ( TxModulName, TxVariablenName ) -> Inhalt
```

Parameter:

TxModulName	Name des Python-Moduls, in dessen Kontext die Variable existiert.
TxVariablenName	Name der abzufragenden Variable
Inhalt	Inhalt der abgefragten Variable

Beschreibung:

Der Inhalt der adressierten Python-Variable wird ermittelt.

Zunächst wird geprüft, ob das angegebene Modul bereits in den Interpreter importiert wurde. Falls nicht, wird der Import durchgeführt. Bei einem leeren Namen wird das Hauptmodul ("__main__") verwendet. Bei Untermodulen eines Pakets muss die übliche "Punkt-Notation", also z.B. "Paketname.Untermodulname" verwendet werden.

Anschließend wird die Variable mit dem angegebenen Namen gesucht und, wenn vorhanden, der Inhalt der Variablen ermittelt.

Beispiel:

Abfragen einer globalen Variablen mit dem Namen 'test':

```
PyRun("test = 2.3")
test = PyGetVar("", "test")
; identisch zu: PyGetVar("__main__", "test")
```

Wenn es sich nicht um eine globale Variable auf Modulebene handelt, sondern um ein Attribut einer Klassen-Instanz, so muss sie über die übliche "Punkt-Notation" in Python angegeben werden, also z.B. "KlassenInstanzName.Attributname"

Beispiel:

Angenommen, das Python-Modul 'persons' enthält eine Definition der Klasse 'person', die u.a. die Attribute 'name' und 'age' enthält. Der folgende Code erzeugt eine Instanz der Klasse 'person' und fragt später die Attribute ab:

```
PyRun("import persons")
PyRun("harry = persons.person()")
...
TxName = PyGetVar("", "harry.name")
Age = PyGetVar("", "harry.age")
```

Anmerkungen:

- Falls die Python-Laufzeitumgebung noch nicht geladen ist, wird diese zunächst durch einen impliziten Aufruf der Funktion PyInit() initialisiert.
- Im Fehlerfall (z.B. unbekanntes Modul, unbekannter Variablenname oder nicht unterstützter Datentyp) wird ein Laufzeitfehler erzeugt.

Datentypen

Der Datentyp der erzeugten FAMOS-Variable wird automatisch aus dem Datentyp der Python-Variable bestimmt.

Dabei gelten folgende Regeln:

Python	FAMOS
Standard-Typen	
'NoneType'	Datensatz mit Länge 0, Datenformat: 8 Byte Reell (wie Konstante 'EMPTY')
'int'	Einzelwert, Datenformat: 8 Byte Integer mit Vorzeichen
'float'	Einzelwert, Datenformat: 8 Byte Reell
'complex'	Komplexer Datensatz (RI) mit Länge 1, Datenformat: 8 Byte Reell
'bool'	Einzelwert, Datenformat: 8 Byte Reell
'str'	Text

Container-Typen 'tuple', 'list', 'range', 'set', 'frozenset'	
mit Elementzahl 0	Datensatz mit Länge 0, Datenformat: 8 Byte Reell
alle Elemente vom Typ 'float'	Datensatz, Datenformat: 8 Byte Reell
alle Elemente vom Typ 'int'	Datensatz, Datenformat: 8 Byte Integer mit Vorzeichen
alle Elemente vom Typ 'bool'	Datensatz, Datenformat: 8 Byte Reell
alle Elemente vom Typ 'complex'	Komplexer Datensatz (RI) mit Länge 1, Datenformat: 8 Byte Reell
alle Elemente vom Typ 'str'	Textfeld
alle Elemente ein 1-dimensionaler Container, wobei alle Container die selbe Länge N und alle Elemente den selben numerischen Typ(float, int, complex) haben	Segmentierter Datensatz, Segmentlänge N
andere Element-Typen oder verschiedene Typen	Datengruppe. Die Kanäle der Gruppe erhalten die Namen "Item1", "Item2"..., der Inhalt entsprechend den hier definierten Konvertierungsregeln. Achtung bei 'set' - die Elemente haben keine definierte Reihenfolge, die Reihenfolge der Kanäle in der Gruppe ist dann zufällig.
Sonstige Container-Typen	
'dictionary'	Datengruppe. Aus jedem Schlüssel/Wert-Paar wird ein Kanal gebildet. Der Name des Kanals wird aus dem Schlüsselnamen gebildet, der Inhalt des Kanals aus dem jeweiligen Wert entsprechend den hier definierten Konvertierungsregeln erzeugt.
'bytes', 'bytearray'	Datensatz, 1 Byte Integer unsigned

NumPy-Arrays ('numpy.ndarray')

Dieser Datentyp ist im weit verbreiteten 'NumPy'-Package definiert. Er ist optimiert für die effiziente Speicherung und schnelle Verarbeitung von mehrdimensionalen Feldern.

Es werden homogene Felder mit Dimension 1 oder 2 unterstützt. Bei Dimension 2 wird ein segmentierter Datensatz erzeugt, wobei die Zeilen des NumPy-Arrays die Segmente bilden.

Das Speicherlayout muss "C-" oder "F-zusammenhängend" (C_CONTIGUOUS, F_CONTIGUOUS) sein und die Daten in "Little-Endian-Byte order" abgelegt sein.

NumPy-Array-Elementtyp	**FAMOS**
'float64', 'float32'	Datensatz (Reell 8 Byte)
'float128', 'float16'	Nicht unterstützt
'int8', 'int16', 'int32'	Datensatz (Reell 8 Byte)
'int64'	Datensatz (Integer 8 Byte signed)
'uint8'	Datensatz (Integer 1 Byte unsigned)
'uint16', 'uint32'	Datensatz (Reell 8 Byte)
'uint64'	Nicht unterstützt
'bool'	Datensatz (Reell 8 Byte)
'complex64', 'complex128'	Komplexer Datensatz (RI, Reell 8 Byte)
'str'	Textfeld
'datetime64', 'timedelta64'	Nicht unterstützt
'bytes' und alle anderen Typen	Nicht unterstützt

Beispiele:

Abfrage der Python-Version über das Attribut "hexversion" des "sys"-Moduls.

```
version = PyGetVar("sys", "hexversion") ; datatype of the result is "signed 8 byte integer"
major = BitShift(version, -24, 64) ; major is always 3
minor = BitAnd(BitShift(version, -16, 32), 0xFF, 64); minor is e.g. 8 or 9
```

Ermittlung des Python-Modul-Suchpfades. Das Ergebnis ist vom Typ Textfeld, jedes Element ist ein Verzeichnis.

```
pythonPath = PyGetVar("sys", "path")
```

Berechnung des unteren und oberen Quartils sowie des Medians mit 'NumPy':

```
PyConfig("Converter.PyArrayType", "ndarray")
signal = ...
PySetVar("", "signal", signal)
PyRun("import numpy")
PyRun("percs = numpy.percentile(signal, [25, 50, 75], interpolation='nearest')")
percs = PyGetVar("", "percs")
lower_quantil = percs[1]
median = percs[2]
upper_quantil = percs[3]
```

Ein Python-Dictionary wird in eine FAMOS-Datengruppe konvertiert:

```
PyRun("person = {'name' : 'John', 'age' : 36}")
person = PyGetVar("", "person")
```

Ergebnis ist eine Datengruppe mit 2 Elementen:

```
person
   |__  name    ("John")
   |__  age    (36)
```

Aus einer Datenstichprobe wird eine 'NormalDist'-Klasse aus dem 'statistics'-Modul gebildet, die eine Normalverteilung mit Mittelwert und Standardabweichung der Stichprobe repräsentiert. Mittelwert und Standardabweichung wird abgefragt, außerdem wird ein langer Zufallsdatensatz erzeugt, der die selbe Verteilung wie die Stichprobe aufweist:

```
PyRun("import statistics")
PySetVar("", "probe_samples" , [1, 1.2, 7.2, 15, 4.2, 6.1])
PyRun("nd = statistics.NormalDist.from_samples(probe_samples)")
mean_ = PyGetVar("","nd.mean")
stdev_ = PyGetVar("","nd.stdev")
PyRun("samples = nd.samples(10000, seed = 1)")
samples = PyGetVar("","samples")
```

Siehe auch:

PySetVar, PyRun, PyRunFile, PyCallFunction

PyInit

Anwendungsbereich: Python-Fernsteuerung

Verfügbar ab: Professional Edition (Python-Kit)

Die Python-Laufzeitumgebung wird in FAMOS geladen und der Interpreter initialisiert.

Deklaration:

```
PyInit ( [TxPythonVerzeichnis] ) -> Erfolg
```

Parameter:

TxPythonVerzeichnis	Heimat-Verzeichnis der zu verwendenden Python-Installation. Optional, wenn nicht angegeben oder leer, wird das Python-Verzeichnis automatisch bestimmt. (optional , Standardwert: "")
Erfolg	1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Ein expliziter Aufruf dieser Funktion ist nicht zwingend erforderlich. Alle FAMOS-Funktionen, die den Python-Interpreter verwenden, führen bei noch fehlender Initialisierung diese automatisch intern durch.

Wenn der Python-Interpreter zum Zeitpunkt des Aufrufs bereits initialisiert ist, tut die Funktion nichts.

Der explizite Aufruf der Funktion kann in den folgenden Anwendungsfällen sinnvoll sein:

- Vorgabe des Heimat-Verzeichnisses der zu verwendenden Python-Installation: Normalerweise kann dieser Parameter weggelassen werden, FAMOS verwendet dann automatisch die höchste unterstützte Version, die auf dem Rechner gefunden werden kann. Die explizite Angabe ist notwendig, wenn mehrere Python-Installationen auf dem Rechner vorhanden sind und eine definierte Version verwendet werden soll. Das Heimat-Verzeichnis einer Python-Installation ist das beim Setup ausgewählte Zielverzeichnis und enthält z.B. die ausführbare Datei 'python.exe'.
- Das Python-Heimat-Verzeichnis ist nicht in der PATH-Umgebungsvariable eingetragen.FAMOS kann dann Python nicht automatisch finden und der Pfad muss hier explizit angegeben werden.
- Test auf eine kompatible Python-Installation vor Beginn der eigentlichen Auswertung, um im Fehlerfall (z.B. nicht unterstützte Python-Version) eine sinnvolle Fehlerbehandlung durchführen zu können. Dies könnte eine z.B. eine Nachricht an den Anwender sein mit der Bitte, die Python-Installation zu prüfen, gefolgt vom geordneten Beenden der Auswertung. Anderenfalls würde ein solches grundlegendes Problem erst während der Ausführung der eigentlichen Auswertung auffallen und zu einem Laufzeitfehler, also einem unkontrollierten Abbrechen der Sequenz, führen.

Multithreading: Alle Funktionen des Python-Kits dürfen in beliebigen Ausführungs-Threads (BEGIN_PARALLEL) aufgerufen werden, wirken aber Thread-übergreifend und global. Alle Aufrufe werden intern zunächst in den FAMOS-Haupt-Thread verschoben und von dort ausgeführt, da die Python-Laufzeitumgebung keine parallelen Aufrufe aus unterschiedlichen Threads unterstützt.

Beispiele:

Zu Beginn einer Auswertung, die zwingend Python 3.9 benötigt, wird das Vorhandensein einer passenden Python-Installation geprüft:

```
IF 0 = PyInit()
   BoxMessage("Can't start Python!", GetLastError(), "!1")
   EXITSEQUENCE
END
version = PyGetVar("sys", "hexversion")
; "sys.hexversion" liefert die Python-Version in hexadezimal kodierter Form
major = BitShift(version, -24, 64)
minor = BitAnd(BitShift(version, -16, 32), 0xFF, 64)
IF major <> 3 OR minor <> 9
   BoxMessage("Wrong Python version!", "Sorry, Python 3.9 requested!", "!1")
   PyTerminate()
   EXITSEQUENCE
END
...
```

Auf einem PC mit mehreren Python-Installationen wird die gewünschte Version durch Angabe des Heimatverzeichnisses ausgewählt:

```
IF 0 = PyInit("c:\python\version_3_8")
   BoxMessage("No Python installation found!", GetLastError(), "!1")
   EXITSEQUENCE
END
...
```

Siehe auch:

PyTerminate, PyConfig, PySetVar, PyRun

PyRun

Anwendungsbereich: Python-Fernsteuerung

Verfügbar ab: Professional Edition (Python-Kit)

Der angegebene Python-Code wird ausgeführt.

Deklaration:

```
PyRun ( TxCode ) -> Erfolg
```

Parameter:

TxCode	Der auszuführende Python-Code. Erlaubte Typen: Text, Textfeld
Erfolg	1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Der angegebene Python-Skript-Code wird an den Python-Interpreter übergeben und ausgeführt.

Es können auch mehrere Codezeilen in einer Textvariablen übergeben werden, diese müssen durch ein LineFeed-Zeichen (ASCII-Code 10, "~010") getrennt werden.

Wenn ein Textfeld übergeben wird, wird jedes Element als eine Codezeile interpretiert.

Anmerkungen:

- Zum Ausführen von komplexen Skripten ist meist die Funktion PyRunFile() besser geeignet.
- Falls die Python-Laufzeitumgebung noch nicht geladen ist, wird diese zunächst durch einen impliziten Aufruf der Funktion PyInit() initialisiert.

Beispiele:

Die Python-Funktion 'sqrt' aus dem 'math'-Modul zur Berechnung der Quadratwurzel wird aufgerufen. Die folgenden 3 Aufrufbeispiele sind gleichwertig:

```
PyRun("import math")
PyRun("x = math.sqrt(2)")
```

```
PyRun("import math~010x = math.sqrt(2)")
```

```
codeLines = ["import math", "x = math.sqrt(2)"]
PyRun(codeLines)
```

Die Ergebnis der Berechnung kann dann anschließend abgefragt werden:

```
x = PyGetVar("", "x")
```

Alternative Berechnung:

```
x = PyCallFunction("math", "sqrt", 2)
```

Die Python-Standardausgabe wird in das FAMOS-Ausgabefenster umgeleitet. Die print()-Funktion kann daher verwendet werden, um dort Texte anzuzeigen:

```
PyRun("person = {'name': 'Harry', 'age': 42}")
PyRun("print('My name is',person['name'],', age ', person['age'])")
; => FAMOS-Ausgabefenster: "My name is  Harry , age  42"
```

Zu Beginn einer Auswertung in Python, die Funktionen einer selbst geschriebenen Bibliothek 'myfunctions.py' verwendet, wird geprüft, ob das Modul auf dem aktuellen PC überhaupt vorhanden ist:

```
IF PyRun("import myfunctions") = 0
   BoxMessage("Error", "Please install the Python module 'myfunctions' first!", "!1")
   EXITSEQUENCE
END
```

Aus einer Datenstichprobe wird eine 'NormalDist'-Klasse aus dem 'statistics'-Modul gebildet, die eine Normalverteilung mit Mittelwert und Standardabweichung der Stichprobe repräsentiert. Mittelwert und Standardabweichung wird abgefragt, außerdem wird ein langer Zufallsdatensatz erzeugt, der die selbe Verteilung wie die Stichprobe aufweist:

```
PyRun("import statistics")
PySetVar("", "probe_samples" , [1, 1.2, 7.2, 15, 4.2, 6.1])
PyRun("nd = statistics.NormalDist.from_samples(probe_samples)")
mean_ = PyGetVar("","nd.mean")
```

```
stdev_ = PyGetVar("","nd.stdev")
PyRun("samples = nd.samples(10000, seed = 1)")
samples = PyGetVar("","samples")
```

Siehe auch:

PyRunFile, PyCallFunction, PyGetVar, PySetVar

PyRunFile

Anwendungsbereich: Python-Fernsteuerung

Verfügbar ab: Professional Edition (Python-Kit)

Die angegebene Python-Skript-Datei wird ausgeführt.

Deklaration:

```
PyRunFile ( TxDateiName [, TxParameter] ) -> Erfolg
```

Parameter:

TxDateiName	Kompletter Pfadname des auszuführenden Python-Skripts
TxParameter	Optionale Liste der zu übergebenden Parameter (optional)
Erfolg	1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die angegebene Python-Skript-Datei wird an den Python-Interpreter übergeben und ausgeführt.

Die (optionale) Liste der Parameter muss in der selben Syntax angeben werden, als wenn das Skript direkt an den Python-Interpreter ('python.exe') übergeben wird. Die einzelnen Parameter werden also durch Leerzeichen getrennt, Parameter mit Leerzeichen müssen in zusätzliche Gänsefüßchen eingeschlossen werden.

Der Aufruf in der Windows-Eingabeaufforderung

```
   python c:\py_scripts\myscript.py 2 "Hello world"
```

wäre insofern identisch zu

```
   PyRunFile("c:\py_scripts\myscript.py", "2 ""Hello world"""
```

Wichtiger Unterschied zwischen beiden Aufrufen: Der erste Aufruf verwendet eine neue, eigenständige Python-Instanz. Im zweiten Aufruf wird das Skript im Kontext der durch FAMOS verwalteten Python-Instanz ausgeführt, bereits importierte Module oder erzeugte globale Variablen stehen dem Skript also zur Verfügung. Umgekehrt bleiben durch das Skript erzeugte globale Variablen nach der Ausführung erhalten.

Anmerkungen:

- Falls die Python-Laufzeitumgebung noch nicht geladen ist, wird diese zunächst durch einen impliziten Aufruf der Funktion PyInit() initialisiert.
- Die Skript-Datei muss im Zeichensatz UTF-8 kodiert sein. Wenn nur ASCII-Zeichen im Bereich bis 127 verwendet werden, entspricht dies dem Standard-ANSI-Zeichensatz.

Beispiele:

Die Python-Skriptdatei 'c:\py_scripts\Add2Numbers.py" addiert 2 als Parameter übergebene Zahlen und hat den folgenden Inhalt:

```
import sys
sum = float(sys.argv[1]) + float(sys.argv[2])
```

Aufruf in FAMOS:

```
PyRunFile("c:\py_scripts\Add2Numbers.py","1.2 3.2")
sum = PyGetVar("","sum") ; => sum = 4.4
```

Die in der Python-Standard-Installation enthaltene Skript-Datei 'untabify.py' bearbeitet die als Parameter übergebene(n) Datei(en) und ersetzt Tabulator-Zeichen durch eine wählbare Anzahl von Leerzeichen.

```
; replace each tab in "file1.txt" with 3 spaces:
PyRunFile("C:\Python\Tools\scripts\untabify.py", "-t 3 ""c:\my files\file1.txt""")
```

Siehe auch:

PyRun, PyCallFunction, PyGetVar, PySetVar

PySetVar

Anwendungsbereich: Python-Fernsteuerung

Verfügbar ab: Professional Edition (Python-Kit)

Der Inhalt einer Python-Variablen wird gesetzt.

Deklaration:

```
PySetVar ( TxModulName, TxVariablenName, Inhalt )
```

Parameter:

TxModulName	Name des Python-Moduls, in dessen Kontext die Variable gesetzt werden soll.
TxVariablenName	Name der zu setzenden Variable.
Inhalt	Neuer Inhalt.

Beschreibung:

Der adressierten Python-Variable wird der angegebene Inhalt zugewiesen.

Zunächst wird geprüft, ob das angegebene Modul bereits in den Interpreter importiert wurde. Falls nicht, wird der Import durchgeführt. Bei einem leeren Namen wird das Hauptmodul ("__main__") verwendet. Bei Untermodulen eines Pakets muss die übliche "Punkt-Notation", also z.B. "Paketname.Untermodulname" verwendet werden.

Anschließend wird die Variable mit dem angegebenen Namen gesucht. Wenn sie bereits existiert, wird sie überschrieben, ansonsten neu erzeugt.

Beispiel: Erzeugen einer globalen Variablen mit dem Namen 'test' und dem Datentyp 'float'.

```
PySetVar("", "test", 2.3)
; identisch zu: PySetVar("__main__", "test", 2.3)
```

Wenn es sich nicht um eine globale Variable auf Modulebene handelt, sondern um ein Attribut einer Klassen-Instanz, so muss sie über die übliche "Punkt-Notation" in Python angegeben werden, also z.B. "KlassenInstanzName.Attributname"

Beispiel: Angenommen, das Python-Modul 'persons' enthält eine Definition der Klasse 'person', die u.a. die Attribute 'name' und 'age' enthält. Der folgende Code erzeugt eine Instanz der Klasse 'person' und setzt die Attribute:

```
PyRun("import persons")
PyRun("harry = persons.person()")
PySetVar("", "harry.name", "harry")
PySetVar("", "harry.age", 18)
```

Anmerkungen:

- Falls die Python-Laufzeitumgebung noch nicht geladen ist, wird diese zunächst durch einen impliziten Aufruf der Funktion PyInit() initialisiert.
- Im Fehlerfall (z.B. unbekanntes Modul, unzulässiger Variablenname oder Datentyp) wird ein Laufzeitfehler erzeugt.
- Multithreading: Alle Funktionen des Python-Kits dürfen in beliebigen Ausführungs-Threads (BEGIN_PARALLEL) aufgerufen werden, wirken aber Thread-übergreifend und global. Alle Aufrufe werden intern zunächst in den FAMOS-Haupt-Thread verschoben und von dort ausgeführt, da die Python-Laufzeitumgebung keine parallelen Aufrufe aus unterschiedlichen Threads unterstützt.

Datentypen

Der Python-Datentyp der erzeugten Variable wird automatisch aus dem Typ und Format des 3. Parameters bestimmt.

Der mit PyConfig("Converter.PyArrayType", ...) konfigurierte Standard-Container-Typ wird berücksichtigt (entweder 'list' (Standard) [1] oder 'numpy.nddarray' [2]).

Dabei gelten folgende Regeln:

FAMOS-Datentyp	**Python-Datentyp**
Einzelwerte	
Integer 8 Byte (full-scale)	Ganze Zahl ('int')
Sonstige numerische Datenformate	Reelle Zahl, 8 Byte ('float')
Normale Datensätze	

Reell 8 Byte mit Länge = 0 (Konstante 'EMPTY')	None ('NoneType')
Integer 8 Byte (full-scale)	[1]: Liste ('list' [..'int'..]) [2]: NumPy-Array ('numpy.ndarray' [..'int64'..])
Integer 1 Byte unsigned (full-scale)	Bytefeld ('bytearray')
Sonstige numerische Datenformate	[1]: Liste ('list' [..'float'..]) [2]: NumPy-Array ('numpy.ndarray' [..'float64'..])
TimeStampASCII und andere Sonderformate	nicht unterstützt
2-komponentige Datensätze	
Komplex, kartesisch, Länge = 1	Komplexe Zahl ('complex')
Komplex, kartesisch, Länge <> 1	[1]: Liste kompl. Zahlen ('list' [..'complex'..]) [2]: NumPy-Array ('numpy.ndarray' [..'complex128'..])
Komplex, Betrag/Phase	nicht unterstützt
XY	nicht unterstützt
Strukturierte Daten (Events/Segmente)	
Segmente ... Integer 1 Byte unsigned (full-scale)	[1]: Liste von Bytefeldern ('list' [..'bytearray'..]). Jedes Bytefeld entspricht einem Segment. [2]: 2-dimensionales NumPy-Array von Bytes ('numpy.ndarray' [..'uint8'..]). Jedes Segment bildet eine Zeile der Matrix.
Segmente ... sonstige Formate	[1]: Liste von Listen ('list' [..'list'..]). Jedes Listenelement entspricht einem Segment. [2]: 2-dimensionales NumPy-Array ('numpy.ndarray'). Jedes Segment bildet eine Zeile der Matrix. Der Datentyp der Listen-/Feldelemente ergibt sich wie vorstehend beschrieben.
Events	nicht unterstützt
Sonstige Datentypen	
Text	String ('str')
Textfeld	Liste von Strings ('list' [..'str'..])
Gruppe	Dictionary ('dict'). Jeder Kanal der Gruppe wird in ein Schlüssel/Wert-Paar übersetzt. Der Schlüssel entspricht dem jeweiligen Kanalnamen.

Anmerkung zum Typ 'numpy.ndarray'

Dieser Datentyp ist im weit verbreiteten 'NumPy'-Package definiert. Er ist optimiert für die effiziente Speicherung und schnelle Verarbeitung von mehrdimensionalen Feldern. Die erzeugten NumPy-Arrays sind "C-zusammenhängend" (C_CONTIGUOUS) gespeichert.

Beispiele:

Erzeugen von globalen Variablen vom Datentyp 'float' und 'int':

```
PySetVar("", "test_float", 2.3)
PySetVar("", "test_int", ToInt(2))
```

Erzeugen von globalen Variablen vom Datentyp 'list' und 'numpy.ndarray' (Datentyp der Elemente: 'float'):

```
PySetVar("", "test_list", [1,2,6,7])
PyConfig("Converter.PyArrayType", "ndarray")
PySetVar("", "test_ndarray", [1,2,6,7])
```

Erzeugen einer Python-Liste mit Texten:

```
PySetVar("", "test_str_list", ["The", "answer", "is", "42"])
```

Erzeugen eines Python-Dictionarys ('dict') aus einer FAMOS-Datengruppe:

```
person:name = "Harry"
person:age = ToInt(42)
PySetVar("", "test_dict", person)
PyRun("print(test_dict)")
; => FAMOS-Ausgabefenster: "test_dict = {'name': 'Harry', 'age': 42}"
```

Siehe auch:

PyGetVar, PyRun, PyRunFile, PyConfig, PyCallFunction

PyTerminate

Anwendungsbereich: Python-Fernsteuerung

Verfügbar ab: Professional Edition (Python-Kit)

Der Python-Interpreter wird beendet.

Deklaration:

```
PyTerminate ( )
```

Parameter:

Beschreibung:

Der zuvor durch PyInit() oder eine andere Python-Zugriffsfunktion initialisierte Python-Interpreter wird beendet.

Ein expliziter Aufruf dieser Funktion ist nicht zwingend erforderlich. Das Terminieren des Python-Interpreters erfolgt automatisch beim Beenden von FAMOS.

Eine mögliche Anwendung ist das zwischenzeitliche Terminieren des Interpreters, um danach z.B. mit PyInit() eine neue, "saubere" Interpreter-Instanz zu starten. Dies ist zwar prinzipiell erlaubt, aber unter Umständen gefährlich und daher nicht empfohlen. Python-Erweiterungsmodule von Drittanbietern haben unter Umständen Probleme nach einem solchen "Interpreter-Neustart", da diese architekturbedingt manchmal nicht sauber entladen werden können. Ein Beispiel ist die Bibliothek 'NumPy', die nach PyTerminate()/PyInit() möglicherweise nicht mehr korrekt funktioniert.

Wenn der Python-Interpreter zum Zeitpunkt des Aufrufs nicht initialisiert ist, tut die Funktion nichts.

Beispiele:

Zu Beginn einer Auswertung, die zwingend Python 3.9 benötigt, wird das Vorhandensein einer passenden Python-Installation geprüft. Wenn eine andere Version vorgefunden wird, wird die Python-Laufzeitumgebung gleich wieder beendet:

```
IF 0 = PyInit()
   BoxMessage("Can't start Python!", GetLastError(), "!1")
   EXITSEQUENCE
END
version = PyGetVar("sys", "hexversion")
; "sys.hexversion" liefert die Python-Version in hexadezimal kodierter Form
major = BitShift(version, -24, 64)
minor = BitAnd(BitShift(version, -16, 32), 0xFF, 64)
IF major <> 3 OR minor <> 9
   BoxMessage("Wrong Python version!", "Sorry, Python 3.9 requested!", "!1")
   PyTerminate()
   EXITSEQUENCE
END
...
```

Siehe auch:

PyInit

R_ChisqTest

Anwendungsbereich: R-Fernsteuerung

Verfügbar ab: Professional Edition (R-Kit)

Funktion zur Ausführung des Chi-Quadrat-Tests

Deklaration:

```
R_ChisqTest ( x [, correct] [, p] ) -> Result
```

Parameter:

x	Daten mit den Merkmalen der Stichprobe.
correct	Kontinuitätskorrektur (optional , Standardwert: 1)
	0 : Kontinuitätskorrektur wird nicht angewendet.
	1 : Kontinuitätskorrektur wird angewendet.(Standard)
p	Normaler Datensatz mit Wahrscheinlichkeiten. (nur im Anpassungstest relevant) (optional)
Result	Ergebnis des Chi-Quadrat-Tests ist eine Datengruppe mit den Elementen:
	statistic : Chi-Quadrat- Wert
	parameter : Anzahl der Freiheitsgrade
	p.value : p-Wert des Tests
	method : Die Zeichenfolge zeigt an, welcher Typ des Chi-Quadrat-Tests ausgeführt wurde.
	observed : Beobachtete Häufigkeiten
	expected : Die erwarteten Häufigkeiten unter der Nullhypothese.
	residuals : Pearson Residuen.
	stdres : Standard- Residuen.

Beschreibung:

Der Chi-Quadrat-Test gehört zur Gruppe der Hypothesentests auf der Grundlage einer Chi-Quadrat-Verteilung. Das Testkonzept besteht im Kern im Vergleich einer empirischen Häufigkeit mit einer theoretischen Häufigkeit. Er macht eine Aussage darüber, ob die beobachteten Häufigkeiten sich signifikant von denen unterscheiden, die man erwarten würde.

Testvoraussetzungen:

Die Stichprobe sollte mindestens 50 Untersuchungseinheiten umfassen.
Liegt die Stichprobengröße zwischen 20 und 50, so sollte die Kontinuitätskorrektur angewendet werden.
Die erwarteten Häufigkeiten in sämtlichen Zellen der Kontingenztabelle sollten größer als 5 sein.
Die Freiheitsgrade = (r-1) * (c-1) sollten grösser als 1 sein (r=Zeilen- und c=Spaltenanzahl der Kontingenztabelle). Ist die Bedingung nicht erfüllt, so sollte die Kontinuitätskorrektur angewendet werden.

Man unterscheidet zwischen den folgenden Tests:

Anpassungs- oder Verteilungstest

Der Anpassungstest prüft, ob eine empirisch gegebene Häufigkeitsverteilung durch eine bestimmte theoretische Verteilung zu beschreiben ist. Er testet, ob die Daten einer bestimmten Wahrscheinlichkeitsverteilung folgen. So kann z.B. überprüft werden, ob die Augenzahlen eines Würfels gleichverteilt sind oder ob ein konkretes empirisches Merkmal in der Grundgesamtheit normalverteilt ist.
Wenn der Parameter **'x' ein normaler Datensatz** ist, wird der Anpassungstest durchgeführt. Dabei wird 'x' als eine eindimensionale Kontingenztabelle behandelt. Die Einträge von 'x' sind die beobachteten Häufigkeiten. Es müssen positive Zahlen sein. Diese relativen Klassenhäufigkeiten werden dann mit den angegebenen Wahrscheinlichkeiten verglichen.
Der Anpassungstest testet immer gegen die Nullhypothese.
Die Nullhypothese lautet, dass die Merkmale von 'x' der angegebenen Wahrscheinlichkeitsverteilung 'p' entsprechen.
Die Wahrscheinlichkeiten können als Parameter 'p' übergeben werden. Die Werteanzahl im Datensatz von 'p' muss übereinstimmen mit 'x'. Ist 'p' nicht angegeben, so wird eine Gleichverteilung angenommen.
Der berechnete p-Wert 'p.value' ist die Wahrscheinlichkeit dafür, dass man unter der Nullhypothese die ermittelte Teststatistik beobachtet. Je kleiner der p-Wert , desto unwahrscheinlicher ist die Gültigkeit der Nullhypothese. Häufig wird die Nullhypothese bei einem p-Wert <= 0.05 (bzw. 5%) abgelehnt.

Unabhängigkeitstest

Beim Unabhängigkeitstest werden 2 diskrete Merkmale a und b einer Stichprobe auf stochastische Unabhängigkeit überprüft. Man interessiert sich dafür, ob die beiden Merkmale in einem Zusammenhang stehen oder unabhängig voneinander sind.
Die Hypothesen des Tests lauten:
Die Nullhypothese lautet, die Merkmale a und b sind unabhängig.
Die Alternativhypothese lautet, die Merkmale a und b sind nicht unabhängig.

Zur Ausführung des Unabhängigkeitstest muss der Parameter **'x' ein normaler segmentierter Datensatz** sein. Dieser wird in eine Matrix konvertiert und dann in eine Kontingenztabelle überführt. Auf der Grundlage dieser Kontingenztabelle wird der Unabhängigkeitstest durchgeführt. In der Kontingenztabelle bilden die Ausprägungen des einen Merkmals die Spalten und die Ausprägungen der zweiten Merkmals die Zeilen. In die Zellen werden die Häufigkeiten beider Merkmale geschrieben, also die Häufigkeiten, für die sowohl das Kriterium aus der Zeile wie auch das aus der Spalte zutrifft.
Beispiel: In der Kontigenztabelle wird das Kaufverhalten von Kunden abgebildet.
Merkmal a= Geschlecht, Merkmal b = Kaufverhalten, Stichprobe = 1000 Kunden
Die Häufigkeiten für das Kaufverhalten unterteilt nach Geschlecht und "kaufen" bzw. "nicht kaufen" werden in Form einer Matrix (segmentierter Datensatz) übergeben. Die Funktion bildet aus der Matrix die Kontingenztabelle, in dem sie die Randsummen ergänzt.

	Matrix				Kontingenztabelle		
Merkmal a / b	kaufen	nicht kaufen		Merkmal a / b	kaufen	nicht kaufen	Summe
Männer	180	170	=>	Männer	180	170	350
Frauen	240	410		Frauen	240	410	650
				Summe	420	580	1000

Die Kontinuitätskorrektur sollte gesetzt werden, wenn die Berechnung auf eine Kontingenztabelle mit 2 Spalten und 2 Zeilen angewendet wird.

Unter Verwendung des berechneten p-Wertes wird ein Test wie folgt entschieden:
p.value < alpha -> Die Nullhypothese ist widerlegt. Die Alternativhypothese wird angenommen.
p.value >= alpha -> Die Nullhypothese ist bestätigt.
Dabei ist alpha das vorher gewählte Signifikanzniveau, üblich ist 0.05 (bzw. 5%).

Kontinuitätskorrektur

Die Kontinuitätskorrektur wird angewendet, wenn nur ein Freiheitsgrad vorhanden ist oder die Stichprobenumfänge zwischen 20 und 50 liegen. Die sogenannte Yates' Korrektur verbessert dabei die Vergleichbarkeit der Testgröße mit der theoretischen Verteilung bei kleinen Stichproben mit nur einem Freiheitsgrad.

Ergebnisse des Chi-Quadrat-Tests

statistic	Chi-Quadrat- Wert= sum((Beobachtete Häufigkeiten - Erwartete Häufigkeiten)^2 /Erwartete Häufigkeiten)
parameter	Anzahl der Freiheitsgrade:= (Zeilenanzahl-1) * (Spaltenanzahl-1) der Kontingenztabelle
p.value	Der p-Wert ist ein Wahrscheinlichkeitsmaß für die Anzeichen gegen die Annahme der Nullhypothese. Geringere Wahrscheinlichkeiten liefern stärkere Anzeichen dafür, dass die Nullhypothese nicht zutrifft. Wenn der p-Wert <= alpha (Signifikanzniveau) ist, so wird die Nullhypothese zurückgewiesen. Wenn der p-Wert > alpha ist, so wird die Nullhypothese angenommen.
method	Der Typ des ausgeführten Chi-Quadrat-Test
observed	Zusammenstellung der beobachteten Zähldaten (Häufigkeiten)
expected	Die erwarteten Häufigkeiten unter der Nullhypothese Erwarteten Häufigkeit[i,j]= (Spaltensumme[i] * Zeilensumme[j])/ Gesamtanzahl.
residuals	Pearson Residuen Residuum[i,j]= (Beobachtete Häufigkeit[i,j] - Erwartete Häufigkeit[i,j]) / sqrt(Erwartete Häufigkeit[i,j])
stdres	Standard Residuen

Beispiele:

Mit dem **Anpassungstest** wird für ein Signifikanzniveau von alpha = 1% getestet, ob die Stichprobe für oder gegen eine Gleichverteilung der sechs Augenzahlen spricht. Der zu untersuchende Würfel wurde 500 mal geworfen und es ergab sich die folgende Häufigkeitsverteilung
für die Augenzahlen 1, 2, 3, 4, 5 und 6:
absolute Häufigkeit 75, 78, 96, 103, 77, 71
Nullhypothese für i=1,...,6 ist pi = P({Augenzahl=i}) = 1/6
Alternativhypothese für i=1,...,6 ist pi = P({Augenzahl=i}) <> 1/6

```
ni=[75,78,96,103,77,71]
result=R_ChisqTest(ni)
Der Chi-Quadrat-Test liefert das Ergebnis als Datengruppe 'result':
```

	statistic	=	10.048
	parameter	=	5
	p.value	=	0.0738866
	method	=	"Chi-squared test for given probabilities"
	observed	=	75,78,96,103,77,71
	expected	=	83.33,83.33,83.33,83.33,83.33,83.33
	residuals	=	-0.9129, -0.5842, 1.3876, 2.1544, -0.6938, -1.3510

	stdres	=	-1.0000, -0.6400, 1.5200, 2.3600, -0.7600, -1.4800

Wir erhalten einen Chi-Quadrat-Wert = 10.048 bei 5 Freiheitsgraden.
Der p-Wert des Tests ist 0.0738866. Er liegt deutlich über dem Signifikanzniveau von 0.01, so man davon ausgehen kann, dass eine Gleichverteilung vorliegt und der Würfel unverfälscht ist.

Kann auf Grund der vorliegenden Zahlen darauf geschlossen werden, dass die Absatzverteilung des Händlers signifikant von der im gesamten Markt abweicht? Die Nullhypothese setzt voraus, dass die Verteilung des Händlerabsatzes, der vorgegebene Verteilung in der Grundgesamtheit entspricht. In diesem **Anpassungstest** ist die Verteilung der Gesamtheit nicht gleichverteilt, so dass die Verteilung als Parameter 'p' übergeben werden muss.

```
x=[33400,35410,2610,12520,16340,9840,6070,7620,1190]; Absatzzahlen des Händlers nach Marken
p=[27,28,2,10,13,8,5,6,1]; Marktanteil in Prozent
result=R_ChisqTest( x,0,p)
Der Chi-Quadrat-Test liefert das Ergebnis als Datengruppe 'result':
```

	statistic	=	26.3469
	parameter	=	8
	p.value	=	0.000915901
	method	=	"Chi-squared test for given probabilities"
	observed	=	33400, 35410, 2610, 12520, 16340, 9840, 6070, 7620, 1190
	expected	=	33750, 35000, 2500, 12500, 16250, 10000, 6250, 7500, 1250
	residuals	=	-1.9052, 2.1915, 2.2000, 0.1789, 0.7060, -1.6000, -2.2768, 1.3856, -1.6971
	stdres	=	-2.2298, 2.5828, 2.2223, 0.1886, 0.7569, -1.6681, -2.3360, 1.4292, -1.7056

Wir erhalten einen Chi-Quadrat-Wert = 26.3469 bei 8 Freiheitsgraden.
Der p-Wert mit 0.0009 liegt unter dem Signifikanzniveau von 0,01. Der Händler kann deshalb davon ausgehen, dass sich seine Absatzstruktur von der des gesamten Marktes unterscheidet.

Mit dem **Unabhängigkeitstest** soll festgestellt werden, ob es einen Zusammenhang zwischen Augen- und Haarfarbe gibt. Die Nullhypothese lautet: Haar- und Augenfarbe sind unabhängig voneinander. Es wurden die Augen- und Haarfarbe von 592 Menschen beobachtet.

Haar\Augenf.	blau	braun	gruen	nuss
blond	94	7	16	10
braun	84	119	29	54
rot	17	26	14	14
schwarz	20	68	5	15

Aus den beobachteten Häufigkeiten wird ein segmentierter Datensatz zusammengesetzt. Die Segmente bilden die Spalten.

```
blau=[94,84,17,20]
braun=[7,119,26,68]
gruen=[16,29,14,5]
nuss=[10,54,14,15]
x=join(blau,braun)
x=join(x,gruen)
x=join(x,nuss)
SetSegLen(x,leng?(blau))
result=R_ChisqTest(x)
```

	statistic	=	138.29
	parameter	=	9
	p.value	=	2.32529e-25
	method	=	"Pearson's Chi-squared test"

Wir erhalten einen Chi-Quadrat-Wert = 138.29 bei 9 Freiheitsgraden.
Da der p-Wert < 0.05 ist, kann die Nullhypothese verworfen werden. Es besteht ein Zusammenhang zwischen Augenfarbe und Haarfarbe.

Mit dem **Unabhängigkeitstest** soll festgestellt werden, ob sich die Erwerbstätigkeit von Frauen und Männern unterscheidet. Die Nullhypothese lautet, die Erwerbstätigkeit ist unabhängig vom Geschlecht. Die Befragung umfasst 3468 Personen.

.	männlich	weiblich

hauptberufl. ganztags	1026	545
hauptberufl. halbtags	41	309
nebenher berufstätig	73	135
nicht erwerbstätig	619	720

Aus den beobachteten Häufigkeiten wird ein segmentierter Datensatz erzeugt und der Chi-Quadrat-Test ausgeführt.

```
man=[1026,41,73,619]
women=[545,309,135,720]
x=join(man,women)
SetSegLen(x,leng?(man))
result = R_ChisqTest(x)
```

	statistic	=	377.938
	parameter	=	3
	p.value	=	1.32914e-81
	method	=	"Pearson's Chi-squared test"

Sowohl für ein Signifikanzniveau von 5%, als auch von 1% wird die Nullhypothese abgelehnt. Daraus kann geschlossen werden, dass es zwischen Erwerbstätigkeit und Geschlecht ein Zusammenhang besteht.

R_Execute

Anwendungsbereich: R-Fernsteuerung

Verfügbar ab: Professional Edition (R-Kit)

Ein R-Script wird ausgeführt.

Deklaration:

```
R_Execute ( TxRScript [, TxRVarNames] [, Variable 1] [, Variable 2] [, Variable 3] [, Variable 4] [, Variable 5] [, Variable 6] [, Variable 7] [, Variable 8] ) -> Result
```

Parameter:

TxRScript	Das R-Skript wird übergeben und ausgeführt.
TxRVarNames	Angabe der R-Variablennamen für das R-Script. Die Variablennamen werden durch ein Semikolon getrennt. (optional)
Variable 1	1. Variable entsprechend der TxRVarNames-Definition (optional)
Variable 2	2. Variable entsprechend der TxRVarNames-Definition (optional)
Variable 3	3. Variable entsprechend der TxRVarNames-Definition (optional)
Variable 4	4. Variable entsprechend der TxRVarNames-Definition (optional)
Variable 5	5. Variable entsprechend der TxRVarNames-Definition (optional)
Variable 6	6. Variable entsprechend der TxRVarNames-Definition (optional)
Variable 7	7. Variable entsprechend der TxRVarNames-Definition (optional)
Variable 8	8. Variable entsprechend der TxRVarNames-Definition (optional)
Result	Ergebnis des R-Scripts

Beschreibung:

Die Funktion führt 3 Schritte aus:

1. Die übergebenen FAMOS-Variablen 'Variable 1' ... 'Variable 8' werden konvertiert und den angegebenen R-Variablen zugewiesen.

2. Das R- Skript wird an das R-System übergeben und ausgeführt.

3. Die Rückgabevariable oder das Ergebnis des letzten Ausdrucks des R- Skripts werden ermittelt, in eine FAMOS- Variable konvertiert und als 'Result' zurückgegeben.

Das R-Skript kann mehrere Ausdrücke enthalten. Die einzelnen Ausdrücke sind durch ein Semikolon zu trennen. Mit der Übergabe des R-Skripts werden alle Ausdrücke ausgeführt.

Soll ein R- Skript aus einer Datei geladen und ausgeführt werden, so muss das TxRScript mit dem Schlüsselwort **RScript:** beginnen. Danach folgt der Dateiname. Beispiel : R_Execute("**RScript:**"+"C:\Program Files\R\R-3.3.1\tests\MaxMinAvg.R","sd_value;input",input)

Der Parameter TxRVarNames enthält die R-Namen der Rückgabevariablen und/oder die R- Namen der Eingangsvariablen. Alle R-Namen sind durch ein Semikolon zu trennen.

TxRVarNames	Übergabe an das R- System	Rückgabe vom R-System
Der Parameter fehlt oder ist leer	Keine Übergabe von Eingangsvariablen	Das Ergebnis des letzten Ausdrucks des R- Skripts wird zurückgegeben.
Es ist keine Rückgabevariable angegeben. Beispiel: ";a;b"	Die Eingangsvariablen a und b werden an das R-System übergeben.	Das Ergebnis des letzten Ausdrucks des R- Skripts wird zurückgegeben.
Es sind keine Eingangs- und keine Rückgabevariable angegeben. Beispiel: ";"	Keine Übergabe von Eingangsvariablen	Das Ergebnis des letzten Ausdrucks des R- Skripts wird zurückgegeben.
Es ist nur die Rückgabevariable angegeben. Beispiel: "c;"	Keine Übergabe von Eingangsvariablen	Der Wert der Rückgabevariablen c wird zurückgegeben.
Es sind die Rückgabevariable und Eingangsvariablen angegeben. Beispiel: "c;a;b"	Die Eingangsvariablen a und b werden an das R-System übergeben.	Der Wert der Rückgabevariablen c wird zurückgegeben.

Die Anzahl der Namen der Eingangsvariablen muss mit der Anzahl der Parameter 'Variable 1'...'Variable 8' übereinstimmen. Sind die Eingangsnamen in TxRVarNames mehr als die Parameter 'Variable 1'...'Variable 8', so kommt es zu einem Sequenzabbruch. Sind die Eingangsnamen in TxRVarNames weniger als die Parameter 'Variable 1'...'Variable 8' so werden die überzähligen Parameter ignoriert.

Die Konvertierungsregeln für die Eingangsvariablen sind in der Funktion R_SetVar() beschrieben. Es ist egal, ob die Eingangsvariablen für ein R-Skript mit der Funktion R_Execute() oder vorher durch Aufrufe der Funktion R_SetVar() übergeben werden.

Die Rückgabevariable kann weiter spezifiziert werden. In R können Variablen eine komplexe Struktur haben. So erzeugt beispielsweise die R-

Funktion t.test(var1, var2) einen Data Frame mit den Komponenten 'statistic', 'parameter', 'p.value', 'conf.int', 'estimate', 'null.value', 'alternative', 'method' und 'data.name'. Ist für die weitere Auswertung aber nur der p.value von Interesse, so kann durch Angabe eines Komponentennamen, diese Komponente zurückgegeben werden.

Der Komponentenname wird durch einen Doppelpunkt vom Variablennamen getrennt. Beispiel: Das R-Skript result=t.test(group1, group2) wird ausgeführt und es soll nur der p.value zurückgegeben werden.

```
p_value =R_Execute("result=t.test(group1, group2)","result:p.value;")
```

Enthält das R-Skript keine Zuweisung an eine Ergebnisvariable, so kann auch eine Komponente des letzten Ausdrucks zurückgegeben werden.

```
p_value =R_Execute("t.test(group1, group2)",":p.value;")
```

In diesem Fall entfällt der Variablenname und es wird nur die gewünschte Komponente angegeben.

Für die Konvertierung der Rückgaben nach FAMOS gelten die gleichen Regeln wie in der Funktion R_GetVar().

Multithreading: Alle Funktionen des R-Kits dürfen in beliebigen Ausführungs-Threads (BEGIN_PARALLEL) aufgerufen werden, wirken aber Thread-übergreifend und global. Alle Aufrufe werden intern zunächst in den FAMOS-Haupt-Thread verschoben und von dort ausgeführt, da die R-Laufzeitumgebung keine parallelen Aufrufe aus unterschiedlichen Threads unterstützt.

Beispiele:

Einfache Berechung mittels R

```
ramp=Ramp(1,1,100)
rscript="mean_value=mean(input)"
result =R_Execute(rscript,"mean_value;input",ramp)
```

In FAMOS wird die Variable 'ramp' erzeugt. Die Werte dieser Variablen werden der R-Variablen 'input' zugewiesen. Das R-Skript berechnet den Mittelwert von dieser Variablen und weist das Ergebnis der R-Variablen mean_value zu. Die Rückgabevariable wird in die FAMOS-Variable 'result' konvertiert.

Gleichzeitige Berechnung mehrerer Werte mit einem Skript

```
ramp = Ramp(1,1,100)
rscript=        "min_value=min(input);"
rscript=rscript+"max_value=max(input);"
rscript=rscript+"mean_value=mean(input);"
rscript=rscript+"sd_value=sd(input);"
sd_Value1  =R_Execute(rscript,"sd_value;input",ramp); = 29.0115
min_Value1 =R_GetVar( "min_value"); =1
max_Value1 =R_GetVar( "max_value"); =100
mean_Value1=R_GetVar( "mean_value");= 50.5
```

In FAMOS wird die Variable 'ramp' erzeugt. Dann wird das R-Skript aus den einzelnen Anweisungen zur Berechnung von Minimum,Maximum,Mittelwert und Standardabweichung zusammengesetzt.Wichtig ist das Semikolon, das das Ende einer Anweisung markiert. Das Skript wird ausgeführt und die Rückgabevariable 'sd_value' zurückgegeben. Die anderen Werte werden danach mit der Funktion R_GetVar() aus dem R-System gelesen.

Berechnung mit Hilfe einer Skript-Datei

Die Datei MaxMinAvgSd.R hat folgenden Inhalt:

```
## Example
min_value <-min(input);
max_value <-max(input);
mean_value <-mean(input);
sd_value <-sd(input)
```

```
ramp = Ramp(1,1,100)
rFile="C:\Program Files\R\R-3.3.1\tests\MaxMinAvgSd.R"
rscript="RScript:"+rFile
sd_Value2  =R_Execute(rscript,"sd_value;input",ramp)
min_Value2 =R_GetVar("min_value")
max_Value2 =R_GetVar( "max_value");
mean_Value2=R_GetVar( "mean_value");
```

Die Funktion R_Execute weist der R-Variablen 'input' die FAMOS-Variable 'ramp' zu. Danach wird die R-Skriptdatei MaxMinAvgSd.R geladen und ausgeführt. Nach der Ausführung liefert die Funktion den Wert der R-Variablen 'sd_value'. Die anderen Werte werden danach mit der Funktion R_GetVar() aus dem R-System gelesen.

Übergabe und Ausführung einer anwenderdefinierten Funktion

```
x=Ramp(1,1,10)
y=Ramp(1,1,10)
R_Function="matrix_mult <- function(a,b){ c = a * b; return (c);}"
R_Execute(R_Function)
z=R_Execute("matrix_mult(x,y)",";x;y",x,y)
```

In R können anwenderdefinierte Funktionen geschrieben und ausgeführt werden. In der 3. Zeile wird das Skript für eine anwenderdefinierte

Funktionen gesetzt. Mit Hilfe dieser Funktion werden die Werte zweier Vektoren multipliziert. In der folgenden Zeile wird diese Funktion an das R-System übergeben. In der letzten Zeile werden die beiden FAMOS-Variablen 'x' und 'y' übergeben und die Funktion wird ausgeführt. Das Ergebnis des Ausdrucks wird als FAMOS-Variable 'z' zurückgegeben.

Einen Datensatz mit normal verteilten Zufallszahlen erzeugen.

Die Argumente der Funktion rnorm sind der Mittelwert der Verteilung, die Standardabweichung und die Anzahl der Abtastwerte.

```
Rscript="rnorm(mean=2,sd=3,n=100)"
x=R_Execute(Rscript)
```

Siehe auch:

R_GetVar, R_SetVar

R_GetOption

Anwendungsbereich: R-Fernsteuerung

Verfügbar ab: Professional Edition (R-Kit)

Einstellungen des R-Systems ermitteln

Deklaration:

```
R_GetOption ( TxParameterName ) -> TxParameterValue
```

Parameter:

TxParameterName	Name des vereinbarten optionalen Parameters
	"R_HOME" : R-Home Verzeichnis.
	"R_KITVERSION" : Version des imc R- Kits.
	"R_PROCESS" : 64- oder 32-Bit Version des R-Systems.
	"R_VERSION" : Version des R-Systems.
TxParameterValue	Parameterwert.

Beschreibung:

Mit dieser Funktionen können Einstellungen des R- Systems ermittelt werden. Bei den Namen der optionalen Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Beispiele:

Mit den folgenden Aufrufen werden die abfragbaren Parameter des R-Systems ermittelt.

```
R_Path=R_GetOption("R_HOME") ; -> C:\Program Files\R\R-3.3.2\bin\x64
R_Version=R_GetOption("R_version"); -> 3.3.2
R_KitVersion=R_GetOption("R_KITVERSION"); -> 7.2.1.0
R_Process=R_GetOption("R_PROCESS"); -> x64 oder i386
R_Unknown=R_GetOption("R_Unknown"); -> Abbruch: Der optionale Parameter 'R_Unknown' ist nicht vereinbart
```

Siehe auch:

R_GetVar

Anwendungsbereich: R-Fernsteuerung

Verfügbar ab: Professional Edition (R-Kit)

Den Wert einer R-Variablen ermitteln.

Deklaration:

```
R_GetVar ( TxRVarName ) -> Result
```

Parameter:

TxRVarName	Symbolischer Name der R-Variablen
Result	Datengruppe, normaler Datensatz, normaler segmentierter Datensatz, Datensatz mit Real- und Imaginärteil, segmentierter Datensatz mit Real- und Imaginärteil, Text oder Textfeld mit dem Wert der R-Variablen

Beschreibung:

Die R- Variable mit dem angegebenen Namen wird aus dem R-System ermittelt und ihr Inhalt in eine FAMOS- Variable konvertiert. Existiert die R-Variable nicht, so kommt es zum Abbruch der Sequenz.

Bei der Angabe des R- Variablennamen ist auf Groß- und Kleinschreibung zu achten.

Für die Konvertierung einer R-Variablen in eine FAMOS- Variable gelten folgende Regeln:

R- Variablentyp	Typ der FAMOS- Variable
Character Vektor mit 1 Element	Text
Character Vektor mit mehr als einem Element	Textfeld
Numerischer Vektor	Normaler Datensatz, reell 8 Byte
Numerische Matrix	Normaler, segmentierter Datensatz, reell 8 Byte
Komplexer Vektor	Komplexer Datensatz (Real- und Imaginärteil), reell 8 Byte
Komplexe Matrix	Komplexer, segmentierter Datensatz (Real- und Imaginärteil), reell 8 Byte
Integer Vektor	Normaler Datensatz, 4 Byte Integer
Integer Matrix	Normaler, segmentierter Datensatz, 4 Byte Integer
Logischer Vektor	Normaler Datensatz, 1 Byte Integer ohne VZ
Logische Matrix	Normaler, segmentierter Datensatz, 1 Byte Integer ohne VZ
Raw Vektor	Normaler Datensatz, 1 Byte Integer ohne VZ
Raw Matrix	Normaler, segmentierter Datensatz, 1 Byte Integer ohne VZ
List	Datengruppe
Data Frame	Datengruppe
Pair List	Datengruppe
Language Objekt	Datengruppe
S4 Objekt	Datengruppe
Symbol	Text
Factor	Normaler Datensatz, 4 Byte Integer

Hat die R-Variable eine zu komplexe Struktur, so kann sie nicht in eine FAMOS-Variable umgewandelt werden. Eine zu komplexe Struktur liegt beispielsweise vor, wenn die R- Variable eine Liste ist und in dieser Liste eine weitere Liste vorhanden ist. In diesem Fall würde die Sequenz mit einer entsprechenden Fehlermeldung abbrechen.

Um dennoch an die Ergebnisse zu kommen, können die Komponenten der R- Variablen einzeln gelesen werden. Dazu ist ein Komponentenname anzugeben. Die Syntax lautet **Variablenname:Komponentenname**. Der Komponentenname wird durch einen Doppelpunkt vom Variablennamen getrennt.

Der Komponentenname kann ein Name oder eine Zahl sein. Handelt es sich um eine Zahl, so wird dieser als Index gewertet. Über diesen Index wird die Komponente ermittelt.

Ist der Komponentenname ein Name, so wird versucht, diesen Namen in der Namensliste der Variablen zu finden. Dazu müssen die Komponenten der R-Variablen benannt sein.

Soll eine Komponente aus einer Matrix zurückgegeben werden, so hat der Komponentenname folgenden Aufbau: **Variablenname:Zeilenname,Spaltenname**. Zeilen- und Spaltenname dürfen wieder eine Zahl oder ein Name sein. Sie werden durch ein Komma getrennt.

mat	Die gesamte Matrix wird zurückgegeben.
mat:	Die gesamte Matrix wird zurückgegeben.
mat:,	Die gesamte Matrix wird zurückgegeben.
mat:row,	Die Zeile row der Matrix wird zurückgegeben.
mat:,col	Die Spalte col der Matrix wird zurückgegeben.
mat:row,col	Der Wert von Zeile row und Spalte col der Matrix wird zurückgegeben.

Enthält die R-Variable Werte, die außerhalb des gültigen Zahlenbereichs von FAMOS liegen, so werden diese wie folgt ersetzt. Für die R- Variablentypen numerischer Vektor, numerische Matrix, komplexer Vektor und komplexe Matrix gilt:

Wert	Rückgabe
Division durch Null	0
Wert > 1e35	1e35
Wert < -1e35	-1e35
NaN	1e35

Für die R- Variablentypen Integer Vektor und Integer Matrix gilt:

Wert	Rückgabe
Division durch Null	-2147483648
NA (Fehlender Wert)	-2147483648
NaN (Keine Zahl)	-2147483648

Für die R- Variablentypen logischer Vektor, logische Matrix, Raw Vektor und Raw Matrix:

Wert	Rückgabe
Division durch Null	0
NA (Fehlender Wert)	0
NaN (Keine Zahl)	0
Werte außerhalb von 0...255	0

Beispiele:

Mit dem R- Skript werden 2 numerische Vektoren erzeugt und die Funktion t.test() ausgeführt. Das Ergebnis wird der Variablen 'result' zugewiesen.

Die Funktion t.test() liefert eine Liste mit den Komponenten 'statistic', 'parameter', 'p.value', 'conf.int', 'estimate', 'null.value', 'alternative', 'method' und 'data.name'

```
RScript=        "group1 <- c(30.02, 29.99, 30.11, 29.97, 30.01, 29.99);"
RScript=RScript+"group2 <- c(29.89, 29.93, 29.72, 29.98, 30.02, 29.98);"
RScript=RScript+"result= t.test(group1, group2)"
result =R_Execute(RScript)

Die Funktion liefert das Ergebnis letzten Ausdrucks als Datengruppe 'result':
```

```
                statistic   (=1.95901)
                parameter   (=7.03056)
                p.value   (=0.0907733)
                conf.int
                estimate
                null.value   (=0)
                alternative   (="two.sided")
                method   (="Welch Two Sample t-test")
                data.name   (="group1 and group2")
Es gibt 2 Möglichkeiten, um nur den Wert von p.value zu ermitteln:
p_value1=R_GetVar("result:p.value"); Rückgabe der benannten Komponente p.value der Variablen result
p_value2=R_GetVar("result:3"); Rückgabe der 3. Komponente der Variablen result
```

Mittels R-Script wird in R ein Data Frame erzeugt. Die erste Komponente ist ein numerischer Vektor, die zweite ein Character- Vektor und die dritte ein logischer Vektor.

```
RScript="d <- c(1,2,3,4);"
RScript=RScript+"e <- c('red','white','blue',NA);"
RScript=RScript+"f <- c(TRUE,TRUE,TRUE,FALSE);"
RScript=RScript+"mydata <- data.frame(d,e,f,stringsAsFactors=FALSE);"
RScript=RScript+"names(mydata) <- c('ID' ,'Color','Passed')";  Die Komponenten des Data Frame werden benannt.
R_Execute(RScript);  Die Variable 'mydata' wird erzeugt.
myDataGroup=R_GetVar("mydata");  Die Variable 'mydata' wird gelesen und in eine Datengruppe konvertiert.
Die Komponente 'ID' wird in einen normalen Datensatz, die Komponente 'Color' in ein Textfeld und die Komponente 'Passed' in einen normalen Datensatz im Format 1 Byte Integer ohne

Es gibt 2 Möglichkeiten, um Komponenten der R-Variable zu lesen:
Die 1. Komponente der Variablen 'mydata' wird gelesen.
ID   = R_GetVar("mydata:1")
Die benannte Komponente 'Color' der Variablen 'mydata' wird gelesen.
Color= R_GetVar("mydata:Color")
```

Siehe auch:

R_SetVar

R_Norm

Anwendungsbereich: R-Fernsteuerung

Verfügbar ab: Professional Edition (R-Kit)

Funktionen zur Berechnung von Dichten, Wahrscheinlichkeitsverteilungen und Quantilen der Standardnormalverteilung

Deklaration:

```
R_Norm ( TxFnt, x [, mean] [, sd] ) -> Result
```

Parameter:

TxFnt	Name der Funktion bestimmt, was berechnet werden soll.
	"d" : Wahrscheinlichkeitsdichte an der Stelle x.
	"p" : Wert der Verteilungsfunktion an der Stelle x.
	"q" : Quantil zu der vorgegebenen Wahrscheinlichkeit x.
x	Einzelwert oder normaler Datensatz x
mean	Erwartungs- bzw. Mittelwert (Standard=0) (optional , Standardwert: 0)
sd	Standardabweichung (Standard=1.0) (optional , Standardwert: 1)
Result	Ergebnis der Berechnung.

Beschreibung:

Die Berechnungen werden mittels R durchgeführt. D.h. aus den Parametern wird ein R-Skript erzeugt und ausgeführt. Anschließend wird das Ergebnis des Ausdrucks gelesen und in eine FAMOS-Variable konvertiert.

x kann ein Einzelwert oder ein normaler Datensatz sein.

Die Anzahl der Ergebniswerte wird durch die Anzahl der Werte von x bestimmt.

Die Angabe des Mittelwertes ist optional. Wird er nicht angegeben, so wird mit dem Standardwert=0 gerechnet.

Die Angabe der Standardabweichung ist optional. Wird sie nicht angegeben, so wird mit dem Standardwert=1 gerechnet.

Eine negative Standardabweichung erzeugt in der Berechnung einen Fehler. Das Ergebnis ist in diesem Fall 1e35.

Beispiele:

Dichte: Berechnung der Wahrscheinlichkeitsdichte an der Stelle x=3

```
y=R_Norm("d",3); y= 0.00443185
```

Verteilungsfunktion: Der Wert der Verteilungsfunktionan der Stelle x gibt die Wahrscheinlichkeit an, dass ein Wert <= x realisiert wird.

```
y=R_Norm("p",1.644854); y= 0.95
```

Die Wahrscheinlichkeit für eine Realisierung <= 1.644854 beträgt 0.95

Quantil: Berechnung des Wertes, der mit einer Wahrscheinlichkeit von 0.95 nicht überschritten wird:

```
y=R_Norm("q",0.95,0,1); y= 1.64485
```

R_SetVar

Anwendungsbereich: R-Fernsteuerung

Verfügbar ab: Professional Edition (R-Kit)

Einer R-Variable einen Wert zuweisen.

Deklaration:

```
R_SetVar ( TxRVarName, FAMOSVariable )
```

Parameter:

TxRVarName	Symbolischer Name der R-Variablen
FAMOSVariable	Der Inhalt der FAMOS- Variable wird an die R-Variable übergeben.

Beschreibung:

Mit dieser Funktion können R- Variablen definiert und ihnen Werte zugewiesen werden.

Dieser Wert kann ein Einzelwert,ein FAMOS-Datensatz, ein Text, ein Textfeld oder eine Datengruppe sein.

Eine FAMOS-Variable wird nach den folgenden Regeln konvertiert:

Typ der FAMOS- Variable	R- Variablentyp
Normaler Datensatz	Numerischer Vektor
Normaler segmentierter Datensatz	Numerische Matrix
Datensatz mit Real- und Imaginärteil	Komplexer Vektor
Segmentierter Datensatz mit Real- und Imaginärteil	Komplexe Matrix
Datensatz mit Betrag und Phase	Komplexer Vektor
Segmentierter Datensatz mit Betrag und Phase	Komplexe Matrix
Text	Charakter Vektor
Textfeld	Charakter Vektor
Datengruppe	R- Liste

Datensätze mit Events, Datensätze mit Betrag und Phase in db und Datensätze im Format Time Stamp ASCII werden nicht unterstützt.

Bei der Konvertierung eines segmentierten Datensatzes wird jedes Segment zu einer Spalte in der R- Matrix.

Datensätze mit Betrag und Phase werden in Datensätze mit Real- und Imaginärteil umgerechnet und dann in einen komplexen Vektor bzw. Matrix konvertiert.

Eine Datengruppe wird in eine R- Liste umgewandelt. Die Komponenten der R-Liste ergeben sich aus den Elementen der Datengruppe.

Multithreading: Alle Funktionen des R-Kits dürfen in beliebigen Ausführungs-Threads (BEGIN_PARALLEL) aufgerufen werden, wirken aber Thread-übergreifend und global. Alle Aufrufe werden intern zunächst in den FAMOS-Haupt-Thread verschoben und von dort ausgeführt, da die R-Laufzeitumgebung keine parallelen Aufrufe aus unterschiedlichen Threads unterstützt.

Beispiele:

Die FAMOS-Variable 'fa' wird der R-Variablen 'a' zugewiesen.

```
fa=1000
R_SetVar("a",fa); Einzelwert -> NumericVector
```

Die Text-Variable 't' wird der R-Variablen 'ch' zugewiesen.

```
t="123.4"
R_SetVar("ch",t); Text -> CharacterVector
```

Der segmentierte Datensatz seg wird der R-Variablen mat zugewiesen.

```
seg=Ramp(1,1,100)
SetSegLen(seg,20)
R_SetVar("mat",seg); Normaler segmentierter Datensatz -> NumericMatrix
```

Es wird eine Datengruppe 'roller' erzeugt. Diese Gruppe wird der R-Variablen 'roller' zugewiesen. Die R-Variable 'roller' ist vom Typ R-Liste.

```
weight=Ramp(0,1,3) ; Normaler Datensatz
weight[1]=1.9
weight[2]=3.1
weight[3]=3.3
depression=Ramp(0,1,4) ; 1 Wert mehr als weight
```

```
depression[1]=2
depression[2]=1
depression[3]=5
depression[4]=5
doc= TxArrayCreate(2) ; Text-Array mit 2 Elementen
doc[1]="abc"
doc[2]="def"
dsseg=Ramp(1,1,100); Normaler segmentierter Datensatz
SetSegLen( dsseg,20)
roller=GrNew(); Datengruppe 'roller' erzeugen
GrChannAppend(roller,weight)
GrChannAppend(roller,depression)
GrChannAppend(roller,doc)
GrChannAppend(roller,dsseg)
R_SetVar("roller",roller) ; FAMOS-Gruppe --> R-Liste
```

Siehe auch:

R_GetVar

R_tTest

Anwendungsbereich: R-Fernsteuerung

Verfügbar ab: Professional Edition (R-Kit)

Funktion zur Ausführung des t-Tests

Deklaration:

```
R_tTest ( x [, y] [, alternative] [, mu0] [, paired] [, var_equal] [, conf_level] ) -> Result
```

Parameter:

x	Daten der ersten Stichprobe
y	Daten der zweiten Stichprobe (im Zweistichprobentest) (optional , Standardwert: 0)
alternative	Formulierung der Alternativhypothese (optional , Standardwert: "t")
	"t" : H1: mu <> mu0 (Standard)
	"g" : H1: mu > mu0
	"l" : H1: mu < mu0
mu0	Hypothetisierter Populationsmittelwert (Standard=0) (optional , Standardwert: 0)
paired	Gepaarte Stichproben (optional , Standardwert: 0)
	0 : Unabhängige Stichproben.(Standard)
	1 : Gepaarte Stichproben.
var_equal	Varianzen der Stichproben sind gleich (optional , Standardwert: 0)
	0 : Varianzheterogenität.(Standard)
	1 : Varianzhomogenität.(Welch-Test)
conf_level	Konfidenzniveau (Standard =0.95) Das Konfidenzniveau gibt an, mit welcher die Wahrscheinlichkeit der Parameter der Grundgesamtheit (z. B. der Mittelwert) im Konfidenzintervall enthalten ist. (optional , Standardwert: 0.95)
Result	Ergebnis des t-Tests ist eine Datengruppe mit den Elementen:
	statistic : Wert der t- Statistik
	parameter : Anzahl der Freiheitsgrade. Die Freiheitsgrade geben die Anzahl von Größen eines Systems an, die bei einem feststehenden arithmetischen Mittel unabhängig voneinander variiert werden können.
	p.value : Der p-Wert ist ein Wahrscheinlichkeitsmaß für die Anzeichen gegen die Annahme der Nullhypothese. Er liegt im Bereich von 0 bis 1. Geringere Wahrscheinlichkeiten liefern stärkere Anzeichen dafür, dass die Nullhypothese nicht zutrifft. Der p-Wert kann mit dem alpha-Wert verglichen werden, um zu entscheiden, ob die Nullhypothese (H0) zurückgewiesen werden soll. Wenn der p-Wert <= alpha (Signifikanzniveau) ist, wird die Nullhypothese zurückgewiesen. Wenn der p-Wert > alpha ist, wird die Nullhypothese nicht zurückgewiesen.
	conf.int : Das Konfidenzintervall gibt den Wertebereich an, in dem die Parameter der Grundgesamtheit mit einer bestimmten Wahrscheinlichkeit liegen. Wie hoch diese Wahrscheinlichkeit sein soll, legt man mit dem Konfidenzniveau fest.
	estimate : Der geschätzte Mittelwert oder die Mittelwerte in Abhängigkeit davon, ob es ein Einstich- oder ein Zweistichprobentest war.
	null.value : Die angegebene Hypothese auf den Mittelwert oder die Differenz der Mittelwerte, je nachdem, ob es ein Einstich- oder ein Zweistichprobentest war.
	alternative : Die Zeichenfolge beschreibt die alternative Hypothese.
	method : Die Zeichenfolge zeigt an, welcher Typ des t-Tests ausgeführt wurde.

Beschreibung:

Der t-Test bezeichnet eine Gruppe von Hypothesentests mit t-verteilter Testprüfgröße. Er kann verwendet werden, um zu bestimmen, ob zwei Stichproben sich statistisch signifikant unterscheiden. Es werden immer nur 2 Stichproben verglichen, die normalverteilt sein müssen.
Bei einem Hypothesentest werden zwei gegensätzliche Hypothesen zu einer Grundgesamtheit untersucht: die Nullhypothese und die Alternativhypothese.
Die Nullhypothese H0 besagt, dass ein Parameter der Grundgesamtheit gleich einem bestimmten Wert ist.
Die Alternativhypothese H1 besagt, dass der Parameter der Grundgesamtheit vom Wert des Parameters der Grundgesamtheit in der Nullhypothese abweicht.
Beim t-Test wird unterschieden zwischen:

Einstichprobentest

Der Einstichproben- Test prüft anhand des Mittelwertes einer Stichprobe, ob der Mittelwert sich von einem vorgegebenen Sollwert unterscheidet. Dabei wird vorausgesetzt, dass die Daten der Stichprobe normalverteilt sind. Bei einer großen Stichprobe kann der t-Test auch ohne die Normalverteilungsannahme verwendet werden.
Für den Einstichproben-Test muss die zweite Stichprobe 'y' leer oder darf nicht angegeben sein.
Der Sollwert wird als 'mu0' angegeben. Der Aufruf lautet:
result=R_tTest(x,empty,"t",mu0)

Die Hypothesen für den Einstichprobentest lauten:

Nullhypothese	Bedeutung
H0: mu = mu0	Der Mittelwert der Grundgesamtheit (mu) ist gleich dem hypothetischen Mittelwert (mu0).
Alternativhypothese	.
H1: mu != mu0	"t" Der Mittelwert der Grundgesamtheit (mu) weicht vom hypothetischen Mittelwert (mu0) ab.
H1: mu > mu0	"g" Der Mittelwert der Grundgesamtheit (mu) ist größer als der hypothetischen Mittelwert (mu0).
H1: mu < mu0	"l" Der Mittelwert der Grundgesamtheit (mu) ist kleiner als der hypothetischen Mittelwert (mu0).

Zweistichprobentest

Der Zweistichprobentest prüft anhand der Mittelwerte zweier unabhängiger Stichproben, wie sich die Mittelwerte zweier Grundgesamtheiten zueinander verhalten. Die Mittelwertsdifferenz wird gegen mu0 bzw. gegen 0 getestet. Dabei wird vorausgesetzt, dass die Daten der Stichproben normalverteilt sind bzw. es genügend große Stichprobenumfänge gibt. Die Daten beider Stichproben müssen zunächst (per F-Test) auf Varianzhomogenität untersucht werden. Liegt Varianzhomogenität vor, so lautet der Aufruf:
result=R_tTest(x,y,"t",0.0,0,1)
,wobei 'x' die Daten der ersten und 'y' die Daten der zweiten Stichprobe enthält. Der Parameter 'var_equal'= 1 bedeutet, dass Varianzhomogenität angenommen wird.
Liegt Varianzheterogenität vor, so lautet der Aufruf:
result=R_tTest(x,y)
Dieser Aufruf entspricht dem Welch-Test.

Die Hypothesen für den Zweistichprobentest lauten:

Nullhypothese	Bedeutung
H0: mux-muy = delta0	Die Differenz zwischen den Mittelwerten der Grundgesamtheiten (mux-muy) ist gleich der hypothetischen Differenz (delta0). Der Parameter mu0 entspricht in diesem Fall delta0.
Alternativhypothese	.
H1: mux-muy != delta0	"t" Die Differenz zwischen den Mittelwerten der Grundgesamtheiten (mux-muy) ist ungleich der hypothetischen Differenz (delta0). Der Parameter mu0 entspricht in diesem Fall delta0.
H1: mux-muy > delta0	"g" Die Differenz zwischen den Mittelwerten der Grundgesamtheiten (mux-muy) ist größer als die hypothetische Differenz (delta0). Der Parameter mu0 entspricht in diesem Fall delta0.
H1: mux-muy < delta0	"l" Die Differenz zwischen den Mittelwerten der Grundgesamtheiten (mux-muy) ist kleiner als die hypothetische Differenz (delta0). Der Parameter mu0 entspricht in diesem Fall delta0.

Gepaarter Zweistichprobentest

Der gepaarter Zweistichprobentest prüft für zwei verbundene (abhängige) Stichproben, ob sich die mittlere Differenz der Messwerte unterscheidet. Er wird durchgeführt, wenn in der selben Untersuchungsgruppe zwei Erhebungen stattgefunden haben, und diese Daten nun untersucht werden sollen. Die Anzahl der Werte in beiden Stichproben muss identisch sein. Die Differenzen der gepaarten Messwerte müssen normalverteilt sein. Es reicht nicht aus zu zeigen, dass die beiden Stichproben einer Normalverteilung folgen.
Der Aufruf lautet:
result=R_tTest(x,y,"t",0,1)
wobei 'x' die Stichprobe der 1. Erhebung und 'y' die Stichprobe der 2. Erhebung sind.

Die Hypothesen für den gepaarten Zweistichprobentest lauten:

Nullhypothese	Bedeutung
H0: mud=mu0	Der Mittelwert der Differenzen der Grundgesamtheit (mud) ist gleich dem hypothetischen Mittelwert der Differenzen (mu0).
Alternativhypothese	.
H1: mud != mu0	"t" Der Mittelwert der Differenzen der Grundgesamtheit (mud) ist ungleich dem hypothetischen Mittelwert der Differenzen (mu0).
H1: mud > mu0	"g" Der Mittelwert der Differenzen der Grundgesamtheit (mud) ist größer als der hypothetische Mittelwert der Differenzen (mu0).
H1: mud < mu0	"l" Der Mittelwert der Differenzen der Grundgesamtheit (mud) ist kleiner als der hypothetische Mittelwert der Differenzen (mu0).

Beispiele:

Einer Lieferung Dioden mit gewünschtem Durchlasswiderstand von 100 mOhm wird eine zufällige Stichprobe der Größe 10 entnommen und

vermessen.

```
resistance =[114.62,110.10,106.31,99.30,107.28,108.35,113.64,117.92,130.15,102.74]
result=R_tTest(resistance,empty,"t",100.00)
Der Einstichprobentest liefert das Ergebnis als Datengruppe 'result':
```

	statistic	=	4.0066
	parameter	=	9
	p.value	=	0.00307962
	conf.int	=	104.8072, 117.2748
	estimate	=	111.041
	null.value	=	100
	alternative	=	"two.sided"
	method	=	"One Sample t-test"

Wir erhalten den Mittelwert = 111.04 und den t-Wert = 4.0066 bei 9 Freiheitsgraden.
Der p-Wert des Tests ist 0.00308. Er liegt deutlich unter dem Signifikanzniveau von 0.05, so dass die Nullhypothese verworfen werden muss.
Der mittlere Durchlasswiderstand entspricht mit großer Wahrscheinlichkeit nicht dem gewünschten Durchlasswiderstand von 100 mOhm.

Von zwei Lieferungen von Dioden soll ermittelt werden, ob der Durchlaßwiderstand der beiden Lieferungen identisch ist (Die Differenz der Erwartungswerte ist = 0). Jeder Lieferung wird eine Stichprobe von 10 Dioden entnommen und vermessen. Es wird angenommen, dass gleiche Varianzen vorliegen.

```
resistanceA =[114.62,110.10,106.31,99.30,107.28,108.35,113.64,117.92,130.15,102.74]
resistanceB =[101.77,109.86,131.41,105.29,104.49,118.62,108.60,139.09,113.72,114.91]
mu0=0
var_equal=1
result=R_tTest(resistanceA,resistanceB,"t",mu0,0,var_equal)
Der Zweistichprobentest liefert das Ergebnis als Datengruppe 'result':
```

	statistic	=	-0.79341
	parameter	=	18
	p.value	=	0.437873
	conf.int	=	-13.6251, 6.1551
	estimate	=	111.0410, 114.7760
	null.value	=	0
	alternative	=	"two.sided"
	method	=	"Two Sample t-test"

Wir erhalten den Mittelwert von resistanceA = 111.04 und von resistanceB = 114.77.
Es ergibt sich ein t-Wert = -0.7934 bei 18 Freiheitsgraden.
Die Nullhypothese wird angenommen: der p-Wert des Tests ist 0.4379.
Die beiden Lieferungen haben mit großer Wahrscheinlichkeit denselben mittleren Widerstand.

Die Reißlasten von Drähten werden mit einer Maschine A und einer Maschine B untersucht. Um die Gleichwertigkeit der beiden Maschinen zu prüfen, werden 12 Drahtproben geteilt und jede Hälfte an einer Maschine getestet. Es ergeben sich die Reißlast-Messungen von:

```
machineA=[ 35, 46, 34, 27, 37, 59, 52, 61, 21, 31, 37, 27]
machineB=[ 39, 51, 32, 23, 41, 53, 51, 55, 19, 36, 37, 26]
paired=1
result=R_tTest( machineA,machineB,"t",0,paired)
Der gepaarte Zweistichprobentest liefert das Ergebnis als Datengruppe 'result':
```

	statistic	=	0.286513
	parameter	=	11
	p.value	=	0.77981
	conf.int	=	-2.2273, 2.8940
	estimate	=	0.333333
	null.value	=	0
	alternative	=	"two.sided"
	method	=	"Paired t-test"

Man erhält einen t-Wert = 0.2865 bei 11 Freiheitsgraden. Der Mittelwert der Differenzen ist 0.333. Die Nullhypothese kann akzeptiert werden, da der p-Wert = 0.7798 ist. Die Maschinen haben gleichwertige mittlere Reißlast-Messungen.

Ramp

Erzeugt eine Rampe (Gerade mit Steigung 1) mit vorgebbarem Startwert, Punkteabstand und Länge
Alternativer Name: **Rampe**

Deklaration:

```
Ramp ( EwStart, EwDelta, EwLänge ) -> Rampe
```

Parameter:

EwStart	Startwert
EwDelta	Abtastzeit (Delta-X)
EwLänge	Länge
Rampe	Resultierender Datensatz in Form einer Rampe

Beschreibung:

Es wird eine Gerade mit der Steigung 1 erzeugt, die also der Gleichung

```
f(x) = x
```

gehorcht. Die erzeugte Gerade hat die Form einer Rampe.

- Der erzeugte Datensatz hat keine Einheiten. Die Länge (3. Parameter) muss eine ganze Zahl größer gleich 0 sein. Die Abtastzeit (2. Parameter) muss größer Null sein.
- Die Funktion Ramp() wird im Allgemeinen benutzt, um Testdatensätze mit Vergleichsfunktionen zu erzeugen.

Beispiele:

Es wird ein Datensatz mit den Werten 2.0, 2.1, 2.2, 2.3, 2.4 erzeugt:

```
Ramp5 = Ramp(2, 0.1, 5)
```

Es wird ein Datensatz mit 10 Werten erzeugt, die alle gleich Null sind. Solche Datensätze mit bekannter Abtastzeit und Anzahl von Werten werden oft als Grundlage für weitere Operationen benötigt, die einen Datensatz voraussetzen:

```
TenZeroes = Ramp(0, 1, 10) * 0
```

Es wird ein sinusförmiger Datensatz erzeugt. Er enthält 2 Perioden einer sin-Funktion mit der Amplitude 3A und der Phasenverschiebung PI/4. Die vordefinierte Konstante PI wird benutzt:

```
Sinus = 3 'A' * sin(Ramp(PI/4, PI/128, 512))
```

Die aufgenommenen Daten vor dem Trigger werden zusammengefügt mit den Daten nach dem Trigger:

```
NDcomplete = Join(NDpreTrigger, NDpostTrigger)
```

Siehe auch:

Random, Leng, XOff, XDel

Random

Generierung von Zufallszahlen mit vorgebbarer Verteilung

Deklaration:

```
Random ( EwAnzahl, EwVerteilung, EwPar1, EwPar2, EwInit ) -> Zufallszahlen
```

Parameter:

EwAnzahl	Länge des erzeugten Datensatzes
EwVerteilung	Angabe der gewünschten Verteilung
	0 : Gleichverteilung
	1 : Exponentialverteilung
	2 : Normalverteilung
	3 : Binomialverteilung
EwPar1	Bei Gleichverteilung das Minimum. Bei Binominalverteilung der Maximalwert, die erzeugten Zahlen liegen im Bereich (0,1,2...EwPar1). Sonst auf 0 zu setzen.
EwPar2	Bei Gleichverteilung das Maximum. Bei Binominalverteilung der gewünschten Verteilungswert p (0< p <1). Sonst auf 0 zu setzen.
EwInit	Startwert für die Initialisierung des Pseudo-Zufallszahlengenerators. 0 bedeutet keine Neuinitialisierung, sonst >0 und ganzzahlig.
Zufallszahlen	Datensatz mit Zufallszahlen

Beschreibung:

Beispiele:

```
r1 = Random(10000, 0, -1, 1, 0)
```

Erzeugt 10000 gleichverteilte Zufallszahlen zwischen -1 und +1.

```
r21 = Random(10000, 2, 0, 0, 11)
```

Erzeugt 10000 normalverteilte Zufallszahlen.

```
r22 = Random(10000, 2, 0, 0, 11)
```

Identisch zu r21 (da gleicher Wert für Initialisierung),

```
r4 = Random(10000, 3, 1, 0.7, 24)
```

Erzeugt Datensatz der Länge 10000, der aus den Werten 0 (ca. 30%) und 1 (ca.70%) besteht.

Siehe auch:

Ramp

RangeSet

Verfügbar ab: Professional Edition

Werte der Eingangsdaten, die innerhalb eines bestimmten Wertebereichs eines Steuerkanals liegen, werden auf einen einen anderen Wert gesetzt.

Deklaration:

```
RangeSet ( Eingangsdaten, Steuerkanal, Bedingung Unten, Untergrenze, Bedingung Oben, Obergrenze, Aktion [, Ersatzwert] ) -> Ergebnis
```

Parameter:

Eingangsdaten	Eingangsdaten
Steuerkanal	Steuerkanal
Bedingung Unten	Bedingung für Untergrenze
	">=" : Steuerkanal >= Untergrenze
	">" : Steuerkanal >= Untergrenze
Untergrenze	Untergrenze
Bedingung Oben	Bedingung für Obergrenze
	"<=" : Steuerkanal <= Obergrenze
	"<" : Steuerkanal < Obergrenze
Obergrenze	Obergrenze
Aktion	Aktion
	"=" : Auf Ersatzwert setzen
	"+" : Ersatzwert addieren
	"first" : Beim Eintritt in den Bereich den ersten Wert der Eingangsdaten als Ersatzwert benutzen
	"before" : Beim Eintritt in den Bereich den vorherigen Wert der Eingangsdaten als Ersatzwert benutzen. 0.0, falls gleich zu Anfang im Bereich.
Ersatzwert	Ersatzwert (optional , Standardwert: 0)
Ergebnis	Ergebnis

Beschreibung:

Beide Bedingungen werden UND-verknüpft.

Die reellen Eingangsdaten können Events und Segmente haben. Äquidistante und XY-Daten werden unterstützt, bei XY-Daten wird die Y-Komponente verrechnet.

Der Steuerkanal ist äquidistant. Die Zuordnung eines Wertes der Eingangsdaten und eines Wertes des Steuerkanals erfolgt Wert für Wert nach Reihenfolge.

Die Eingangsdaten und der Steuerkanal müssen dieselbe Länge und Struktur (Segmente und Events) aufweisen.

Für das Ersetzen von Lost Value, Overflow, Not a Number, Übersteuerung, Fühlerbruch sei auf LostValueReplace() verwiesen.

Beispiele:

Bei einer Drehzahlbestimmung aus Pulsen entstehen in den Zeiten des Stillstands aufgrund des langen Pulsabstandes kleine Drehzahl über null.

Alle Werte von 0 bis 10 auf 0.0 setzen.

```
rpm = RangeSet ( rpm, rpm, ">=", 0, "<=", 10, "=", 0 )
```

Bei einer Gangerkennung erscheint rückwärts als 7, soll aber -1 werden

```
gear = RangeSet ( gear, gear, ">=", 7, "<=", 7, "=", -1 )
```

Wenn die Temperatur zu klein ist, ist die gemessene Kraft ungültig und wird auf 0 gesetzt.

```
force = RangeSet ( force, temperature, ">=", -1e35, "<=", -40, "=", 0 )
```

Siehe auch:

CodeRange, LowerValue, UpperValue, Oben, Setze, LostValueReplace

Recip

Reziprokwert (Kehrwert)
Alternativer Name: **Rez**

Deklaration:

```
Recip ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Parameter. Erlaubte Typen: [ND],[XY].
Ergebnis	Reziprokwert des Parameters

Beschreibung:

Es wird der Reziprokwert, d. h. der Kehrwert, gebildet.

Anmerkungen

- Die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Es wird auch der Kehrwert der Einheit gebildet, z. B. aus "V" wird "1/V".
- Eine Division durch Null ist nicht erlaubt, d. h. der Parameter darf nicht Null werden. Wenn doch, wird eine entsprechende Warnung erzeugt.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Aus einem elektrischen Leitwert wird durch Kehrwertbildung ein elektrischer Widerstand ermittelt:

```
resistance = Recip(conductance)
```

Diese Zeile ist identisch zu folgender:

```
resistance = 1 / conductance
```

Siehe auch:

/(Division)

Rect

Transformation eines komplexen Datensatzes in kartesische Koordinaten (Real/Imaginärteil).
Alternativer Name: **Kart**

Deklaration:

```
Rect ( KomplexeDaten ) -> KomplexAlsRI
```

Parameter:

KomplexeDaten	Zu transformierender komplexer Datensatz. [BP], [DP] oder [RI].
KomplexAlsRI	Resultierender komplexer Datensatz in kartesischen Koordinaten. [RI]

Beschreibung:

Ein komplexer Datensatz wird in kartesische Koordinaten mit Real- und Imaginärteil gewandelt. Dabei ist es egal, in welchem Typ er vorliegt. Liegt er bereits in kartesischen Koordinaten vor, hat die Funktion keine Wirkung.

- Die y-Einheiten werden entsprechend der durchgeführten Transformation angepasst.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Ein Spektrum wird in die meist anschaulichere Darstellungsart BP transformiert:

```
MPspectrum = Pol(RIspectrum)
```

Ein Spektrum wird berechnet und in kartesischen Koordinaten ausgedrückt:

```
RIdata = Rect(Spec(NDdata))
```

Siehe auch:

Pol, idB, Compl

Red

Nachträgliches Abtasten mit vorgebbarem Reduktionsfaktor.
Alternativer Name: **Tast**

Deklaration:

```
Red ( Daten, EwFaktor ) -> Ergebnis
```

Parameter:

Daten	Abzutastender Datensatz. Erlaubte Datentypen: [ND]
EwFaktor	Reduktionsfaktor
Ergebnis	Ergebnis der Nachabtastung.

Beschreibung:

Ein Datensatz wird nachträglich abgetastet. Dabei wird die Datenmenge reduziert. Sie können den Faktor angeben, um den die Datenmenge reduziert wird. Beim Abtasten wird jeder n-te Wert des übergebenen Datensatzes in den zu erzeugenden Datensatz eingetragen.

- Die Einheit des Datensatzes wird beibehalten, die Abtastzeit um den übergebenen Faktor vergrößert.
- Der Reduktionsfaktor sollte keine Einheit haben und eine ganze Zahl größer 1 sein.
- Die Funktion Red() kehrt die Funktionen IPol() und Lip() um.
- Beachten Sie, dass beim Abtasten eines Datensatzes Aliasing-Effekte auftreten, wenn hohe Frequenzanteile im Datensatz vorhanden sind. Sie sollten dann vor einem Abtasten eine Glättungsfunktion zur Tiefpassfilterung einsetzen, z. B. Smo().

Beispiele:

```
NDshort = Red(NDlong, 5)
```

Ein Datensatz von 10000 Punkten Länge wird um den Faktor 5 auf 2000 Punkte reduziert. Dabei wurde jeder 5. Wert genommen.

```
NDShort = Red(Smo5(NDlong), 2)
```

Zur Unterdrückung von Aliasing-Effekten wird der Datensatz vor dem Abtasten mit einem angemessenen Tiefpass gefiltert. Bei größeren Reduktionsfaktoren sollten Sie i.Allg. die Funktion Smo() benutzen.

Siehe auch:

Red2, RedEx, RSamp, RSampEx, IPol, Lip

Red2

Nachträgliches Abtasten, so dass die Datensatzlänge eine Zweierpotenz wird.
Alternativer Name: **Tast2**

Deklaration:

```
Red2 ( Daten ) -> Ergebnis
```

Parameter:

Daten	Abzutastender Datensatz. Erlaubte Datentypen: [ND]
Ergebnis	Ergebnis der Nachabtastung.

Beschreibung:

Diese Funktion ist speziell im Hinblick auf die Anwendung von Funktionen wie FFT(), ACF(), CCF() und Spec() geschaffen. Ein Datensatz wird nachträglich abgetastet, so dass seine Länge eine Zweierpotenz annimmt. Damit können die genannten Funktionen angewendet werden, ohne dass der Datensatz abgeschnitten wird.

Die Funktion Red2() arbeitet ähnlich wie die Funktion RSamp(). Der Datensatz wird zwischen seinen Punkten (Abtastwerten) linear interpoliert gedacht. Diese linear interpolierte Funktion wird mit einer geeigneten Abtastzeit abgetastet. Die neue Abtastzeit kann dabei etwas größer oder etwas kleiner als die ursprüngliche sein. Bei der Funktion Red2() ist das Verhältnis zwischen neuer und alter Abtastzeit i.Allg. nicht ganzzahlig.

Liegt die Länge des übergebenen Datensatzes knapp über einer Zweierpotenz (bis etwa 10% darüber), wird die Länge auf diese kleinere Zweierpotenz reduziert. Liegt die Länge oberhalb dieser Grenze, wird auf die nächstgrößere Zweierpotenz verlängert. Diese Vorgehensweise wird gewählt, damit einerseits keine merklichen Aliasing-Effekte auftreten, andererseits die Datensätze nicht unnötig groß werden (Rechenzeit!).

- Die Einheiten bleiben unverändert, die Abtastzeit wird angepasst.
- Die Annahme, dass zwischen den Abtastwerten die Werte durch lineare Interpolation approximierbar sind, ist nicht immer zulässig. Dann wird vor Anwendung der Funktion Red2 eine Spline-Interpolation empfohlen.
- Die Funktion Red2 berücksichtigt die maximal verarbeitbare Länge von Datensätzen für die Funktionen FFT, ACF, CCF und Spec. Extrem lange Datensätze werden dann auf höchstens 134.217.728 (2^27) Werte reduziert.

Beispiele:

```
MPspectr = Spec(Red2(NDdata))
```

Berechnung des Spektrums eines Datensatzes mit 2000 Punkten.

```
NDacf = ACF(Red2(NDdata))
```

Berechnung der Autokorrelationsfunktion eines Datensatzes von 6000 Punkten.

Siehe auch:

Red, RSamp, RSampEx, IPol, Lip, FFT, CCF

RedEx

Verfügbar ab: Professional Edition

Abtasten mit vorgebbarem Reduktionsfaktor und Start

Deklaration:

```
RedEx ( Datensatz, Reduktionsfaktor [, Startindex] ) -> Ergebnis
```

Parameter:

Datensatz	Datensatz
Reduktionsfaktor	Reduktionsfaktor, jedes soundsovielte Sample wird übernommen, ganze Zahl >= 1
Startindex	Startindex, >= 1, das erste Sample wird an diesem Index genommen. (optional , Standardwert: 1)
Ergebnis	Ergebnis

Beschreibung:

Ein Datensatz wird nachträglich abgetastet. Dabei wird die Datenmenge reduziert. Sie können den Faktor angeben, um den die Datenmenge reduziert wird. Beim Abtasten wird jeder n-te Wert des übergebenen Datensatzes in den zu erzeugenden Datensatz eingetragen.

Die Einheit des Datensatzes wird beibehalten, die Abtastzeit dx um den übergebenen Faktor vergrößert.

Beim Abtasten eines Datensatzes treten Aliasing-Effekte auf, wenn hohe Frequenzanteile im Datensatz vorhanden sind. Sie sollten dann vor einem Abtasten eine Glättung oder Tiefpassfilterung einsetzen, z. B. Glatt().

Ist ein Startindex > 1 angegeben, wird der x-Offset des Ergebnisses entsprechend vergrößert.

Der Datensatz darf Events und Segmente haben, er darf äquidistant, XY oder komplex sein.

Beispiele:

Der 2., 5., 8. Messwert etc. wird ausgeschnitten.

```
short = RedEx ( Data, 3, 2 )
```

Siehe auch:

Tast, IPli, IPol, PTastEx, GrenIndex, MatrixIpol

RemoveSamples

Aus einem Datensatz werden Werte entfernt.

Deklaration:

```
RemoveSamples ( Daten, EwSampleIndex, EwAnzahl, EwEventIndex, Null )
```

Parameter:

Daten	Datensatz, aus dem Werte entfernt werden sollen.
EwSampleIndex	Index des (ersten) zu löschenden Wertes. Der erste Wert des Datensatzes hat den Index 1. Bei eventierten Daten bezieht sich der Index auf den Beginn des ausgewählten Events.
EwAnzahl	Anzahl der zu löschenden Werte. -1, um bis zum Ende des Datensatzes bzw. des gewählten Events zu löschen.
EwEventIndex	Für eventierte Daten ist hier der Index des gewünschten Events (1..) anzugeben. 0 für nicht-eventierte Daten.
Null	Reservierter Parameter. Immer auf 0 zu setzen.

Beschreibung:

Die Funktion kann auf normale (gleichmäßig abgetastete) Datensätze sowie auf [XY]- und komplexe Daten angewendet werden.

Der Datentyp der Variablen bleibt erhalten, bei normalen Datensätzen wird also die Zeit- bzw. x-Achse 'zusammengeschoben'. Wenn beispielsweise die zeitliche Zuordnung aller Werte erhalten bleiben soll, muss der Datensatz vorher in einen äquivalenten XY-Datensatz konvertiert werden (siehe Beispiel).

Bei [XY] und [Komplex] wird der korrespondierende Wert in der 2. Komponente (X, Phase, Imaginärteil) ebenfalls entfernt.

Segmentierte Datensätze und Daten des Typs Zeitstempel-ASCII sind nicht erlaubt.

Die Funktion löscht beginnend mit [SampleIndex] maximal [Anzahl] Werte, höchstens aber bis zum Ende des Datensatzes bzw. des aktuellen Events.

Beispiele:

Entfernen der ersten beiden Samples eines Datensatzes.

```
RemoveSamples(Signal, 1, 2, 0, 0)
```

Das dritte Event eines Datensatzes wird nach dem 10. Sample abgeschnitten.

```
RemoveSamples(SignalWithEvents, 11, -1, 3, 0)
```

Aus einem äquidistant abgetasteten Datensatz wird ein Stück aus der Mitte entfernt, der zeitliche Bezug der nachfolgenden Samples soll jedoch erhalten bleiben. Der Datensatz wird dazu zuerst in einen XY-Datensatz gewandelt.

```
SignalXY = XYof(Ramp(xoff?(Signal), xdel?(Signal), Leng?(Signal)), Signal)
RemoveSamples(SignalXY, 200, 100, 0, 0)
```

Siehe auch:

Cut, CutIndex, SamplesGate, Repl, ReplIndex

RENAME

Eine Variable erhält einen neuen Namen.
Alternativer Name: **BENENNEN**

Deklaration:

```
RENAME AlterName NeuerName
```

Parameter:

AlterName	Alter Name der Variablen
NeuerName	Neuer Name der Variablen

Beschreibung:

Eine Variable wird umbenannt. Der alte Name wird als erster Parameter, der neue Name als zweiter Parameter übergeben.

Existiert bereits eine Variable mit dem gewünschten neuen Variablennamen, so wird diese Variable zuerst gelöscht, die alte Variable wird von der neuen also überschrieben.

Beispiele:

Ein Datensatz wird berechnet, umbenannt und angezeigt:

```
Data = ...
RENAME Data {BrakeTest - Test #1}
SHOW {BrakeTest - Test #1}
```

Ein Kanal einer Datengruppe wird umbenannt und angezeigt:

```
MyGroup:Channel1 = data
RENAME MyGroup:Channel1 MyData
SHOW MyGroup:MyData
```

Siehe auch:

SHOW, DELETE

RenameMeasurement

Eine Messung wird umbenannt.

Deklaration:

```
RenameMeasurement ( TxAlterName, TxNeuerName ) -> EwErfolg
```

Parameter:

TxAlterName	Aktueller Name der umzubenennenden Messung.
TxNeuerName	Neuer Name der Messung.
EwErfolg	Erfolg der Funktion (optional)
	0 : Es ist entweder keine Messung mit dem Namen [TxAlterName] vorhanden oder es existiert bereits eine Messung mit dem Namen [TxNeuerName].
	1 : OK

Beschreibung:

Die Funktion benennt eine Messung um.

Das Konzept der Messungszugehörigkeit einer Variablen findet bisher hauptsächlich im Datenquellen-Browser Anwendung. Dort wird beim Laden einer Messwert-Datei jeder erzeugten Variablen automatisch ein Messungsname zugeordnet, um gleichnamige Variablen, die aus verschiedenen Datenquellen stammen, bequem unterscheiden zu können. Das Ändern des Messungsnamens betrifft alle Variablen, die bisher dieser Messung zugeordnet waren und löst automatisch die Aktualisierung von Messungs- und Kanalliste in der Variablenliste/Messungsansicht aus.

Wenn [TxAlterName] aktuell in der Messungsliste des Datenselektors vermerkt ist (also einer symbolischen Messungsnummer zugeordnet ist), wird der entsprechende Listeneintrag ebenfalls umbenannt.

Bei allen Kurvenfenstern, die einen Kanal zu dieser Messung darstellen, wird der Verweis auf die Messung umbenannt. Die Anzeige bleibt erhalten.

Der neue Name darf nicht dem Namen einer bereits existierenden Messung entsprechen.

Die Funktion entspricht in der Wirkungsweise dem Befehl "Messung umbenennen" im Kontextmenü der Messungsliste in der Variablenliste/Messungsansicht.

Verzeichnis- oder Dateinamen im Dateisystem werden durch diese Funktion nicht geändert.

Beispiele:

Nachdem mit dem Datenquellen-Browser eine neue Messung angelegt wurde, wird geprüft, ob diese einen Kanal mit dem Namen 'speed' enthält. Falls ja, wird der automatisch vergebene Messungsname durch die Triggerzeit des Kanals ersetzt.

Ereignis-Sequenz 'Messung verfügbar'

```
TxVarName = SelBuildVarName(PA1, "speed", 0)
IF TxVarName <> ""
   TxNewMeasName = TimeToText(Time?(<TxVarName>), 0)
   RenameMeasurement(PA1, TxNewMeasName)
END
```

Alle aktuell geladenen Messungen, deren Namen mit dem Präfix 'BLF' beginnt, erhalten einen aussagekräftigeren Namen:

```
measurements = MeasNames?("BLF*")
FOREACH ELEMENT name IN measurements
   RenameMeasurement(name, TReplace(name, "BLF", "BottomLeftSensor"))
END
```

Siehe auch:

SetMeasurementName, MeasNames?

Repl

Ersetzt in einem Datensatz ein Stück durch neue Daten.
Alternativer Name: **Stück**

Deklaration:

```
Repl ( Daten, NeuesStück ) -> ErgebnisDaten
```

Parameter:

Daten	Zu ändernder Datensatz. Erlaubte Typen: [ND].
NeuesStück	Einzufügendes Teilstück.
ErgebnisDaten	Geänderter Datensatz mit entsprechend neuen Werten

Beschreibung:

Im Datensatz [Daten] werden diejenigen Datenpunkte durch die Datenpunkte von [NeuesStück] ersetzt, die durch den x-Offset und die Länge von [NeuesStück] definiert sind. [NeuesStück] wird also in [Daten] an der richtigen Stelle hineinkopiert. Die Funktion Repl() überträgt die Datenpunkte Punkt für Punkt, auch wenn die Abtastzeiten unterschiedlich sind.

Die Funktion Repl() ist besonders geeignet, um mit der Funktion Cut() herausgeschnittene und bearbeitete Stücken zurück in den gesamten Datensatz zu kopieren.

Datenpunkte aus dem angegebenen Stück werden nur übertragen, solange sie nicht außerhalb des Gesamt-Datensatzes liegen.

Sind die Abtastzeiten beider Datensätze verschieden, benutzen Sie die Funktion RSamp, um die Abtastzeiten anzugleichen.

Sie können alternativ mit der Funktion ReplIndex() arbeiten, wenn Sie die Einfügeposition über den Index des Punktes im Datensatz angeben wollen. Diese Funktion ist auch für XY-Datensätze geeignet.

Beispiele:

Aus einem Datensatz wird mit Hilfe der Funktion Cut() der Teil von x = 1 bis x = 2 herausgeschnitten, geglättet und anschließend mit der Funktion Repl() in den Originaldatensatz zurück geschrieben. Diese Sequenz wirkt, als hätte man den Bereich von x = 1 bis x = 2 geglättet, den Rest aber unverändert gelassen:

```
NDpart = Cut(NDdata, 1, 2)
NDpart = Smo5(NDpart)
NDdata = Repl(NDdata, NDpart)
```

Siehe auch:

ReplIndex, Cut, CutIndex, ValueIndex

ReplIndex

Ersetzt in einem Datensatz ein Stück durch neue Daten.
Alternativer Name: **StückIndex**

Deklaration:

```
ReplIndex ( Daten, NeuesStück, EwStartIndex ) -> ErgebnisDaten
```

Parameter:

Daten	Zu ändernder Datensatz. Erlaubte Typen: [ND],[XY].
NeuesStück	Einzufügendes Teilstück.
EwStartIndex	Index in [Daten], an dem das Ersetzen beginnt.
ErgebnisDaten	Geänderter Datensatz mit entsprechend neuen Werten

Beschreibung:

Im Datensatz [Daten] werden ab einer angegebenen Position [StartIndex] die Datenpunkte durch die Datenpunkte von [NeuesStück] ersetzt.

Die Länge von [NeuesStück] bestimmt die Anzahl der zu ersetzenden Datenpunkte.

[StartIndex] muss zwischen 1 und der Datensatzlänge von [Daten] liegen.

Die Länge des Ergebnisses und des ersten Parameters sind gleich, ggf. werden überzählige Werte von [NeuesStück] ignoriert (Warnung wird erzeugt).

Die Datentypen von [Daten] und [NeuesStück] müssen gleich sein. Falls es sich um XY-Datensätze handelt, werden auch die X-Koordinaten ersetzt. Bei reellen Datensätzen werden nur die Y-Werte ersetzt, die X-Skalierung (Offset und Abtastzeit bzw. x-Inkrement) bleibt unverändert.

Die Funktion ist besonders geeignet, um mit der Funktion CutIndex() herausgeschnittene und bearbeitete Stücke zurück in den gesamten Datensatz zu kopieren.

Alternativ kann die Funktion Repl() verwendet werden, bei der die Ersetzungsposition durch die X-Koordinaten des ersetzenden Datensatzes bestimmt wird.

Beispiele:

Bis auf den ersten und letzten Wert werden alle y-Werte eines Datensatzes verdoppelt:

```
part = CutIndex(Data, 2, leng?(Data)-1)
part = part * 2
Data = ReplIndex(Data, part, 2)
```

Siehe auch:

Repl, Cut, CutIndex, ValueIndex, MatrixMerge, MatrixFromLine

REQUEST

Verfügbar ab: Professional Edition

Daten über DDE empfangen
Alternativer Name: **ABFRAGEN**

Der Befehl ist veraltet, statt dessen sollte die leistungsfähigere Funktion DDEInq() verwendet werden.

Deklaration:

```
REQUEST Applikation Thema Eintrag Variablenname
```

Parameter:

Applikation	Name der anzusprechenden DDE-Applikation (engl.:Application).
Thema	Bezeichnung des DDE-Themas (engl.: Topic).
Eintrag	Name der abzufragenden Variablen (engl.: Item).
Variablenname	Name der empfangenden FAMOS-Variablen

Beschreibung:

imc FAMOS arbeitet als DDE-Client, die angesprochene Applikation als DDE-Server. Über die Einträge "Applikation" und "Thema" wird eine andere DDE-fähige Applikation von imc FAMOS ausgewählt und eine Konversation angefordert. Wird der Anforderung entsprochen, überträgt imc FAMOS den Namen des gewünschten Eintrags. imc FAMOS wartet dann, bis der Server die angeforderten Daten sendet oder die Anforderung negativ beantwortet. Danach beendet imc FAMOS die Konversation.

Die unter [Variablenname] angegebene Variable enthält nun die empfangenen Daten. War die Übertragung nicht möglich, enthält sie den Rückgabewert der angesprochenen DDE-Applikation.

Die Daten können sowohl Einzelwerte, als auch Datensätze im ASCII- oder FAMOS-DDE-Format sein.

- Die einzelnen Parameter des REQUEST-Befehls dürfen keine Leerzeichen enthalten.
- FAMOS erzeugt eine Fehlermeldung, wenn die angesprochene DDE-Applikation (Server) nicht antwortet oder die Anforderung nicht akzeptiert.
- Ist die angesprochene Applikation (Server) beschäftigt (engl. busy), wartet imc FAMOS, bis sie frei ist.
- FAMOS-Befehle können wahlweise groß oder klein geschrieben werden (engl.: Case insensitive), wogegen einige DDE-Applikationen Unterscheidungen treffen (engl.: Case sensitive). Achten Sie also auf richtige Schreibweise.

Beispiele:

```
REQUEST Trans Zeitbasis TB x
```

"Trans" ist der Name einer fiktiven DDE-fähigen Applikation, "Zeitbasis" ist der Name des DDE-Themas. Der Rückgabewert wird in die Variable "x" geschrieben.

```
REQUEST Trans1 Status Trigger Svtrig
REQUEST Trans2 Status Armiert SVarm
```

Daten können von variierenden DDE-Applikationen angefordert werden. Nach Ausführung dieses Befehls enthalten die Einzelwert-Variablen Informationen über den Status der beiden "Trans"-Geräte. Z.B. könnte die Variable SVtrig gleich 1.0 sein, wenn der Trigger von "Trans1" ausgelöst war.

Siehe auch:

DDEInq, DDESend, DDESet

RGB

Bilden eines Farbwertes aus den Farbanteilen

Deklaration:

```
RGB ( EwRot, EwGrün, EwBlau ) -> EwFarbWert
```

Parameter:

EwRot	Rot-Anteil der Farbe 0..255
EwGrün	Grün-Anteil der Farbe 0..255
EwBlau	Blau-Anteil der Farbe 0..255
EwFarbWert	Farbwert

Beschreibung:

Aus den Intensitäten (0..255) der 3 Grundfarben wird ein Farbwert ermittelt, welcher der Funktion SetFarbe() übergeben werden kann.

Datenformat des Ergebnisses: 4 Byte ganzzahlig ohne Vorzeichen.

Ab Version 7.5: Als Parameter können nicht nur Einzelwerte, sondern auch Datensätze übergeben werden, die dann jeweils Wert für Wert miteinander verrechnet werden. Events und Segmente sind erlaubt, allerdings müssen alle 3 Parameter exakt dieselbe Struktur aufweisen (gleiche Gesamt- und Segmentlänge, gleiche Eventlängen). Der Ergebnisdatensatz weist dann dieselbe Strukturierung auf. Bei Segmentierung wird angenommen, dass das Ergebnis als Bilddaten im RGB-Format interpretiert werden kann und das entsprechende interne Flag gesetzt, siehe auch Funktion SetFlag().

Beispiele:

```
fb = RGB(255, 0, 0) ;Rot
fb = RGB(0, 0, 0) ;Schwarz
fb = RGB(0, 0, 255) ;Blau
fb = RGB(255, 255, 255) ;Weiß
```

Im folgenden Beispiel wird die Video-Datei 'sample.avi' in den aktuellen Panel-Videoplayer geladen und das 100. Bild extrahiert. Anschließend wird das Bild zunächst in Graustufen und schließlich in ein Scharz/Weiß-Bild gewandelt.

```
VpVideoLoad("c:\tmp\sample.avi", 1)
VpSetPosFrames(100, 1)
image = VpGetImages(1)
RedValue =   RGBConvert(image, "R")
GreenValue = RGBConvert(image, "G")
BlueValue =  RGBConvert(image, "B")
; Gewichtete Graustufen-Konvertierung
; Koeffizienten entsprechend "Luma coding" nach "ITU-R Recommendation BT.601":
GreyValue = 0.3* RedValue + 0.59* GreenValue + 0.11 *BlueValue
imageGrey = RGB(GreyValue, GreyValue, GreyValue)
; Konvertierung nach Schwarz/Weiß. Alle Grauwerte > 127 werden zu weiß, alle anderen schwarz:
BWValue = (GreyValue > 127)*255
imageBlackWhite = RGB(BWValue, BWValue, BWValue)
```

Erzeugung eines Bilddatensatzes mit 640x480 Pixeln, alle Pixel rot:

```
r = Leng(0, 640*480) + 255
g = r*0
b = r*0
image = RGB(r, g, b)
SetSegLen(image, 640)
SetFlag(image, 1, 1)
```

Siehe auch:

RGBConvert, SetColor, Color?, SetFlag, VpGetImages

RGBConvert

Verfügbar ab: Professional Edition

Gegebene RGB-Daten werden konvertiert bzw. verarbeitet.

Deklaration:

```
RGBConvert ( RGB_Daten, Berechnung [, Parameter1] [, Parameter2] ) -> Ergebnis
```

Parameter:

RGB_Daten	Eingangsdaten im RGB-Format
Berechnung	Berechnung
	"R" : Rot-Anteil (R von RGB) im Bereich 0 .. 255
	"G" : Grün-Anteil (G von RGB) im Bereich 0 .. 255
	"B" : Blau-Anteil (B von RGB) im Bereich 0 .. 255
	"RGB" : Eingangsdaten, die als RGB Farben zu deuten sind, werden in das 4 Byte RGB Format mit Farbflag konvertiert. Dabei wird gerundet.
	"GREY" : Eingangsdaten, die im Bereich 0 .. 255 vorliegen, werden in das 1 Byte Graustufenformat mit Farbflag konvertiert. Dabei wird gerundet und auf den Wertebereich 0 .. 255 begrenzt.
	"bright1" : Bei RGB-Eingangsdaten die Helligkeit sqrt(0.299*R^2 + 0.587*G^2 + 0.114*B^2) im Bereich 0 .. 255
	"bright2" : Bei RGB-Eingangsdaten die Helligkeit (0.2126*R + 0.7152*G + 0.0722*B) im Bereich 0 .. 255
	"bright3" : Bei RGB-Eingangsdaten die Helligkeit (0.299*R + 0.587*G + 0.114*B) im Bereich 0 .. 255
	"HSV-H" : Bei RGB-Eingangsdaten der Farbwert. H-Anteil (hue) der HSV-Darstellung im Bereich 0 .. 359. Z.B. 0 für rot, 120 für grün, 240 für blau.
	"HSV-S" : Bei RGB-Eingangsdaten die Farbsättigung. S-Anteil der HSV-Darstellung im Bereich 0 .. 255, entspricht 0 .. 100%.
	"HSV-V" : Bei RGB-Eingangsdaten der Hellwert. V-Anteil der HSV-Darstellung im Bereich 0 .. 255, entspricht 0 .. 100%.
	"HSL-H" : Bei RGB-Eingangsdaten der Farbwert. H-Anteil (hue) der HSL-Darstellung im Bereich 0 .. 359. Z.B. 0 für rot, 120 für grün, 240 für blau.
	"HSL-S" : Bei RGB-Eingangsdaten die Farbsättigung. S-Anteil der HSL-Darstellung im Bereich 0 .. 255, entspricht 0 .. 100%.
	"HSL-L" : Bei RGB-Eingangsdaten der Hellwert. L-Anteil der HSL-Darstellung im Bereich 0 .. 255, entspricht 0 .. 100%.
	"add" : Addiert bei RGB-Eingangsdaten den RGB-Wert, der in Parameter1 gegeben ist.
	"sub" : Subtrahiert bei RGB-Eingangsdaten den RGB-Wert, der in Parameter1 gegeben ist.
	"inv" : Eingangsdaten, die im Bereich 0 .. 255 vorliegen, werden invertiert: Ergebnis = 255 - Eingangsdaten. Das Ergebnis liegt im Bereich 0 .. 255 vor.
	"1.GREY" : Eingangsdaten, die im Bereich 0.0 .. 1.0 vorliegen, werden in das 1 Byte Graustufenformat mit Farbflag konvertiert. Dabei wird gerundet und auf den Wertebereich 0 .. 255 begrenzt.
	"similar12" : Bestimmt bei RGB-Eingangsdaten die jeweils größte Ähnlichkeit eines RGB-Wertes zu einem Farbübergang zwischen den in Parameter1 und Parameter2 gegebenen RGB-Farben. Das Resultat liegt bei 255 für Gleichheit und 0 für überhaupt keine Übereinstimmung.
Parameter1	1. Parameter, Bedeutung je nach Berechnung (optional , Standardwert: 0)
Parameter2	2. Parameter, Bedeutung je nach Berechnung (optional , Standardwert: 0)
Ergebnis	Ergebnis

Beschreibung:

Die äquidistanten Eingangsdaten können Events und Segmente haben.

Bei Ergebnissen im Bereich 0..255 (als Graustufen zu deuten) und Ergebnissen mit RGB-Werten wird das Farbflag (Flag für Farbinformation) gesetzt.

Beispiele:

R, G, B aus RGB bestimmen

```
colors = RGB(200, 100, 0)
R = RGBConvert( colors, "R" )
G = RGBConvert( colors, "G" )
B = RGBConvert( colors, "B" )
```

Helligkeit bestimmen. Ergibt Graustufen: 0..255

```
colors = RGB(200, 100, 0)
bright1 = RGBConvert( colors, "bright1" )
bright2 = RGBConvert( colors, "bright2" )
bright3 = RGBConvert( colors, "bright3" )
bright4 = RGBConvert( colors, "HSV-V" )
bright5 = RGBConvert( colors, "HSL-L" )
```

Alle Stellen eines Bildes markieren, die so etwa grün bis dunkelgrün sind.

```
picture = Random(36,0, 0, 256*256*255, 17 )
setseglen(picture,6)
picture = RGBConvert( picture, "RGB" )
picture = MatrixIpol(picture,10,10,1)
sim = RGBConvert( picture, "similar12", RGB(0,150,0), RGB(0,250,0) )
decide = ( sim > 170 )
decide = RGBConvert( decide, "1.GREY" )
```

Siehe auch:

RGB, SetFlag

RgCurveSet

Anwendungsbereich: Reportgenerator

Der Inhalt eines Kurvenobjektes im aktiven Dokument wird gesetzt.

Deklaration:

```
RgCurveSet ( Titel, Kurve, Null ) -> Fehlercode
```

Parameter:

Titel	Titel des Kurven-Objektes im Dokument
Kurve	Auswahl des zu übetragenden Kurvenfensters. Siehe Beschreibung.
Null	Reservierter Parameter, auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Ein Kurvenobjekt im aktiven Dokument wird mit dem Inhalt eines geöffneten Kurvenfensters belegt. Falls das Kurvenobjekt bereits mit einer Kurvengrafik gefüllt ist, wird diese ersetzt.

Auswahl des zu übertragenden Kurvenfensters (2.Parameter):

Bei **freien Kurvenfenstern** erfolgt die Identifizierung des Kurvenfensters über dessen Bezugsdatensatz. Der Bezugsdatensatz eines Kurvenfensters wird beim Erzeugen des Kurvenfensters mit den Cw*(..)-Funktionen des Kurven-Kits oder dem FAMOS-Befehl "SHOW" festgelegt. Oftmals entspricht er dem ersten angezeigten Datensatz im Fenster und dem Fenstertitel.

Bei Kurvenfenstern, die in einem **anwenderdefinierten Dialog eingebettet** sind, wird der Name des Dialog-Elementes angegeben.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

Wie die Grafik im Kurvenobjekt positioniert und angezeigt wird, hängt von den Einstellungen des Kurvenobjektes im Report ab. Dies betrifft insbesondere die Positionierung der Grafik und die Grafik-Eigenschaften, wie Linienstärken etc. Dafür werden entweder die aktuellen Einstellungen am Kurvenfenster oder die im Kurvenobjekt gespeicherten Einstellungen verwendet.

- Die Funktion kann nur angewendet werden, wenn eine Druckbild-Konfiguration geladen ist.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Datensatz wird geladen und in einem Kurvenfenster angezeigt. Ein Report wird geladen, ein enthaltenes Kurvenobjekt mit dem Kurvenfenster belegt und der Report gedruckt.

```
Kanal1 = FileObjRead(file, 1)
TxErr$ = CwLoadCCV(Kanal1, "kanal1.ccv")
err = RgDocOpen("Tabelle.drb", 0)
IF err = 0
   err = RgCurveSet("Curve1", Kanal1, 0)
   IF err = 0
      RgDocPrint(0)
   END
   RgDocClose(0)
END
```

Siehe auch:

RgDocOpen , DrSetzen

RgDocClose

Anwendungsbereich: Reportgenerator

Der Reportgenerator schließt das aktive Dokument.

Deklaration:

```
RgDocClose ( Option ) -> Fehlercode
```

Parameter:

Option	Optionsparameter
	0 : Das aktive Dokument wird geschlossen.
	1 : Alle Dokumente werden geschlossen.
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das aktive Dokument oder alle Dokumente werden geschlossen. Der Reportgenerator selbst bleibt geöffnet.

Die Dokumente werden nicht gesichert, eventuelle Änderungen gehen verloren.

- Jedes mit RgDocOpen geöffnete Dokument sollte nach erfolgter Bearbeitung wieder geschlossen werden. Dadurch wird verhindert, dass viele Dokumente unnötigerweise gleichzeitig geladen sind.
- Solange zu einem Zeitpunkt nur 1 Dokument geöffnet ist, ist auch das aktive Dokument immer eindeutig bestimmt. Sonst muss durch Verwendung der Funktion RgDocSetActive sichergestellt werden, dass auch das gewünschte Dokument angesprochen wird.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Report wird geladen. Eine in diesem enthaltene Tabelle wird mit neu berechneten Daten gefüllt. Der Report wird gedruckt und wieder geschlossen.

```
Daten = ...
err = RgDocOpen("Tabelle.drb", 0)
IF err = 0
   err = RgTableSetColumn("tab1", 2, 2, Daten, 0)
   IF err = 0
      RgDocPrint(0)
   END
   RgDocClose(0)
END
```

Siehe auch:

RgDocOpen, RgDocSetActive, RgWindow

RgDocCopy

Anwendungsbereich: Reportgenerator

Eine Seite des aktiven Dokuments wird komplett in die Zwischenablage kopiert.

Deklaration:

```
RgDocCopy ( Seitennummer ) -> Fehlercode
```

Parameter:

Seitennummer	Seitennummer der zu kopierenden Seite (1..).
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das aktive Dokument wird komplett in die Zwischenablage kopiert. Dabei wird das Windows-Metafile-Format verwendet.

- Zwecks Kompatibiltät mit älteren Versionen ist für den Parameter [Seitennummer] auch eine 0 zulässig. Es wird dann die erste Seite kopiert.
- Solange zu einem Zeitpunkt nur 1 Dokument geöffnet ist, ist auch das aktive Dokument immer eindeutig bestimmt. Sonst muss durch Verwendung der Funktion RgDocSetActive sichergestellt werden, dass auch das gewünschte Dokument angesprochen wird.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Report wird geladen und komplett in die Zwischenablage kopiert. Anschließend wird "WinWord" per DDE angewiesen, den Inhalt der Zwischenablage in das aktuelle Word-Dokument einzufügen.

```
err = RgDocOpen("Tabelle.drb", 0)
IF err = 0
   err = RgDocCopy(0)
   IF err = 0
      ret = DDESend("WinWord", "System", "[EditPaste]")
   END
   RgDocClose(0)
END
```

Siehe auch:

RgDocOpen, RgDocSetActive, RgWindow

RgDocExport

Anwendungsbereich: Reportgenerator

Der Reportgenerator exportiert das aktive Dokument in ein wählbares Grafik-Format.

Deklaration:

```
RgDocExport ( TxDateiName, Format, Auflösung, Farbtiefe/PDF-Methode, Option ) -> Fehlercode
```

Parameter:

TxDateiName	Dateiname, unter dem das aktive Dokument zu speichern ist.
Format	Grafik-Format
	0 : Windows-Metadatei (*.emf)
	1 : Aldus Placeable Metadatei (*.wmf)
	2 : Windows Bitmap (*.bmp)
	3 : JPEG-FileFormat (*.jpg)
	4 : Portable Networks Graphic-FileFormat (*.png)
	5 : PDF-Format (*.pdf)
Auflösung	Gibt die zu wählende Auflösung bei den Bitmap-Formaten (BMP, PNG, JPG) an. Die Einheit ist 'dpi' (Dots per Inch = Punkte pro Zoll). Übliche Werte sind z.B. 150dpi oder 300dpi. Eine 0 bedeutet, dass die Voreinstellung am Reportgenerator verwendet wird. Bei anderen Formaten als BMP, JPG, PNG auf 0 zu setzen.
Farbtiefe/PDF-Methode	Gibt den zu erzeugenden Farbtyp bei den Bitmap-Formaten BMP und PNG bzw. die zu verwendende Exportmethode bei PDF an. Bei anderen Formaten auf 0 zu setzen. Die Bedeutung ist also abhängig vom Wert des [Format]-Parameters:
	5 : PDF: Exportmethode
	0 Die Voreinstellung in FAMOS ("Optionen"/"Datei-Export"/"PDF") wird verwendet. **1** auto (minimale Dateigröße) **2** Bitmap **3** Vektorgrafik bevorzugt
	2,4 : Bitmap: Farbtiefe
	0 Voreinstellung Reportgenerator **1** 16 Farben **2** 256 Farben **3** 16 Mio Farben
Option	JPEG: Qualität (in Prozent 10%..100%), eine 0 bedeutet, dass die Voreinstellung am Reportgenerator verwendet wird. Bei anderen Formaten als JPEG auf 0 zu setzen.
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Reportgenerator exportiert das aktive Dokument unter dem angegebenen Dateinamen und in dem angegebenen Format.

Mehrseitige Dokumente: Falls das Dokument über mehrere Seiten verfügt, werden im PDF-Format alle Seiten exportiert. Bei allen anderen Formaten wird die erste Seite verwendet. Wenn Sie eine andere Seite exportieren wollen, verwenden Sie die Funktion [RgDocExportEx].

Wenn für den Dateinamen ein leerer Text angegeben wird, wird der Report unter seinem aktuellen Titel abgelegt.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird die Standarderweiterung für das gewählte Dateiformat verwendet.

Falls kein voller Pfadname angeben ist, wird bei aktivem Projekt das aktuelle Projektverzeichnis verwendet. Ansonsten wird das Verzeichnis benutzt, dass durch den letzten Aufruf von RgSetDir festgelegt worden ist. Falls diese Funktion noch nicht aufgerufen wurde, wird das in FAMOS festgelegte Standardverzeichnis (Verzeichnis für Druckbilddateien) verwendet.

- Der exportierte Bereich ist das kleinstmögliche Rechteck, welches alle Reportobjekte enthält und ist damit i.a. kleiner als die Blattgröße.
- Solange zu einem Zeitpunkt nur 1 Dokument geöffnet ist, ist auch das aktive Dokument immer eindeutig bestimmt. Sonst muss durch

Verwendung der Funktion RgDocSetActive sichergestellt werden, dass auch das gewünschte Dokument angesprochen wird.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.
- Soll ein Dateiname keine Dateinamen-Erweiterung haben, wird der Dateiname mit einem "." abgeschlossen.

Beispiele:

Ein Report wird geladen. Ein enthaltenes Kurvenobjekt wird aktualisiert. Der Report wird unter neuem Namen im sowohl im JPEG-Format (Qualität 100%, Auflösung 150dpi) als auch im PDF-Format gespeichert und wieder geschlossen.

```
CwNewWindow(Daten, "show")
err = RgDocOpen("cu_mask", 0)
IF err = 0
   err = RgCurveSet("curve1", Daten, 0)
   IF err = 0
      err = RgDocExport("cu_0001.jpg", 3, 150, 0, 100)
      err = RgDocExport("cu_0001.pdf", 5, 0, 0, 0)
   END
   RgDocClose(0)
END
```

Siehe auch:

RgDocOpen, RgDocSave, RgDocSetActive, RgWindow, RgDocExportEx

RgDocExportEx

Anwendungsbereich: Reportgenerator

Der Reportgenerator exportiert die angegebene Seite des aktiven Dokuments in ein wählbares Grafik-Format.

Deklaration:

```
RgDocExportEx ( TxDateiName, Seitennummer, Format, Auflösung, Farbtiefe/PDF-Methode, Option ) -> Fehlercode
```

Parameter:

TxDateiName	Dateiname, unter dem das exportierte Dokument zu speichern ist.
Seitennummer	Gibt die Nummer (1..) der zu exportierenden Seite an. Für PDF ist hier auch eine 0 zulässig, dann wird das gesamte Dokument exportiert.
Format	Grafik-Format
	0 : Windows-Metadatei (*.emf)
	1 : Aldus Placeable Metadatei (*.wmf)
	2 : Windows Bitmap (*.bmp)
	3 : JPEG-FileFormat (*.jpg)
	4 : Portable Networks Graphic-FileFormat (*.png)
	5 : PDF-Format (*.pdf)
Auflösung	Gibt die zu wählende Auflösung bei den Bitmap-Formaten (BMP, PNG, JPG) an. Die Einheit ist 'dpi' (Dots per Inch = Punkte pro Zoll). Übliche Werte sind z.B. 150dpi oder 300dpi. Eine 0 bedeutet, dass die Voreinstellung am Reportgenerator verwendet wird. Bei anderen Formaten als BMP, JPG, PNG auf 0 zu setzen.
Farbtiefe/PDF-Methode	Gibt den zu erzeugenden Farbtyp bei den Bitmap-Formaten BMP und PNG bzw. die zu verwendende Exportmethode bei PDF an. Bei anderen Formaten auf 0 zu setzen. Die Bedeutung ist also abhängig vom Wert des [Format]-Parameters:
	5 : PDF: Exportmethode
	0 Die Voreinstellung in FAMOS ("Optionen"/"Datei-Export"/"PDF") wird verwendet. **1** auto (minimale Dateigröße) **2** Bitmap **3** Vektorgrafik bevorzugt
	2,4 : Bitmap: Farbtiefe
	0 Voreinstellung Reportgenerator **1** 16 Farben **2** 256 Farben **3** 16 Mio Farben
Option	PDF: Geben Sie eine 1 an, wenn an eine bereits existierende Datei angehängt werden soll. JPEG: Qualität (in Prozent 10%..100%), eine 0 bedeutet, dass die Voreinstellung am Reportgenerator verwendet wird. Bei anderen Formaten auf 0 zu setzen.
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Reportgenerator exportiert die gewählte Seite des aktiven Dokuments unter dem angegebenen Dateinamen und in dem angegebenen Format.

Wenn für den Dateinamen ein leerer Text angegeben wird, wird der Report unter seinem aktuellen Titel abgelegt.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird die Standarderweiterung für das gewählte Dateiformat verwendet.

Falls kein voller Pfadname angeben ist, wird bei aktivem Projekt das aktuelle Projektverzeichnis verwendet. Ansonsten wird das Verzeichnis benutzt, dass durch den letzten Aufruf von RgSetDir festgelegt worden ist. Falls diese Funktion noch nicht aufgerufen wurde, wird das in FAMOS festgelegte Standardverzeichnis (Verzeichnis für Reportdateien) verwendet.

- Der exportierte Bereich ist das kleinstmögliche Rechteck, welches alle Reportobjekte enthält und ist damit i.a. kleiner als die Blattgröße.
- Solange zu einem Zeitpunkt nur 1 Dokument geöffnet ist, ist auch das aktive Dokument immer eindeutig bestimmt. Sonst muss durch Verwendung der Funktion RgDocSetActive sichergestellt werden, dass auch das gewünschte Dokument angesprochen wird.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.
- Soll ein Dateiname keine Dateinamen-Erweiterung haben, wird der Dateiname mit einem "." abgeschlossen.

Beispiele:

Die Reportvorlage "template.drb" enthält 2 Seiten mit gleichem Aufbau, die mit verschiedenen Kurvenbildern befüllt werden. Der Report wird unter neuem Namen sowohl **seitenweise** im JPEG-Format (Qualität 100%, Auflösung 150dpi) als auch komplett im PDF-Format gespeichert und wieder geschlossen.

```
RgDocOpen("template.drb",0)

; 1.Seite füllen
RgDocSetActivePage(1)
CwLoadCCV(data1, "data1.ccv")
RgCurveSet("curve1", data1, 0)

; 2. Seite füllen
RgDocSetActivePage(2)
CwLoadCCV(data2, "data2.ccv")
RgCurveSet("curve1", data2, 0)

RgDocExportEx("cu_0001_page1.jpg",1, 3, 150, 0, 100)
RgDocExportEx("cu_0001_page2.jpg",2, 3, 150, 0, 100)
RgDocExport("cu_0001.pdf", 5, 0, 0, 0)
```

Siehe auch:

RgDocOpen, RgDocSave, RgDocSetActive, RgWindow

RgDocGetPageCount

Anwendungsbereich: Reportgenerator

Die Anzahl der Seiten im aktuellen Report wird ermittelt.

Deklaration:

```
RgDocGetPageCount ( ) -> Seitenzahl
```

Parameter:

Seitenzahl	
	>= 0 : Anzahl der Seiten
	< 0 : Fehlercode.

Beschreibung:

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Im aktuellen Dokument werden alle Seiten bis auf die ersten 2 gelöscht.

```
LastPage = RgDocGetPageCount()
WHILE LastPage > 2
   RgDocRemovePage(LastPage)
   LastPage = LastPage-1
END
```

Siehe auch:

RgDocInsertPage

RgDocInsertPage

Anwendungsbereich: Reportgenerator

In den aktuellen Report wird eine neue Seite eingefügt.

Deklaration:

```
RgDocInsertPage ( TxVorlagenDatei, SeitenNummer, EinfügePosition, Null ) -> Fehlercode
```

Parameter:

TxVorlagenDatei	Dateiname der Reportdatei, die die Vorlage für die neu einzufügende Seite enthält. Wenn der Parameter leer ist, wird das aktuelle Dokument verwendet.
SeitenNummer	Legt fest, welche Seite aus [TxVorlagenDatei] zu verwenden ist.
EinfügePosition	Die neue Seite wird an der hier angegebenen Position eingefügt. Eine 0 bedeutet, dass am Ende angehängt wird.
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Mit dieser Funktion kann dem aktuellen Report eine neue Seite zugefügt werden. Die neue Seite kann entweder ein Duplikat einer Seite des aktuellen Reports sein oder aus einer anderen Reportdatei importiert werden.

Die neue Seite wird aktiviert, d.h. nachfolgende Aufrufe von objektspezifischen Funktionen wirken nur auf diese Seite (siehe auch RgDocSetActivePage(..)).

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Der Datensatz "u0" soll in Form einer 2-spaltigen Tabelle (x,y) dokumentiert werden. Der Report "table.drb" dient als Seiten-Vorlage, er enthält u.a. ein Tabellenobjekt mit 50 Zeilen.

```
Length = leng?(u0)
Rows = 50

FirstSample = 1
WHILE FirstSample <= Length
   IF FirstSample = 1
      ; Vorlage öffnen
      RgDocOpen("table.drb", 0)
   ELSE
      ; neue Seite anhängen
      RgDocInsertPage("table.drb", 1, 0, 0)
   END

   ; y-Daten für Tabelle ausschneiden
   y = CutIndex(u0, FirstSample, FirstSample+Rows-1)
   RgTableSetColumn("table", 2, 1, y, 0)

   ; x-Daten für Tabelle konstruieren
   x = Ramp((FirstSample-1)*xdel?(u0)+ xoff?(u0), xdel?(u0), leng?(y))
   RgTableSetColumn("table", 1, 1, x, 0)

   FirstSample = FirstSample+Rows
END
RgDocSave("u0_table.drb", 0)
RgDocClose(0)
```

Siehe auch:

RgDocRemovePage

RgDocNew

Anwendungsbereich: Reportgenerator

Der Reportgenerator erzeugt ein neues Dokument.

Deklaration:

```
RgDocNew ( Titel, Null ) -> Fehlercode
```

Parameter:

Titel	Titel für das neue Element
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Reportgenerator wird angewiesen, ein neues Dokument unter dem angegebenen Titel zu öffnen.

Zur Zeit bereits geöffnete Dokumente im Reportgenerator bleiben weiterhin geöffnet, das neu erzeugte Dokument wird zum aktiven Dokument.

Falls der Reportgenerator noch nicht geöffnet ist, wird er als Symbol gestartet. Ansonsten wird er in den Vordergrund gebracht.

- Im allgemeinen ist die Verwendung dieser Funktion nicht sinnvoll, da das komplett neue Erzeugen eines Reportes per Fernsteuerung bisher nur eingeschränkt möglich ist. Normalerweise wird eine Druckbildmaske manuell erzeugt und diese dann mit RgDocOpen geladen.
- Wenn ein leerer Titel ("") angegeben wird, wird ein Standardtitel ("Report" + lfd. Nummer) verwendet.
- Der angegebene Titel dient zur vorläufigen Identifikation des Dokumentes (Funktion RgDocSetActive). Beim Speichern des Dokumentes (Funktion RgDocSave) kann bei Bedarf dann ein kompletter Pfad angegeben werden, unter dem der Report gespeichert werden soll.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein neuer Report wird erzeugt und mit einer in der Zwischenablage vorhandenen Bitmap-Grafik belegt. Der Report wird gedruckt und ohne abzuspeichern wieder geschlossen.

```
err = RgDocNew("temp", 0)
IF err = 0
   err = DrRdClip("", 5, 10, 0, 0, 2, 0, 0)
   IF err = 0
      RgDocPrint(0)
   END
   RgDocClose(0)
END
```

Siehe auch:

RgDocOpen, RgDocClose, RgDocSetActive, RgWindow, DrRdClip

RgDocOpen

Anwendungsbereich: Reportgenerator

Der Reportgenerator lädt die angegebene Datei.

Deklaration:

```
RgDocOpen ( TxDateiName, Null ) -> Fehlercode
```

Parameter:

TxDateiName	Name der zu ladenden Datei.
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Reportgenerator wird angewiesen, die angegebene Report-Datei zu laden. Zur Zeit bereits geöffnete Dokumente im Reportgenerator bleiben weiterhin geöffnet, das neu geladene Dokument wird zum aktiven Dokument.

Falls der Reportgenerator noch nicht geöffnet ist, wird er als Symbol gestartet. Ansonsten wird er in den Vordergrund gebracht.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird ".drb" angenommen.

Wenn bei dem Dateinamen kein kompletter Pfad angegeben ist, wird die Reportdatei nacheinander in den folgenden Verzeichnissen gesucht:

- Aktuelles Arbeitsverzeichnis: Dieses entspricht dem Verzeichnis, aus dem die aufrufende Sequenz geladen wurde.
- Projektverzeichnis: Wenn ein Projekt geladen ist, wird danach im aktuellen Projektverzeichnis gesucht.
- Standardverzeichnis für Reportdateien: Dieses Verzeichnis ist durch den letzten Aufruf von RgSetDir festgelegt. Falls diese Funktion noch nicht aufgerufen wurde, wird die FAMOS-Voreinstellung für Reportdateien verwendet.

Jedes mit RgDocOpen geöffnete Dokument sollte nach erfolgter Bearbeitung wieder mit RgDocClose geschlossen werden. Dadurch wird verhindert, dass viele Dokumente unnötigerweise gleichzeitig geladen sind.

Solange zu einem Zeitpunkt immer nur 1 Dokument geöffnet ist, ist auch das aktive Dokument immer eindeutig bestimmt. Sonst muss durch Verwendung der Funktion RgDocSetActive sichergestellt werden, dass auch das gewünschte Dokument angesprochen wird.

Hat der Dateiname keine Dateinamens-Erweiterung, muss er mit einem "." abgeschlossen werden.

Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Multithreading: Die Funktionen des Report-Kits dürfen überall aufgerufen werden und wirken global. Der hier geladene Report ist also für alle Ausführungs-Threads gültig.

Beispiele:

Ein Report wird geladen. Ein enthaltenes Kurvenobjekt wird aktualisiert. Der Report wird gedruckt und wieder geschlossen.

```
CwNewWindow(Daten, "show")
err = RgDocOpen("Kurve.drb", 0)
IF err = 0
   err = RgCurveSet("curve1", Daten, 0)
   IF err = 0
      RgDocPrint(0)
   END
   RgDocClose(0)
END
```

Siehe auch:

RgDocClose, RgDocSetActive, RgWindow

RgDocPrint

Anwendungsbereich: Reportgenerator

Der Reportgenerator druckt das aktive Dokument.

Deklaration:

```
RgDocPrint ( Auswahl ) -> Fehlercode
```

Parameter:

Auswahl	Auswahl des zu druckenden Bereichs. Geben Sie entweder eine 0 an, um das gesamte Dokument zu drucken, oder die gewünschte Seitennummer.
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Reportgenerator wird angewiesen, das aktive Dokument zu drucken. Es wird dabei der Drucker verwendet, der am Druckbild unter "Datei: Drucker einrichten" eingestellt ist.

- Zur Ausführung dieser Funktion muss der Reportgenerator geöffnet und mindestens 1 Dokument geladen sein. Bei mehreren geöffneten Dokumenten sollten Sie mit RgDocSetActive sicherstellen, dass tatsächlich das von Ihnen gewünschte Dokument gedruckt wird.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText(..) erfragt werden.

Beispiele:

Ein Report wird geladen, gedruckt und wieder geschlossen.

```
err = RgDocOpen("Tabelle.drb", 0)
IF err = 0
   err = RgDocPrint(0)
   IF err < 0
      TxError$=RgGetErrorText(err)
      ok=BoxMessage("Drucken",TxError$,"!1")
   END
   RgDocClose(0)
END
```

Siehe auch:

RgDocOpen, RgDocSetActive

RgDocPrintSetup

Anwendungsbereich: Reportgenerator

Festlegung des zu verwendenden Druckers für einen nachfolgenden Ausdruck des Dokumentes.

Deklaration:

```
RgDocPrintSetup ( TxDrucker, TxAusgabe, Null ) -> Fehlercode
```

Parameter:

TxDrucker	Name des Druckers
TxAusgabe	Name des Ausgabemediums. Entweder leerer Text für Standard oder ein Dateiname.
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Mit dieser Funktion kann der zu verwendende Drucker und das Ausgabemedium für einen nachfolgenden Ausdruck des Dokumentes mittels RgDocPrint() festgelegt werden.

Eine sinnvolle Anwendung dieser Funktion ist beispielsweise die Umleitung des Ausdrucks in eine Datei.

Der Druckername (erster Parameter) muss genauso angegeben werden, wie er im Dialog "Drucker einrichten" am Reportgenerator aufgelistet ist.

Sinnvolle Angaben für den Ausgabenamen (zweiter Parameter) sind entweder ein leerer Text oder ein Dateiname. Im Falle eines leeren Textes wird der Standardanschluß des Druckers verwendet, wie er in der Windows-Systemsteuerung definiert ist (z.B. lokaler Druckerport LPT1). Wenn dagegen ein Dateiname angegeben wird, wird der Ausdruck in die angegebene Datei umgeleitet.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Report wird geladen und mit dem "Acrobat PDF Writer" in eine Datei gedruckt. Die resultierende PDF-Datei kann mit dem "Acrobat Reader" gelesen werden

```
err = RgDocOpen("Report1.drb", 0)
IF err = 0
   err = RgDocPrintSetup("Acrobat PDFWriter", "c:\tmp\Report1.pdf", 0)
   IF err = 0
      RgDocPrint(0)
   ELSE
     ; Druckertreiber nicht installiert oder Dateiname ungültig
      TxMeldung = RgGetErrorText(err)
      ok = BoxMessage("Fehler!", TxMeldung, "!1")
   END
RgDocClose(0)
END
```

Siehe auch:

RgDocPrint

RgDocRemovePage

Anwendungsbereich: Reportgenerator

Aus dem aktuellen Report wird eine Seite entfernt.

Deklaration:

```
RgDocRemovePage ( Seitennummer ) -> Fehlercode
```

Parameter:

Seitennummer	Gibt die Seitennummer (1..) der zu löschenden Seite an.
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Eine Seite wird aus dem aktuellen Report entfernt.

Ein Report muss wenigstens eine Seite haben, das Löschen in einem einseitigen Report ist also nicht möglich.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Im aktuellen Dokument werden alle Seiten bis auf die ersten 2 gelöscht.

```
LastPage = RgDocGetPageCount()
WHILE LastPage > 2
   RgDocRemovePage(LastPage)
   LastPage = LastPage-1
END
```

Die Reportvorlage "template.drb" enthält 10 Seiten. Abhängig von den zu dokumentierenden Daten werden eventuell nur die ersten 8 Seiten benötigt. In diesem Fall werden die überflüssigen Seiten vor dem Ausdruck entfernt.

```
RgDocOpen("template.drb",0)
ExtendedReport = 0

Page = 1
WHILE Page <= 8
   RgDocSetActivePage(Page)
   ; Füllen der jeweiligen Seite
   ; ...
   ; ggf: ExtendedReport = 1
   Page = Page+1
END

IF ExtendedReport
   RgDocSetActivePage(9)
   ; ... Seite füllen
   RgDocSetActivePage(10)
   ; ... Seite füllen
ELSE
   ; Überflüssige Seiten löschen
   RgDocRemovePage(10)
   RgDocRemovePage(9)
END
RgDocPrint(0)
```

Siehe auch:

RgDocInsertPage

RgDocSave

Anwendungsbereich: Reportgenerator

Der Reportgenerator speichert das aktive Dokument.

Deklaration:

```
RgDocSave ( TxDateiName, Null ) -> Fehlercode
```

Parameter:

TxDateiName	Dateiname, unter dem das aktive Dokument zu speichern ist.
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Reportgenerator speichert das aktive Dokument unter dem angegebenen Dateinamen.

Wenn für den Dateinamen ein leerer Text angegeben wird, wird der Report unter seinem aktuellen Titel abgelegt.

Falls der angegebene Dateiname keine Namenserweiterung besitzt, wird ".drb" angehängt.

Falls kein voller Pfadname angeben ist, wird bei aktivem Projekt das aktuelle Projektverzeichnis verwendet. Ansonsten wird das Verzeichnis benutzt, dass durch den letzten Aufruf von RgSetDir festgelegt worden ist. Falls diese Funktion noch nicht aufgerufen wurde, wird das in FAMOS festgelegte Standardverzeichnis (Verzeichnis für Druckbilddateien) verwendet.

- Solange zu einem Zeitpunkt nur 1 Dokument geöffnet ist, ist auch das aktive Dokument immer eindeutig bestimmt. Sonst muss durch Verwendung der Funktion RgDocSetActive sichergestellt werden, dass auch das gewünschte Dokument angesprochen wird.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.
- Soll ein Dateiname keine Dateinamen-Erweiterung haben, wird der Dateiname mit einem "." abgeschlossen.

Beispiele:

Ein Report wird geladen. Ein enthaltenes Kurvenobjekt wird aktualisiert. Der Report wird unter neuem Namen gespeichert und wieder geschlossen.

```
CwNewWindow(Daten, "show")
err = RgDocOpen("ku_maske", 0)
IF err = 0
   err = RgCurveSet("curve1", Daten, 0)
   IF err = 0
      err = RgDocSave("ku_0001", 0)
   END
   RgDocClose(0)
END
```

Siehe auch:

RgDocOpen, RgDocClose, RgDocSetActive, RgWindow

RgDocSetActive

Anwendungsbereich: Reportgenerator

Der Reportgenerator aktiviert ein geöffnetes Dokument.

Deklaration:

```
RgDocSetActive ( TxTitel, Null ) -> Fehlercode
```

Parameter:

TxTitel	Titel des zu aktivierenden Reportes
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Reportgenerator aktiviert ein vorhandenes Dokument. Die Funktion ist nur sinnvoll, wenn gerade mehrere Dokumente im Reportgenerator geöffnet werden. Die meisten Funktionen wirken auf das jeweils aktive Dokument.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.
- Multithreading: Die Funktionen des Report-Kits dürfen überall aufgerufen werden und wirken global. Der hier aktivierte Report ist also für alle Ausführungs-Threads gültig.

Beispiele:

Während mehrere Dokumente geöffnet sind, wird der Anwender gefragt, ob er das zuletzt geladene drucken möchte. Solange die Abfragebox offen ist, könnte der Anwender am Reportgenerator manuell ein anderes Dokument aktivieren. Nach Ende der Abfrage muss also sichergestellt werden, dass der folgende Druckbefehl tatsächlich auf den gewünschten Report wirkt.

```
err = RgDocOpen("egal", 0)
...
err = RgDocOpen("d:\drb\drb1.drb", 0)
IF err = 0
   ok = BoxMessage("Drb1", "Ducken ?", "?2")
   IF ok = 1
      err = RgDocSetActive("drb1.drb", 0)
      IF err =0
         RgDocPrint(0)
      END
   END
END
```

Siehe auch:

RgDocOpen, RgDocClose, RgWindow

RgDocSetActivePage

Anwendungsbereich: Reportgenerator

Die aktuelle Seite wird festgelegt. Nachfolgende objektspezifische Funktionsaufrufe beziehen sich auf die hier gewählte Seite.

Deklaration:

```
RgDocSetActivePage ( Seitennummer ) -> Fehlercode
```

Parameter:

Seitennummer	Gibt die Seitennummer (1..) der zu aktivierenden Seite an. Eine 0 bedeutet, dass objektspezifische Funktionen auf alle Seiten anzuwenden sind.
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Die hier gewählte Seite wird von einigen Funktionen des Reportgenerator-Kits verwendet, um den Zielbereich der auszuführenden Operation festzulegen. Dies betrifft z.B. alle Funktionen, die Eigenschaften von Objekten ändern oder abfragen. Die Identifikation des anzusprechenden Objektes geschieht dabei über den Objekt-Titel. Wenn mittels RgDocSetActivePage(..) eine spezielle Seite ausgewählt wurde, wird nur auf auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht.

Ohne einen solchen Aufruf (oder nach RgDocSetActive(0)) wird dagegen über alle Seiten des Dokumentes gesucht. Bei Funktionen, die Eigenschaften abfragen, wird dann das erste gefundene Objekt verwendet. Bei Funktionen, die Eigenschaften setzen, wird die Operation ggf. mehrfach auf alle gefundenen Objekte angewendet. Dies ist z.B. praktisch, wenn alle Seiten einen identischen Aufbau haben und ein bestimmtes Element (z.B. Datum in Fusszeile) für alle Seiten auf den gleichen Inhalt gestzt werden soll.

- Die Funktion RgDocInsertPage(..) setzt ebenfalls die aktuelle Seite.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.
- Multithreading: Die Funktionen des Report-Kits dürfen überall aufgerufen werden und wirken global. Der hier selektierte Seite ist also für alle Ausführungs-Threads gültig.

Beispiele:

Die Reportvorlage "template.drb" enthält 3 Seiten. Die erste ist ein festes Titelblatt, die anderen 2 haben den gleichen Aufbau und sollen mit verschiedenen Kurvenbildern befüllt werden. Außerdem haben alle Seiten in der Legende ein Textfeld, welches mit dem aktuellen Datum befüllt wird.

```
RgDocOpen("template.drb",0)

; 2.Seite füllen
CwLoadCCV(data1, "data1.ccv")
RgDocSetActivePage(2)
RgCurveSet("curve1", data1, 0)

; 3. Seite füllen
CwLoadCCV(data2, "data2.ccv")
RgDocSetActivePage(3)
RgCurveSet("curve1", data2, 0)

; Auf allen Seiten Datum eintragen
RgDocSetActivePage(0)
RgTextSet("Date", TimeToText(TimeSystem?(), 0),0)
```

Siehe auch:

RgDocInsertPage

RgGetErrorText

Anwendungsbereich: Reportgenerator

Fehlertext aus Fehlercode ermitteln

Deklaration:

```
RgGetErrorText ( FehlerCode ) -> TxFehlerText
```

Parameter:

FehlerCode	Fehlercode (< 0)
TxFehlerText	Zugehöriger Fehlertext

Beschreibung:

Die Funktion liefert zu einem Fehlercode, der im Fehlerfall von einer Funktion zurückgegeben worden ist, den zugehörigen Fehlertext.

Beispiele:

Es wird versucht, einen Report zu laden. Falls dieses fehlschlägt, wird eine Ausgabebox mit der Fehlermeldung erzeugt.

```
err = RgDocOpen("d:\drb\drb1.drb", 0)
IF err < 0
   Meldung$ = RgGetErrorText(err)
   ok = BoxMessage("Fehler!", Meldung$, "!1")
ELSE
   ...
END
```

RgObjDelete

Anwendungsbereich: Reportgenerator

Ein Reportobjekt wird aus dem aktiven Report entfernt.

Deklaration:

```
RgObjDelete ( Titel ) -> Fehlercode
```

Parameter:

Titel	Titel des Objektes
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das angegebene Objekt wird gelöscht.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

In das Textobjekt "MyText" soll der Inhalt einer Textvariablen übertragen werden. Falls dieser Text leer ist, wird das Textobjekt gelöscht.

```
IF TLeng(TxContents) > 0
   RgTextSet("MyText", TxContents, 0)
ELSE
   RgObjDelete("MyText")
END
```

Siehe auch:

RgObjMove

RgObjFind

Anwendungsbereich: Reportgenerator

Durchsucht den Report nach einem Objekt.

Deklaration:

```
RgObjFind ( TxTitel, Null ) -> Ergebnis
```

Parameter:

TxTitel	Titel des gesuchten Objektes
Null	Reserviert, immer auf 0 zu setzen
Ergebnis	Erfolg der Funktion
	< 0 : Fehlercode.
	= 0 : Objekt nicht gefunden
	> 0 : Objekt mit diesem Titel gefunden.
	1 : Text
	2 : Tabelle
	3 : Kurve
	10 : Linie
	11 : Linie (vertikal)
	12 : Linie (horizontal)
	13 : Polylinie
	20 : Rechteck
	21 : Ellipse
	22 : Polygon
	30 : Bitmap
	31 : Metafile
	40 : OLE-Objekt

Beschreibung:

Es wird ermittelt, ob ein Objekt mit dem angegebenen Titel im aktiven Dokument existiert. Wenn dies der Fall ist, wird zugleich dessen Typ ermittelt.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird die Objektsuche (nach dem gegebenen Titel) auf eben diese Seite beschränkt. Ansonsten wird über alle Seiten des Dokuments gesucht und das erste gefundene Objekt verwendet.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Druckbild wird geladen. Wenn in diesem ein Textobjekt mit dem Titel "Datum" existiert, wird es mit dem aktuellen Datum belegt.

```
err   = RgDocOpen("report1", 0)
suche = RgObjFind ("Datum", 0)
IF suche = 1
   TxDatum$ = TimeToText(TimeSystem?(), 1)
   RgTextSet("Datum", TxDatum$, 0)
END
```

Siehe auch:

RgObjGetTitle, RgObjGetType, RgObjGetCount

RgObjGetCount

Anwendungsbereich: Reportgenerator

Die Anzahl der Objekte im aktiven Dokument wird ermittelt.

Deklaration:

```
RgObjGetCount ( Null ) -> Ergebnis
```

Parameter:

Null	Reserviert, immer auf 0 zu setzen
Ergebnis	
	>= 0 : Anzahl der Objekte
	< 0 : Fehlercode.

Beschreibung:

Die Anzahl der Objekte im aktiven Dokument wird ermittelt. Dabei werden alle Objekte, unabhängig von ihrem Typ, gezählt.

Die Funktion wird i. a. in Verbindung mit RgObjGetTitle oder RgObjGetType verwendet.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird die Anzahl der Objekte eben dieser Seite ermittelt. Ansonsten wird die Anzahl aller Objekte auf allen Seiten zurückgegeben.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

In einem Report wird die Schriftfarbe für alle Textobjekte, die einen Titel besitzen, auf Grün gesetzt.

```
err    = RgDocOpen("report1", 0)
anzahl = RgObjGetCount(0)
index  = 1
WHILE index <= anzahl
   IF RgObjGetType(index) = 1 ; also Text
      TxTitel$ = RgObjGetTitle(index)
      f = RGB(0,255,0)
      err = RgObjSetColor(TxTitel$, 2, f, 0, 0)
   END
   index = index +1
END
```

Siehe auch:

RgObjGetTitle, RgObjFind, RgObjGetType

RgObjGetPos

Anwendungsbereich: Reportgenerator

Die Position eines Reportobjektes wird abgefragt.

Deklaration:

```
RgObjGetPos ( Titel , Position, Option ) -> Position
```

Parameter:

Titel	Titel des Objektes
Position	Positonsparameter
	0 : X-Position der linken oberen Ecke
	1 : Y-Position der linken oberen Ecke
	2 : Breite
	3 : Höhe
Option	Option
	0 : Standard
	1 : Nur für Kurvenobjekte. Die Position des Koordinatensystems wird ermittelt.
Position	Position, Abstand [in Millimeter] von linken bzw. oberen Blattrand bzw. Abmessung in mm.

Beschreibung:

Die Position eines Objektes im aktiven Dokument wird abgefragt.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird die Objektsuche (nach dem gegebenen Titel) auf eben diese Seite beschränkt. Ansonsten wird über alle Seiten des Dokuments gesucht und das erste gefundene Objekt verwendet.

Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Textobjekt mit dem Titel "MyText" wird um 1cm nach rechts verschoben sowie die Höhe verdoppelt.

```
x = RgObjGetPos("MyText", 0, 0)
y = RgObjGetPos("MyText", 1, 0)
RgObjMove("MyText", x+10, y, "", 0)
height = RgObjGetPos("MyText", 3, 0)
RgObjSetSize("MyText", -1, height*2, 0)
```

Siehe auch:

RgObjSetSize, RgObjMove

RgObjGetTitle

Anwendungsbereich: Reportgenerator

Der Titel eines Objektes wird ermittelt.

Deklaration:

```
RgObjGetTitle ( ObjektIndex ) -> TxTitel
```

Parameter:

ObjektIndex	Index des Objektes, 1.. Anzahl der Objekte
TxTitel	Titel des angegebenen Objektes

Beschreibung:

Der Titel eines Objektes im aktiven Dokument wird ermittelt. Die Funktion wird i. a. im Zusammenhang mit RgObjGetCount(..) verwendet.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, bezieht sich der Index auf die Liste der Objekte eben dieser Seite. Ansonsten bezieht sich der Index auf die Liste aller Objekte (über alle Seiten).

Beispiele:

In einem Report wird die Farbe für alle waagerechten Linien, die einen Titel besitzen, auf Rot gesetzt.

```
err    = RgDocOpen("report1", 0)
anzahl = RgObjGetCount(0)
index  = 1
WHILE index <= anzahl
   IF RgObjGetType(index) = 12
      TxTitel$ = RgObjGetTitle(index)
      f = RGB(255,0,0)
      err = RgObjSetColor(TxTitel$, 0, f, 0, 0)
   END
   index = index +1
END
```

Siehe auch:

RgObjGetCount, RgObjFind, RgObjGetType

RgObjGetType

Anwendungsbereich: Reportgenerator

Der Typ eines Objektes wird ermittelt.

Deklaration:

```
RgObjGetType ( ObjektIndex ) -> Typ
```

Parameter:

ObjektIndex	Index des Objektes, 1.. Anzahl der Objekte
Typ	Typ des Objektes
	< 0 : Fehlercode.
	1 : Text
	2 : Tabelle
	3 : Kurve
	10 : Linie
	11 : Linie (vertikal)
	12 : Linie (horizontal)
	13 : Polylinie
	20 : Rechteck
	21 : Ellipse
	22 : Polygon
	30 : Bitmap
	31 : Metafile
	40 : OLE-Objekt

Beschreibung:

Der Typ eines Objektes im aktiven Dokument wird ermittelt. Die Funktion wird i. a. im Zusammenhang mit der Funktion RgObjGetCount verwendet.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, bezieht sich der Index auf die Liste der Objekte eben dieser Seite. Ansonsten bezieht sich der Index auf die Liste aller Objekte (über alle Seiten).

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

In einem Report wird die linke obere Zelle aller vorhandenen Tabellen mit dem aktuellen Datum belegt.

```
TxDatum$ = TimeToText(TimeSystem?(), 1)
err      = RgDocOpen("report1", 0)
anzahl   = RgObjGetCount(0)
index    = 1
WHILE index <= anzahl
   IF RgObjGetType(index) = 2
      TxTitel$ = RgObjGetTitle(index)
      err=RgTableSetCell(TxTitel$,1,1, TxDatum$, 0)
   END
   index = index +1
END
```

Siehe auch:

RgObjGetCount, RgObjFind, RgObjGetType

RgObjMove

Anwendungsbereich: Reportgenerator

Die Position eines Reportobjektes wird geändert.

Deklaration:

```
RgObjMove ( Titel , X-Position, Y-Position, ReferenzObjekt, Null ) -> Fehlercode
```

Parameter:

Titel	Titel des Objektes
X-Position	Neuer Abstand des Objektes vom linken Blattrand in Millimeter.
Y-Position	Neuer Abstand des Objektes vom oberen Blattrand in Millimeter.
ReferenzObjekt	Wenn angegeben, dann gelten beide Koordinaten relativ zur Position (linke obere Ecke) dieses Referenzobjektes.
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Die Position eines Objektes im aktiven Dokument wird geändert.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Textobjekt mit dem Titel "MyText" wird um 1cm nach rechts verschoben sowie die Höhe verdoppelt.

```
x = RgObjGetPos("MyText", 0, 0)
y = RgObjGetPos("MyText", 1, 0)
RgObjMove("MyText", x+10, y, "", 0)
height = RgObjGetPos("MyText", 3, 0)
RgObjSetSize("MyText", -1, height*2, 0)
```

Siehe auch:

RgObjSetSize, RgObjGetPos

RgObjSetColor

Anwendungsbereich: Reportgenerator

Eine Farbe eines Objektes wird geändert.

Deklaration:

```
RgObjSetColor ( Titel , Option, RGBColor, Spalte, Zeile ) -> Fehlercode
```

Parameter:

Titel	Titel des Objektes
Option	Option
	0 : Farbe des Rahmens bzw. der Linie
	1 : Farbe des Hintergrundes
	2 : Farbe der Schrift
RGBColor	Farbe
	>=0 : RGB-Wert der gewünschten Farbe.
	-1 : Transparent
Spalte	Index der Spalte bzw. 0
Zeile	Index der Zeile bzw. 0
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Eine Farbe (Rahmen, Hintergrund oder Text) eines Objektes im aktiven Dokument wird geändert.

Bei Tabellenobjekten bestimmen die Angaben für [Spalte, Zeile] die gewünschte Zelle, links oben ist [1,1]. Wenn beide 0 sind, gilt die Farbänderung für die gesamte Tabelle. Ansonsten wird nur die angegebene Zelle geändert.

Bei allen anderen Objekttypen sind Spalte und Zeile beide auf 0 zu setzen.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

- Der als dritte Parameter übergebene RGB-Wert stellt den Anteil der Intensitäten der 3 Grundfarben Rot, Grün und Blau dar. Um einen solchen Farbwert zu erzeugen, verwenden Sie die FAMOS-Funktion RGB.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Wert wird berechnet und an ein Textobjekt übertragen. Falls ein Grenzwert überschritten ist, wird der Text rot hinterlegt.

```
TxTitel$ = ...
Maxim = max(Daten)
err = RgTextSet(TxTitel$, TForm(Maxim, "f32"), 0)
IF Maxim > 50
   f = RGB(255,0,0)
   err = RgObjSetColor(TxTitel$, 1, f, 0, 0)
END
```

Siehe auch:

RGB

RgObjSetSize

Anwendungsbereich: Reportgenerator

Die Größe eines Reportobjektes wird geändert.

Deklaration:

```
RgObjSetSize ( Titel , Breite, Höhe, Null ) -> Fehlercode
```

Parameter:

Titel	Titel des Objektes
Breite	Neue Breite des Objektes in Millimeter oder -1 für unverändert.
Höhe	Neue Höhe des Objektes in Millimeter oder -1 für unverändert.
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Die Größe eines Objektes im aktiven Dokument wird geändert.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Ein Textobjekt mit dem Titel "MyText" wird um 1cm nach rechts verschoben sowie die Höhe verdoppelt.

```
x = RgObjGetPos("MyText", 0, 0)
y = RgObjGetPos("MyText", 1, 0)
RgObjMove("MyText", x+10, y, "", 0)
height = RgObjGetPos("MyText", 3, 0)
RgObjSetSize("MyText", -1, height*2, 0)
```

Siehe auch:

RgObjMove, RgObjGetPos

RgSetDir

Anwendungsbereich: Reportgenerator

Das Standardverzeichnis wird gesetzt.

Deklaration:

```
RgSetDir ( TxVerzeichnis ) -> Fehlercode
```

Parameter:

TxVerzeichnis	Neues Standardverzeichnis
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das Verzeichnis zum Laden und Speichern von Druckbild-Dateien wird gesetzt. Wenn bei den entsprechenden Befehlen wie z. B. RgDocOpen kein voller Pfadname angegeben wird, wird dieses Verzeichnis verwendet.

Dieser Befehl ist für die Gruppe der kompatiblen Funktionen, wie z. B. DrKonfig, wirkungslos.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Eine Druckbilddatei wird aus einem Verzeichnis geladen und nach erfolgter Aktualisierung in einem anderen Verzeichnis gespeichert.

```
err = RgSetDir("d:\drb\masken")
IF err = 0
   err = RgDocOpen("report1", 0)
END
err = RgCurveSet(...)
...
err = RgSetDir("d:\drb\ergebnis")
err = RgDocSave("report1", 0)
```

Siehe auch:

RgDocOpen, RgDocSave

RgTableColumns?

Anwendungsbereich: Reportgenerator

Ermittelt die Zahl der Spalten einer Tabelle.

Deklaration:

```
RgTableColumns? ( Titel ) -> Ergebnis
```

Parameter:

Titel	Titel der Tabelle
Ergebnis	
	> 0 : Anzahl der Spalten
	< 0 : Fehlercode

Beschreibung:

Die Anzahl der Spalten der angegebenen Tabelle im aktiven Dokument wird ermittelt.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird die Objektsuche (nach dem gegebenen Titel) auf eben diese Seite beschränkt. Ansonsten wird über alle Seiten des Dokuments gesucht und das erste gefundene Objekt verwendet.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Die Spalten einer Tabelle werden mit den einzelnen Events eines eventierten Datensatzes belegt.

```
Anzahl    = RgTableColumns?("Tab1")
AnzahlEvn = EventNum?(Daten)
IF AnzahlEvn < Anzahl
   Anzahl = AnzahlEvn
END
i = 1
WHILE i <= Anzahl
   err = RgTableSetColumn("Tab1",i,2, Daten[i],0)
   i = i + 1
END
```

Siehe auch:

RgTableRows?, RgTableSetCell, RgTableSetColumn, RgTableSetRow

RgTableGetCellText

Anwendungsbereich: Reportgenerator

Fragt den Inhalt einer Tabellenzelle ab.

Deklaration:

```
RgTableGetCellText ( Titel, Spalte, Zeile, Null ) -> TxInhalt
```

Parameter:

Titel	Titel der Tabelle
Spalte	Spalte (1..)
Zeile	Zeile (1..)
Null	Reserviert, immer auf 0 zu setzen
TxInhalt	Inhalt der angegebenen Tabellenzelle

Beschreibung:

Der Inhalt einer Zelle der angegebenen Tabelle im aktiven Dokument wird abgefragt.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird die Objektsuche (nach dem gegebenen Titel) auf eben diese Seite beschränkt. Ansonsten wird über alle Seiten des Dokuments gesucht und das erste gefundene Objekt verwendet.

- Die maximale Länge des Rückgabetextes beträgt 255 Zeichen.

Beispiele:

Der Inhalt einer Tabellenzelle (links oben) wird abgefragt. In dieser ist als Platzhalter ein "xxx" für den Namen abgelegt. Dieser Platzhalter wird ersetzt und das Tabellenzelle aktualisiert.

Die Angaben fuer Zeile und Spalte bestimmen die Position der abzufragenden Zelle, links oben ist [1,1]

```
Tx$= RgTableGetCellText("Tab1", 1, 1, 0)
Tx$= TReplace(Tx$,"xxx","Heinz Muster")
err = RgTableSetCell("Tab1", 1, 1, Tx$, 0)
```

Siehe auch:

RgTableSetColumn, RgTableSetRow, RgTableSetCell

RgTableRows?

Anwendungsbereich: Reportgenerator

Ermittelt die Zahl der Zeilen einer Tabelle.

Deklaration:

```
RgTableRows? ( TxTitel ) -> Ergebnis
```

Parameter:

TxTitel	Titel der Tabelle
Ergebnis	
	> 0 : Anzahl der Zeilen
	< 0 : Fehlercode

Beschreibung:

Die Anzahl der Zeilen der angegebenen Tabelle im aktiven Dokument wird ermittelt.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird die Objektsuche (nach dem gegebenen Titel) auf eben diese Seite beschränkt. Ansonsten wird über alle Seiten des Dokuments gesucht und das erste gefundene Objekt verwendet.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Die Zeilen einer Tabelle werden mit den einzelnen Segmenten eines segmentierten Datensatzes belegt.

```
Anzahl    = RgTableRows?("Tab1")
AnzahlSeg = leng?(Daten) / Seglen?(Daten)
IF AnzahlSeg < Anzahl
   Anzahl = AnzahlSeg
END
i = 1
WHILE i <= Anzahl
   err = RgTableSetRow("Tab1",1,i, Daten[i],0)
   i = i + 1
END
```

Siehe auch:

RgTableColumns?, RgTableSetCell, RgTableSetColumn, RgTableSetRow

RgTableSetCell

Anwendungsbereich: Reportgenerator

Setzt den Inhalt einer Tabellenzelle.

Deklaration:

```
RgTableSetCell ( TxTitel, Spalte, Zeile, Inhalt, Null ) -> Fehlercode
```

Parameter:

TxTitel	Titel der Tabelle
Spalte	Spalte (1..)
Zeile	Zeile (1..)
Inhalt	Zu übertragende Zahl oder Text
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Eine Zelle der angegebenen Tabelle im aktiven Dokument wird mit einem Text oder einzelnen Wert belegt. Wenn eine Zahl übertragen wird, gilt die in der Tabelle festgelegte Formatierung.

Die Angaben fuer Zeile und Spalte bestimmen die Position der zu beschreibenden Zelle, links oben ist [1,1]

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

- Wenn bei <XXInhalt> ein Datensatz angegeben wird, der eine Länge größer 1 besitzt, wird der erste Wert des Datensatzes verwendet.
- Wenn ein Zahlenwert übertragen wird, wird dieser entsprechend der festgelegten Eigenschaften für diese Tabelle formatiert (Zahlenformat, Zahl der Nachkommastellen etc.). Diese Attribute werden im Eigenschaften-Dialog für Tabellenobjekte, Karte "Zahlen", eingestellt.
- Wenn Sie diese Voreinstellung nicht verwenden möchten, können Sie die Zahl vorher selbst mit der Funktion TForm in einen Text der gewünschten Form wandeln und diesen dann übertragen.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Die zweite Spalte einer Tabelle wird mit einem gerade berechneten Datensatz belegt. Die erste Zeile enthält den Variablennamen, die zweite Zeile das Maximum des Datensatzes in einem festen Format. Ab der 3. Zeile beginnen die Zahlenwerte, die so dargestellt werden, wie es in der Tabelle festgelegt ist.

```
Kanal1= ...
err = RgTableSetCell("Tab1", 2, 1, "Kanal1",0)
TxMax$ = TForm(Max(Kanal1), "f32")
err = RgTableSetCell("Tab1", 2, 2, TxMax$, 0)
err = RgTableSetColumn("Tab1", 2, 3, Kanal1, 0)
```

Siehe auch:

RgTableSetColumn, RgTableSetRow, RgTableGetCellText

RgTableSetColumn

Anwendungsbereich: Reportgenerator

Setzt den Inhalt einer Tabellenspalte.

Deklaration:

```
RgTableSetColumn ( TxTitel, Spalte, Zeile, Inhalt, Null ) -> Fehlercode
```

Parameter:

TxTitel	Titel der Tabelle
Spalte	Spalte (1..)
Zeile	Zeile (1..)
Inhalt	Zu übertragende Daten.
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Eine Spalte der angegebenen Tabelle im aktiven Dokument wird mit einem Datensatz belegt. Die einzelnen Zahlen werden entsprechend der in der Tabelle festgelegten Formatierung eingetragen. Die erste Zahl wird an der angegebenen Position eingetragen, alle weiteren darunter.

Die Angaben fuer Zeile und Spalte bestimmen die Position der ersten zu beschreibenden Zelle, links oben ist [1,1]

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

- Es werden so viele Zahlen übertragen, bis entweder der letzte Wert des Datensatzes oder die letzte Tabellenzeile erreicht ist.
- Die Zahlenwerte werden entsprechend der festgelegten Eigenschaften für diese Tabelle formatiert (Zahlenformat, Zahl der Nachkommastellen etc.). Diese Attribute werden im Eigenschaften-Dialog für Tabellenobjekte, Karte "Zahlen", eingestellt.
- Zur Übertragung von Text in eine Tabellenzelle verwenden Sie die Funktion RgTableSetCell.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Die Spalten einer Tabelle werden mit den einzelnen Events eines eventierten Datensatzes belegt.

```
Anzahl    = RgTableColumns?("Tab1")
AnzahlEvn = EventNum?(Daten)
IF AnzahlEvn < Anzahl
   Anzahl = AnzahlEvn
END
i = 1
WHILE i <= Anzahl
   err = RgTableSetColumn("Tab1",i,2, Daten[i],0)
   i = i + 1
END
```

Siehe auch:

RgTableSetRow, RgTableSetCell

RgTableSetRow

Anwendungsbereich: Reportgenerator

Setzt den Inhalt einer Tabellenzeile.

Deklaration:

```
RgTableSetRow ( TxTitel, Spalte, Zeile, Daten, Null ) -> Fehlercode
```

Parameter:

TxTitel	Titel der Tabelle
Spalte	Spalte (1..)
Zeile	Zeile (1..)
Daten	Zu übertragende Daten.
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Eine Zeile der angegebenen Tabelle im aktiven Dokument wird mit einem Datensatz belegt. Die einzelnen Zahlen werden entsprechend der in der Tabelle festgelegten Formatierung eingetragen. Die erste Zahl wird an der angegebenen Position eingetragen, alle weiteren rechts davon.

Die Angaben fuer Zeile und Spalte bestimmen die Position der ersten zu setzenden Zelle, links oben ist [1,1]

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

- Es werden so viele Zahlen übertragen, bis entweder der letzte Wert des Datensatzes oder die letzte Tabellenspalte erreicht ist.
- Die Zahlenwerte werden entsprechend der festgelegten Eigenschaften für diese Tabelle formatiert (Zahlenformat, Zahl der Nachkommastellen etc.). Diese Attribute werden im Eigenschaften-Dialog für Tabellenobjekte, Karte "Zahlen", eingestellt
- Zur Übertragung von Text in eine Tabellenzelle verwenden Sie die Funktion RgTableSetCell.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Die Zeilen einer Tabelle werden mit den einzelnen Segmenten eines segmentierten Datensatzes belegt.

```
Anzahl    = RgTableRows?("Tab1")
AnzahlSeg = leng?(Daten) / Seglen?(Daten)
IF AnzahlSeg < Anzahl
   Anzahl = AnzahlSeg
END
i = 1
WHILE i <= Anzahl
   err = RgTableSetRow("Tab1",1,i, Daten[i],0)
   i = i + 1
END
```

Siehe auch:

RgTableSetColumn, RgTableSetCell

RgTextGet

Anwendungsbereich: Reportgenerator

Fragt den Inhalt eines Textobjektes ab.

Deklaration:

```
RgTextGet ( TxTitel, Null ) -> TxInhalt
```

Parameter:

TxTitel	Titel des Objektes
Null	Reserviert, immer auf 0 zu setzen
TxInhalt	Inhalt des Objektes

Beschreibung:

Der Inhalt eines Textobjektes im aktiven Dokument wird abgefragt.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird die Objektsuche (nach dem gegebenen Titel) auf eben diese Seite beschränkt. Ansonsten wird über alle Seiten des Dokuments gesucht und das erste gefundene Objekt verwendet.

- Die maximale Länge des Rückgabetextes beträgt 255 Zeichen.

Beispiele:

Der Inhalt eines Textobjektes wird abgefragt. In diesem ist als Platzhalter ein "xxx" für den Namen abgelegt. Dieser Platzhalter wird ersetzt und das Textobjekt aktualisiert.

```
Tx$= RgTextGet("Unterschrift", 0)
Tx$= TReplace(Tx$,"xxx","Heinz Muster")
err = RgTextSet("Unterschrift", Tx$, 0)
```

Siehe auch:

RgTextSetData, RgTextSet

RgTextSet

Anwendungsbereich: Reportgenerator

Setzt den Inhalt eines Textobjektes.

Deklaration:

```
RgTextSet ( TxTitel, TxInhalt, Null ) -> Fehlercode
```

Parameter:

TxTitel	Titel des Objektes
TxInhalt	Neuer Inhalt des Objektes
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Setzt den Inhalt eines Textobjektes. Der aktuelle Inhalt wird komplett mit dem angegebenen Text überschrieben.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Der Inhalt eines Textobjektes wird abgefragt. In diesem ist als Platzhalter ein "xxx" für den Namen abgelegt. Dieser Platzhalter wird ersetzt und das Textobjekt aktualisiert.

```
Tx$= RgTextGet("Unterschrift", 0)
Tx$= TReplace(Tx$,"xxx","Heinz Muster")
err = RgTextSet("Unterschrift", Tx$, 0)
```

Siehe auch:

RgTextSetData, RgTextGet

RgTextSetData

Anwendungsbereich: Reportgenerator

Ersetzt Platzhalter im Textobjekt.

Deklaration:

```
RgTextSetData ( TxTitel, Daten, Null ) -> Fehlercode
```

Parameter:

TxTitel	Titel des Objektes
Daten	Zu übetragende Daten (Datensatz, Einzelwert oder Text)
Null	Reserviert, immer auf 0 zu setzen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Die Daten werden an das angegebene Textobjekt im aktiven Dokument übertragen. Die im Objekt definierten Platzhalter werden entsprechend ersetzt.

Folgende Platzhalter sind für Text-Objekte definiert:

#d	Aktuelles Datum im Windows-Format
#z	Aktuelle Uhrzeit im Windows-Format
#u	Einheit eines Einzelwertes bzw. y-Einheit eines Datensatzes
#e?	Zahl im Gleitkomma-Format, Angabe der Nachkomma-Stellen
#f?.?	Zahl im Festkomma-Format, Angabe der Vor- und Nachkommastellen
#s	Text

Dabei stehen "?" statt Ziffern.

Mehrseitige Reporte: Eine vorher mittels RgDocSetActivePage(..) bzw. RgDocInsertPage(..) ausgewählte aktuelle Seite wird berücksichtigt. Wenn eine Seite entsprechend vorgewählt wurde, wird nur auf dieser Seite nach einem Objekt mit dem gegebenen Titel gesucht. Ansonsten wird die Objektsuche über alle Seiten des Dokuments durchgeführt und die Operation ggf. mehrfach für alle passenden Objekte ausgeführt.

Hinweis:

Das Platzhalter-Konzept für Textobjekte hatte bei früheren Versionen des Reportgenerators große Bedeutung wegen der eingeschränkten Möglichkeiten zur Fernsteuerung von Druckbildern und des Fehlens eines Tabellenobjektes.

Bei neu zu erstellenden Reporten und Programmen bzw. Sequenzen ist es im allgemeinen flexibler, den gewünschten Text komplett im Programm zu formatieren und dann den gesamten Inhalt des Textobjektes zu ersetzen (Funktion RgTextSet). Es ist auch möglich, den Inhalt eines Textobjektes abzufragen und nach geeigneter Bearbeitung wieder zu setzen (Funktion RgTextGet).

Zur Anzeige von Datensätzen in Tabellenform sollte immer ein Tabellenobjekt verwendet werden.

- Eine genaue Beschreibung der Platzhalter finden Sie im Kapitel "Reportgenerator" bei der Beschreibung der Textobjekte.
- Die Platzhalter "#d" (aktuelles Datum) und "#z" (aktuelle Uhrzeit) werden beim ersten Zugriff auf ein Text-Objekt mit der Funktion "RgTextSetData" ersetzt.
- Wenn Datensätze an Text-Objekte mit Platzhaltern für reelle Zahlen übertragen werden, so wird für jedes Sample im Datensatz (jeden y-Wert) ein Platzhalter ersetzt. Sie können auf diese Weise eine ganze Tabelle (bzw. eine Spalte oder Zeile) mit einem einzigen Befehl übertragen.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Einmal ersetzte Platzhalter in Text-Objekten bleiben für immer ersetzt. Wenn Sie einen anderen Report erstellen möchten, müssen Sie zuerst wieder die Maske mit der Funktion RgDocOpen laden!

Beispiele:

Zuerst wird eine Maske geladen.

```
TxFehler$ = RgDocOpen("report", 0)
```

Im Druckbild (in dieser Maske) sind 2 Text-Objekte definiert:

Titel	Inhalt

Name	#s
Zahl	x: #e2 und #f2.1

Die folgenden Zeilen werden ausgeführt:

```
RgTextSetData("Name", "Herbert", 0)
RgTextSetData("Zahl", 27.5, 0)
RgTextSetData("Zahl", 28.9, 0)
```

Danach enthalten die Text-Objekte folgende Texte:

Titel	Inhalt
Name	Herbert
Zahl	x: 2.75E+01 und 28.9

Siehe auch:

RgTextSet, RgTextGet, RgCurveSet, RgTableSetCell, RgTableSetRow, RgTableSetColumn

RgWindow

Anwendungsbereich: Reportgenerator

Startet oder schließt den Reportgenerator

Deklaration:

```
RgWindow ( Auftrag ) -> Fehlercode
```

Parameter:

Auftrag	Auftrag
	1 : Reportgenerator als normales Fenster starten
	2 : Reportgenerator als Sinnbild starten
	3 : Reportgenerator schließen
Fehlercode	Erfolg der Funktion
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Mit dieser Funktion wird der Reportgenerator gestartet, beendet oder die Fenstergröße zwischen Normalanzeige und Symbolanzeige umgeschaltet.

- Beim Beenden des Reportgenerators gehen alle noch nicht gespeicherten Änderungen in geöffneten Dokumenten verloren.
- Im Fehlerfall (Rückgabewert < 0) kann der zugehörige Fehlertext mit der Funktion RgGetErrorText erfragt werden.

Beispiele:

Eine Druckbildvorlage wird geladen. Wenn der Reportgenerator noch nicht offen war, wird er dabei gestartet und als Symbol angezeigt. Es werden diverse Übertragungen an das Druckbild durchgeführt. Anschließend wird das fertige Druckbild in Normalgröße angezeigt, damit der Anwender entscheiden kann, ob der Report gedruckt werden soll. Zum Schluß wird der Reportgenerator geschlossen.

```
err = RgDocOpen("d:\usr\report1", 0)
IF err = 0
   ...
   ;; Diverse Aktualisierungen
   ...
   err = RgWindow(1)
   ok =  BoxMessage("Frage", "Drucken?" "?2")
   IF ok
      RgDocPrint(0)
   END
   RgWindow(3)
END
```

Siehe auch:

RgDocOpen, RgDocClose

RMS

Der Effektivwert (quadratisches Mittel) der Zahlenwerte eines Datensatzes wird ermittelt.
Alternativer Name: **Eff**

Deklaration:

```
RMS ( Daten ) -> EwEffektivwert
```

Parameter:

Daten	Daten, von denen der Effektivwert berechnet werden soll [ND].
EwEffektivwert	Effektivwert des Datensatzes

Beschreibung:

Es wird die Wurzel aus dem Mittelwert der Quadrate aller Werte eines Datensatzes berechnet. In der Elektrotechnik wird dieser Kennwert auch als Effektivwert bezeichnet. Der Effektivwert ist definiert als die Wurzel aus dem mittleren Quadrat, das seinerseits die Summe der Quadrate aller Werte ist, bezogen auf die Anzahl der Werte.

Der Effektivwert ist nie kleiner Null. Im Gegensatz zum arithmetischen Mittelwert gehen positive und negative Werte gleich ein, löschen sich also nicht aus.

Beispiele:

Der Effektivwert eines Spannungsverlaufs wird berechnet:

```
RMS_voltage = RMS(voltage)
```

Siehe auch:

Mean, StDev, MvRMS, ExpoRMS, Stat

Rosette

Verfügbar ab: Professional Edition

Bei Rosetten werden aus den gemessenen Dehnungen die Hauptdehnung und die Hauptspannung ermittelt.

Deklaration:

```
Rosette ( GitterA, GitterB, GitterC, Rosettenform, Berechnung [, Querdehnungszahl] [, Elastizitätsmodul] [, Minimale Dehnung] ) -> Ergebnis
```

Parameter:

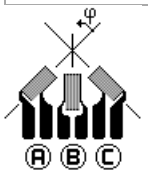
GitterA	Gitter A
GitterB	Gitter B
GitterC	Gitter C
Rosettenform	Welche Rosettenform?
	"0-45-90" : 0-45-90 Grad Rechteck Rosette
	"0-60-120" : 0-60-120 Grad Delta Rosette
Berechnung	Was wird berechnet?
	"eps1" : Hauptdehnung 1, eps1
	"eps2" : Hauptdehnung 2, eps2
	"sig1" : Hauptspannung 1, sig1
	"sig2" : Hauptspannung 2, sig2
	"sigV" : Vergleichsspannung nach Mises, sigV
	"phi" : Orientierungswinkel phi, Hauptrichtung 1 gemessen gegen Gitter A
Querdehnungszahl	Querkontraktionszahl, Querdehnungszahl, Querdehnzahl, Poissonzahl, z.B. bei Metallen im Bereich 0.3 bis 0.4 (optional , Standardwert: 0)
Elastizitätsmodul	Elastizitätsmodul in GPa, z.B. bei Stahl 210 GPa (optional , Standardwert: 0)
Minimale Dehnung	Grenze für die Dehnungen, aus denen durch arc tan Berechnung der Winkel ermittelt wird. Sind die Werte <= dieser Grenze, wird der Winkel auf null gesetzt. (optional , Standardwert: 0)
Ergebnis	Ergebnis

Beschreibung:

Nummerierung der Messgitter: Um bei der Messung mit 3-Element-Rosetten korrekte Werte zu erzielen, müssen die Messgitter in einer ganz bestimmten Weise nummeriert werden.

Es ist belanglos, ob die Rosette quadratisch ist oder eine Kantenrosette oder eine runde Rosette. Die Winkel der Dehnmessstreifen zueinander legen die Berechnung fest.

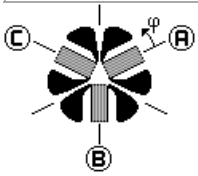
0-45-90 Grad Rechteck-Rosette



Hauptdehnung

$$\varepsilon_{1,2} = \frac{\varepsilon_a + \varepsilon_c}{2} \pm \frac{1}{\sqrt{2}}\sqrt{(\varepsilon_a - \varepsilon_b)^2 + (\varepsilon_c - \varepsilon_b)^2}$$

0-60-120 Grad Delta-Rosette



Hauptdehnung

$$\varepsilon_{1,2} = \frac{\varepsilon_a + \varepsilon_b + \varepsilon_c}{3} \pm \sqrt{\left(\frac{2\varepsilon_a - \varepsilon_b - \varepsilon_c}{3}\right)^2 + \frac{1}{3}(\varepsilon_b - \varepsilon_c)^2}$$

Hauptspannung

$$\sigma_1 = \frac{E}{1-\nu^2}(\varepsilon_1 + \nu\varepsilon_2) \qquad \sigma_2 = \frac{E}{1-\nu^2}(\varepsilon_2 + \nu\varepsilon_1)$$

Vergleichsspannung nach Mises

$$\sigma_v^{\,2} = \sigma_1^{\,2} + \sigma_2^{\,2} - \sigma_1\sigma_2$$

Einheiten

Die Eingangskanäle für die Gitter A, B und C sind i. Allg. in der Einheit micrometer/Meter bzw. microstrain bzw. 1e-6 angegeben. Die Minimale Dehnung für die Winkelbereinigung muss in derselben Einheit angegeben werden. Die Hauptdehnungen eps1 und eps2 werden dann auch in derselben Einheit ermittelt.

Die Hauptspannung und die Vergleichsspannung nach Mises werden in N/mm^2 ermittelt. Das setzt voraus, dass die Eingangskanäle in micrometer/Meter angegeben sind. Außerdem muss der Elastizitätsmodul in GPa vorgegeben sein. Falls sie in anderer Einheit angegeben sind, muss der Anwender nachträglich die Einheit des Ergebnisses korrigieren.

Der Winkel wird in Grad ermittelt.

Für die ermittelten Hauptdehnungen gilt eps1 >= eps2, d.h. die Hauptdehnung 1 ist die größere der beiden Hauptdehnungen.

Das Plus-Zeichen in der Formel führt zu eps1, das Minus-Zeichen zu eps2.

Das Plus-Zeichen in der Formel für die Hauptdehnung führt bei positivem Mittelpunkt der Verformungskreises zu betragsmäßig größeren Werten, das Minus-Zeichen bei negativem Mittelpunkt.

Winkel, Winkelbereinigung

Der gefundene Winkel phi zählt ab Gitter A entgegen dem Uhrzeigersinn. Er ist die Hauptrichtung 1. Er wird im Bereich -90 bis +90 Grad bestimmt.

Winkelmessungen in Richtung des entgegengesetzten Uhrzeigersinns werden also mit positivem Vorzeichen versehen, in Richtung des Uhrzeigersinns mit negativem Vorzeichen.

Hauptrichtung 2 steht immer senkrecht auf Hauptrichtung 1 und kann im Anschluss durch Addition von 90 Grad ermittelt werden.

Winkelbereinigung: Bei der Bestimmung des Winkels wird der arc tan eines Bruches bestimmt. Wenn Zähler und Nenner dieses Bruches sehr klein sind, ist der Winkel i. Allg. sehr ungenau. Wenn die Summe der Absolutbeträge von Zähler und Nenner kleiner gleich der angegebenen "Minimalen Dehnung" sind, wird deshalb der Winkel auf null gesetzt.

Anschaulich kann man auch sagen, dass bei einer Dehnung von null natürlich auch kein Winkel bestimmt werden kann. So sorgt bei Rauschen um null die Funktion dafür, dass der Winkel nicht Zufallswerte annimmt, sondern ruhig bei null liegt.

Bei Zweifel wird der Parameter "Minimale Dehnung" gar nicht angegeben und erhält seinen Default-Wert.

Allgemeines

Alle 3 Eingangskanäle (GitterA, GitterB, GitterC) müssen dieselbe Zeitbasis und selbe Struktur bezüglich Länge, Segmenten und Events aufweisen.

Die Funktion erwartet äquidistante Eingangsdaten. Fall XY-Daten vorliegen, kann man die Funktion auf die .Y-Komponente anwenden.

Querdehnungszahl und Elastizitätsmodul werden nur zur Berechnung der Spannungen benötigt und können ansonsten auf 0 gesetzt sein.

Beispiele:

```
eps1 = Rosette ( epsA, epsB, epsC, "0-45-90", "eps1")
eps2 = Rosette ( epsA, epsB, epsC, "0-45-90", "eps2")
phi = Rosette ( epsA, epsB, epsC, "0-45-90", "phi", 0, 0, 0.001)
poisson = 0.3
emod = 210 ; GPa
sig1 = Rosette ( epsA, epsB, epsC, "0-45-90", "sig1", poisson, emod)
sig2 = Rosette ( epsA, epsB, epsC, "0-45-90", "sig2", poisson, emod)
sigV = Rosette ( epsA, epsB, epsC, "0-45-90", "sigV", poisson, emod)
```

Round

Die Y-Werte eines Datensatzes werden auf die angegebene Genauigkeit gerundet.

Deklaration:

```
Round ( Daten, EwFaktor ) -> Gerundet
```

Parameter:

Daten	Zu rundender Datensatz. Erlaubte Typen: [ND],[XY].
EwFaktor	Die Werte werden auf ganzzahlige Vielfache dieses Wertes gerundet.
Gerundet	Der gerundete Datensatz.

Beschreibung:

Die Y-Werte werden so gerundet, dass sich im Ergebnis nur ganzzahlige Vielfache des angegebenen Faktors befinden. Mit dem Faktor 0.01 runden Sie beispielsweise auf zwei Nachkommastellen, mit dem Faktor 0.25 erhalten Sie Werte wie 0, 0.25, 0.5, 0.75...

Wenn sich ein Wert genau in der Mitte zwischen 2 Rundungswerten befindet, wird zur betragshöheren Zahl hin gerundet.

```
Ergebnis = Round(0.5, 1)  ; Ergebnis:  1
Ergebnis = Round(-0.5, 1)  ; Ergebnis: -1
```

Da reelle Zahlen intern meist nicht exakt darstellbar sind, erhalten Sie unter Umständen unerwartete Ergebnisse, wenn der angezeigte Wert nicht exakt dem dem intern gespeicherten Wert entspricht. Beispielsweise, die Zahl 0.3 kann intern als 0.299999... gespeichert sein, das Ergebnis von Round(0.3, 0.2) wäre damit 0.2 statt wie erwartet (aufgerundet) 0.4.

Beispiele:

Runden auf ganze Zahlen:

```
Ergebnis = Round(2.156, 1)  ; Ergebnis: 2.0
```

Runden auf 2 Dezimalstellen:

```
Ergebnis = Round(2.156, 0.01)  ; Ergebnis: 2.16
```

Runden auf Vielfache von 0.25:

```
Ergebnis = Round(2.156, 0.25)  ; Ergebnis: 2.25
```

Runden auf Vielfache von 100:

```
Ergebnis = Round(51, 100)  ; Ergebnis: 100
```

Siehe auch:

Floor

RSamp

Prototyp-Nachabtasten, Abtasten eines Datensatzes zu den Abtastzeiten einer Referenz mit linearer Interpolation.
Alternativer Name: **PTast**

Deklaration:

```
RSamp ( Daten, Referenz ) -> Ergebnis
```

Parameter:

Daten	Abzutastender Datensatz. Erlaubte Datentypen: [ND],[XY]
Referenz	Referenz, Prototyp für Nachabtastung
Ergebnis	Ergebnis der Nachabtastung.

Beschreibung:

Der erste übergebene Datensatz wird außerhalb seines x-Bereichs mit seinem Anfangs- und Endwert konstant fortgesetzt gedacht. Z. B. für alle x-Koordinaten kleiner als der x-Offset wird der Anfangswert des Datensatzes als y-Koordinate angenommen. Es liegt also eine konstante Extrapolation vor. Der übergebene Datensatz wird ferner innerhalb seines x-Bereichs linear interpoliert gedacht.

Diese nun für alle x-Koordinaten definierte Funktion wird zu den Zeiten (x-Koordinaten) abgetastet, die der Referenzdatensatz (2. Parameter) vorgibt. Damit wird erreicht, dass beide Datensätze synchron abgetastet vorliegen. Der erzeugte Datensatz hat dieselbe Abtastzeit und denselben x-Offset wie der Referenzdatensatz.

Bei XY-Daten muss die X-Spur monoton wachsend sein.

Die Funktion RSamp() ist immer dann sinnvoll einzusetzen, wenn zwei Datensätze mit verschiedenen Abtastzeiten vorliegen und in irgendeiner Weise zu vergleichen oder verrechnen sind. Datensätze können z.B. nur sinnvoll addiert werden (siehe Operator + (Addition)), wenn ihre Abtastzeiten übereinstimmen.

- Die x-Einheiten beider Parameter sollten gleich sein.
- Die Anwendung der Funktion RSamp() ist nur sinnvoll in dem x-Bereich, der beiden Parametern gemeinsam ist. Benutzen Sie ggf. die Funktion Cut(), um den x-Bereich des Referenzdatensatzes einzuschränken.
- Wenn die Funktion RSamp die Datenmenge reduziert, können wie bei allen Abtastfunktionen Aliasing-Effekte auftreten. Benutzen Sie dann Glättungsfunktionen, wie Smo(), bevor Sie abtasten.
- Wenn Sie zwei Datensätze mit verschiedener Abtastzeit miteinander kombinieren möchten, kann die Funktion RSamp() prinzipiell auf jeden der beiden Datensätze angewendet werden. Im Hinblick auf Genauigkeit und Aliasing ist es meistens vorzuziehen, den langsam abgetasteten Datensatz nachträglich mit RSamp() abzutasten. Wählen Sie den schneller abgetasteten Datensatz nur, wenn die Datenmengen sonst zu groß werden. Schalten Sie dann Aliasing-Effekte aus, indem Sie vorher glätten.
- Wenn die Abtastzeiten von zwei Datensätzen ein ganzzahliges Vielfaches voneinander sind, können alternativ zur Funktion RSamp auch die Funktionen Red() oder Lip() und IPol() benutzt werden.

Wenn die lineare Interpolation, die der Funktion RSamp() zugrunde liegt, unzureichend ist (z.B. zu kantig), kann der nun abzutastende Datensatz vorher mit einer Spline-Interpolation mittels IPol() bearbeitet werden, zum Beispiel:

```
NDbetter = RSamp(IPol(NDdata, 10), NDref)
```

Ein Faktor von 10 ist angemessen, wenn die Abtastzeit des Referenzdatensatzes etwa 10mal so hoch ist.

Beispiele:

Der Momentanleistungsverlauf am 50Hz-Netz ist zu ermitteln. Dazu wurde die Netzspannung 1 Sekunde lang mit einer Abtastzeit von 0.2ms abgetastet. Der Strom ist wesentlich hochfrequenter, da ein Stromrichter angeschlossen ist. Deshalb wurde der Strom mit einer Abtastzeit von 0.005 ms über dem selben Zeitraum abgetastet. Die Momentanleistung ist das Produkt aus Strom und Spannung.

```
NDpower = NDcurrent * RSamp(NDvoltage, NDcurrent)
```

Ein Datensatz mit der unüblichen Abtastzeit von 0.15s soll ab der Zeit 2s mit einer Abtastzeit von 0.1s 100mal abgetastet werden:

```
NDnew = RSamp(NDdata, Ramp(2, 0.1, 100))
```

Siehe auch:

RSamp0, RSampEx, RedEx, Red, Red2, IPol, IPli

RSamp0

Prototyp-Nachabtasten, Abtasten eines Datensatzes zu den Abtastzeiten einer Referenz mit konstanter Interpolation.
Alternativer Name: **PTast0**

Deklaration:

```
RSamp0 ( Daten, Referenz ) -> Ergebnis
```

Parameter:

Daten	Abzutastender Datensatz. Erlaubte Datentypen: [ND],[XY]
Referenz	Referenz, Prototyp für Nachabtastung
Ergebnis	Ergebnis der Nachabtastung.

Beschreibung:

Der als erster Parameter übergebene Datensatz wird zu den Zeiten (x-Koordinaten) nachabgetastet, die der Referenzdatensatz (2. Parameter) vorgibt.

Als Ergebniswert für eine bestimmte x-Koordinate der Referenz wird dabei derjenige Wert des Eingangsdatensatzes verwendet, dessen x-Koordinate am nächsten zur Referenz-Koordinate liegt.

Es wird also **im Unterschied zur Funktion RSamp() nicht linear interpoliert**, der Ergebnisdatensatz besteht nur aus Werten, die auch im Eingangsdatensatz vorhanden sind.

Bei XY-Daten muss die X-Spur monoton wachsend sein.

Die Funktion RSamp0() wird dann sinnvoll verwendet, wenn das Eingangssignal aus vordefinierten diskreten Werten gebildet wird. Dies ist z.B. bei digitalen Daten der Fall oder auch bei Messdaten, die Ihrer Bedeutung nach nur ganzzahlig sein können (z.B. die aktuelle Getriebestufe bei Antrieben).

Durch die Nachabtastung wird erreicht, dass beide Datensätze synchron abgetastet vorliegen. Der erzeugte Datensatz hat dieselbe Abtastzeit und denselben x-Offset wie der Referenzdatensatz.

Das Ergebnis besitzt das gleiche Datenformat wie der abgetastete Datensatz.

Der erste übergebene Datensatz wird außerhalb seines x-Bereichs mit seinem Anfangs- und Endwert konstant fortgesetzt gedacht. Z. B. für alle x-Koordinaten kleiner als der x-Offset wird der Anfangswert des Datensatzes als y-Koordinate angenommen (konstante Extrapolation).

Beispiele:

An einem Fahrzeug werden Geschwindigkeit <velocity> und Getriebestufe (<gear>, enthält nur die Werte 1 - 5) gemessen. Es sollen alle Zeitpunkte ermittelt werden, an denen im 3. Gang die Geschwindigkeit kleiner als 50 km/h ist. Da beide Kanäle mit unterschiedlichen Abtastzeiten aufgenommen worden sind, müssen vor der Verrechnung beide Datensätze auf die gleiche Abtastzeit gebracht werden.

```
gear_resampled = RSamp0(gear, velocity)
result = (gear_resampled = 3) AND (velocity < 50)
```

Siehe auch:

RSamp, RSampEx, Red, RedEx, Red2

RSampEx

Prototyp-Nachabtasten, Abtasten eines Datensatzes zu den Abtastzeiten einer Referenz mit wählbarer Interpolation.
Alternativer Name: **PTastEx**

Deklaration:

```
RSampEx ( Daten, Referenz, EwInterpolation, EwOption ) -> Ergebnis
```

Parameter:

Daten	Abzutastender Datensatz. Erlaubt sind die Datentypen ND (äquidistant abgetastet) und XY.
Referenz	Referenz, Prototyp für Nachabtastung
EwInterpolation	Interpolationsart
	0 : Linear. Der Eingangsdatensatz wird linear interpoliert, um die Werte an den Referenzpunkten zu bestimmen. Analoges Verhalten wie bei der Funktion RSamp().
	1 : Konstant, davorliegender Wert. Der Eingangsdatensatz wird konstant interpoliert, d.h. jeder Wert wird konstant gehalten, bis der nächste Wert gültig wird. Als Ergebniswert für eine bestimmte x-Koordinate der Referenz wird also derjenige Wert des Eingangsdatensatzes verwendet, dessen x-Koordinate unmittelbar VOR der Referenz-Koordinate liegt.
	2 : Konstant, nächstliegender Wert. Als Ergebniswert für eine bestimmte x-Koordinate der Referenz wird derjenige Wert des Eingangsdatensatzes verwendet, dessen x-Koordinate am NÄCHSTEN zur Referenz-Koordinate liegt. Analoges Verhalten wie bei der Funktion RSamp0.
EwOption	Optionsparameter
	0 : Die Triggerzeit der beiden Datensätze wird nicht berücksichtigt. Das Ergebnis hat dieselbe Triggerzeit wie der Eingangsdatensatz.
	1 : Die Triggerzeit der beiden Datensätze wird berücksichtigt. Das Ergebnis hat dieselbe Triggerzeit wie der Prototyp.
Ergebnis	Ergebnis der Nachabtastung.

Beschreibung:

Der als erster Parameter übergebene Datensatz wird zu den Zeiten (x-Koordinaten) nachabgetastet, die der Referenzdatensatz (2. Parameter) vorgibt. Durch die Nachabtastung wird erreicht, dass beide Datensätze synchron abgetastet vorliegen. Der erzeugte Datensatz hat dieselbe Abtastzeit und denselben x-Offset wie der Referenzdatensatz.

Lineare Interpolation wird im Allgemeinen bei kontinuierlichen Eingangssignalen verwendet.

Die **konstanten** Interpolationsarten werden dann sinnvoll verwendet, wenn das Eingangssignal aus vordefinierten diskreten Werten gebildet wird. Dies ist z.B. bei digitalen Daten der Fall oder auch bei Messdaten, die Ihrer Bedeutung nach nur ganzzahlig sein können (z.B. die aktuelle Getriebestufe bei Antrieben). Der Ergebnisdatensatz besteht nur aus Werten, die auch im Eingangsdatensatz vorhanden sind, und besitzt das selbe Datenformat.

Der erste übergebene Datensatz wird außerhalb seines x-Bereichs mit seinem Anfangs- und Endwert konstant fortgesetzt gedacht. Z. B. für alle x-Koordinaten kleiner als der x-Offset wird der Anfangswert des Datensatzes als y-Koordinate angenommen (konstante Extrapolation).

Lineare Interpolation bei digitalen Eingangsdaten ist nicht möglich, der Interpolationsparameter wird dann ggf. automatisch auf den Wert 1 (konstante Interpolation) korrigiert.

Beispiele:

An einem Fahrzeug werden Geschwindigkeit <velocity> und Getriebestufe (<gear>, enthält nur die Werte 1 - 5) gemessen. Es sollen alle Zeitpunkte ermittelt werden, an denen im 3. Gang die Geschwindigkeit kleiner als 50 km/h ist. Da beide Kanäle mit unterschiedlichen Abtastzeiten aufgenommen worden sind, müssen vor der Verrechnung beide Datensätze auf die gleiche Abtastzeit gebracht werden.

```
gear_resampled = RSampEx(gear, velocity, 2, 0)
result = (gear_resampled = 3) AND (velocity < 50)
```

Siehe auch:

RSamp0, RSamp, Red, RedEx, Red2

SamplesGate

Verfügbar ab: Professional Edition

Alle Werte ins Ergebnis übernehmen, die durch einen Steuerdatensatz ausgewählt werden.

Deklaration:

```
SamplesGate ( Datensatz, Steuerung ) -> Ergebnis
```

Parameter:

Datensatz	Datensatz
Steuerung	Steuerung
Ergebnis	Ergebnis

Beschreibung:

Die Funktion übernimmt einen Messwert des Datensatzes, wenn der zugehörige Wert der Steuerung ungleich null ist. Der Messwert wird verworfen, wenn der zugehörige Wert der Steuerung gleich null ist.

Beide Parameter haben dieselbe Struktur bezüglich Länge und Events.

Der Datensatz darf keine Segmente haben.

Der Datensatz darf äquidistant, XY oder komplex sein.

Der x-Offset (x0) bleibt erhalten.

Beispiele:

Alle Messwerte, für die der Gang 4 ist, bestimmen

```
Data4 = SamplesGate ( Data, Gear = 4 )
```

Alle Werte löschen, bei denen eine Markierung gesetzt ist (Marked[i] ist 1).

```
Remain = SamplesGate ( Data, not ( Marked ) )
```

Siehe auch:

RemoveSamples, WertIndex, GrenIndex

SAVE

Eine Variable wird in eine Datei gespeichert. Es wird das imc/FAMOS-Dateiformat benutzt.
Alternativer Name: **SICHERN**

Deklaration:

```
SAVE Variablenname Dateiname
```

Parameter:

Variablenname	Variable, die gespeichert werden soll.
Dateiname	Dateiname (oder auch kompletter Pfad), unter dem die Variable gesichert werden soll.

Beschreibung:

Dieser Befehl ist veraltet, statt dessen können Sie auch die leistungsfähigere und bequemer anzuwendende Funktion FileSave() verwenden.

Die als Parameter angegebene Variable wird gesichert. Ist kein Dateiname angegeben, wird die Variable unter ihrem Namen in dem momentan für das Sichern von Variablen eingestellten Pfad gespeichert.

Wenn der Dateiname angegeben ist, wird die Variable unter diesem Namen abgespeichert. Falls es sich um eine komplette Pfadangabe handelt, wird dieser Pfad benutzt, ansonsten der aktuell eingestellte Pfad zum Sichern von Dateien.

Der Dateiname darf auch in Anführungszeichen angegeben werden. Dies ist dann zwingend notwendig, wenn Leerzeichen im Pfad enthalten sind.

Zum Speichern von Dateien im imc/FAMOS-Format können Sie auch die leistungsfähigeren Funktionen FileSave() oder FileOpenDSF() verwenden. Mit diesen Funktionen können Sie beispielsweise auch mehrere Datensätze zusammen in einer Datei speichern.

Beispiele:

```
data = Int(data)
SAVE data
```

Die Variable data wird integriert und in einer Datei gesichert.

```
Hist = Histo(Data1, 10, 6)
SAVE Hist c:\data\histogr\hist001
Hist = Histo(Data2, 10, 6)
SAVE Hist c:\data\histogr\hist002
```

Von den Datensätzen Data1 und Data2 wird je ein Histogramm berechnet und diese im angegebenen Verzeichnis unter den Namen 'hist001.dat' und 'hist002.dat' gesichert.

```
Hist:N001 = Histo(Data1, 10, 6)
Hist:N002 = Histo(Data2, 10, 6)
SAVE Hist c:\data\histogr\hist002
```

Eine Datengruppe, bestehend aus 2 Kanälen, wird erzeugt und unter dem angegeben Dateinamen gespeichert.

```
SAVE Daten1 "c:\imc\Meine Daten\Auswert.dat"
```

Der Dateiname enthält Leerzeichen und muss deshalb in Anführungszeichen angegeben werden.

Siehe auch:

FileSave, FileOpenDSF, SDIR, ASCSAVE, LOAD

SavitzkyGolay

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Savitzky-Golay Filter zum Glätten von Datensätzen

Deklaration:

```
SavitzkyGolay ( Eingangsdaten, Ordnung, Links, Rechts, Rand ) -> Ergebnis
```

Parameter:

Eingangsdaten	Zu glättende Eingangsdaten
Ordnung	Ordnung (0..9) bzw. Grad des verwendeten Polynoms
Links	Anzahl Punkte links, >= 0
Rechts	Anzahl Punkte rechts, s>= 0
Rand	Beeinflussung des Ein- und Ausschwingens an den Rändern
	0 : Randbereiche werden konstant vom letzten gültigen Wert aus fortgesetzt.
	1 : Randbereiche werden komplett mit Nullen gefüllt.
	2 : Randbereiche entstehen durch Filterung, wobei angenommen wird, dass der Datensatz außerhalb aus Nullen besteht.
	3 : Randbereiche entstehen durch Filterung, wobei angenommen wird, dass der Datensatz außerhalb konstant fortgesetzt ist.
	4 : Randbereiche werden abgeschnitten. Die Länge verkürzt sich, der x-Offset verschiebt sich.
Ergebnis	Gefilterte Daten

Beschreibung:

Das Savitzky-Golay Filter ist ein digitales Filter, dessen Koeffizienten durch Polynom-Regression ermittelt werden.

Obwohl der Einsatz früher vor allem in der Spektroskopie erfolgte, kann das Filter sinnvoll nicht nur zum Glätten von Spektren, sondern von beliebigen Signalen angewendet werden.

Das resultierende digitale Filter hat die Breite [Punkte links + Punkte rechts + 1].

Die Punkte links gewichten vergangene Werte. Die Punkte rechts gewichten zukünftige Werte. Ist ihre Anzahl > 0, ist das Filter nicht kausal.

Bei großer Breite steigt der Rechenaufwand stark.

Die Punkte links bestimmen, wie breit das Einschwingen am Anfang ist. Die Punkte rechts bestimmen, wie breit das Ausschwingen am Ende ist.

Nur bei Ordnung < [Punkte links + Punkte rechts] gibt es eine glättende Wirkung.

Verfügbar ab imc FAMOS 7.1

Beispiele:

```
Smoothed = SavitzkyGolay( vibration, 3, 10, 10, 0 )
```

Scale

Skaliert einen Datensatz durch lineare Transformation auf einen angegebenen neuen Wertebereich
Alternativer Name: **Skali**

Deklaration:

```
Scale ( Daten, EwOben, EwUnten ) -> Ergebnis
```

Parameter:

Daten	Zu skalierender Datensatz. Erlaubte Typen: [ND],[XY].
EwOben	Obere Grenze
EwUnten	Untere Grenze
Ergebnis	Umskalierter Datensatz. Y-Wertebereich liegt zwischen [EwUnten] und [EwOben]

Beschreibung:

Der y-Wertebereich (Amplitude) eines Datensatzes wird auf einen vorgegebenen Bereich neu skaliert. Dabei wird der neue Wertebereich durch eine untere und eine obere Grenze (Minimum und Maximum) definiert. Der 2. und 3. Parameter definieren den neuen Wertebereich. Alle Zahlenwerte des Datensatzes werden linear vom alten in den neuen Wertebereich transformiert. Die Funktion ermöglicht ein bequemes Normieren auf einen passenden Wertebereich.

- Der erste Parameter darf strukturiert sein (Events/ Segmente).
- Die Einheit des Datensatzes bleibt erhalten. Ist eine Normierung auf eine Größe ohne Einheit gewünscht, ist anschließend durch eine Konstante 1 mit entsprechender Einheit zu dividieren.
- Zum Skalieren sucht die Funktion Scale() Minimum und Maximum des Datensatzes und leitet daraus zusammen mit dem geforderten Bereich die Transformationsvorschrift ab.

Beispiele:

```
NDnorm = Scale(NDdata, 0, 100) * 1 '%'
```

Ein Datensatz mit den Zahlenwerten 0.7, 0.8, 0.9, 0.8 soll auf einen Wertebereich von 0...100% skaliert werden: Der erzeugte Datensatz enthält dann die Werte 0%, 50%, 100%, 50%.

Siehe auch:

Clip, Min, Max

SDIR

Verzeichnis zum Speichern von Dateien setzen

Deklaration:

```
SDIR Verzeichnis
```

Parameter:

Verzeichnis	Vollständiger Pfadname des gewünschten Verzeichnisses.

Beschreibung:

Statt dem Kommando SDIR sollte in neu zu erstellenden Sequenzen die Funktion SetOption() verwendet werden.

Das Verzeichnis zum Speichern von Dateien wird neu gesetzt.

Nach Ausführung dieses Befehls wird dieses Verzeichnis zum Speichern von Dateien benutzt. Dieses Verzeichnis wird von den Befehlen SAVE und ASCSAVE benutzt, wenn kein voller Pfadname für die Datei angegeben wird.

Der Verzeichnisname darf auch in Anführungszeichen angegeben werden. Dies ist dann zwingend notwendig, wenn Leerzeichen im Namen enthalten sind.

Dieses Kommandos kann auch ohne Parameter (also ohne Verzeichnisangabe) aufgerufen werden. Dann wird das unter "Optionen/Verzeichnisse" eingestellte Verzeichnis wieder als aktueller Standard verwendet.

Das hier gewählte Verzeichnis bleibt gültig bis:

- der Befehl SDIR erneut aufgerufen wird
- die Funktion SetOption("Dir.DataFiles",...) aufgerufen wird
- mit dem Dialogfeld "Speichern" im Menü "Datei" eine Datei in ein anderes Verzeichnis gespeichert wird
- mit dem Dialog "Optionen"/ "Verzeichnisse" ein neues Verzeichnis zum Speichern von Dateien definiert wird

Multithreading: Der Befehl wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
FAMOS
LOAD daten
daten = 2 * daten + 3
SAVE daten
SDIR c:\versuch
SAVE daten
```

Der Datensatz DATEN wird geladen und verändert, danach im aktuell eingestellten Verzeichnis im gespeichert, anschließend noch einmal unter C:\VERSUCH\DATEN.DAT.

```
SDIR "c:\Meine Versuche vom 12.1.98"
```

Der Pfadname enthält Leerzeichen und muss deshalb in Anführungszeichen geschrieben werden.

Siehe auch:

SetOption, SAVE, ASCSAVE, LDIR, FileSave, FileOpenDSF

SDOF_Response

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Der Einmassenschwinger (SDOF, single degree of freedom system) wird mit dem übergebenen Beschleunigungsverlauf angeregt. Die Antwort (Response) wird ermittelt.

Deklaration:

```
SDOF_Response ( Beschleunigung, Frequency, Dämpfung, Modell, AppendSamples ) -> Response
```

Parameter:

Beschleunigung	Der Beschleunigungs-Zeit-Verlauf, die Zeit in Sekunden skaliert.
Frequency	Eigenfrequenz des ungedämpften Systems in "Hz"
Dämpfung	0 <= Dämpfung < 0.9. (Damping ratio) Die relative Dämpfung des Systems. Typisch 0.05 oder 0.01. 0.0 ist ein ungedämpftes System.
Modell	Nach welchem Modell erfolgt die Berechnung?
	0 : absolute acceleration model
	1 : relative displacement model
AppendSamples	Das Ergebnis ist um so viele Samples länger als die Eingangsdaten, >= 0
Response	Response

Beschreibung:

Die Funktion ermittelt den Verlauf der Auslenkung eines Systems von Feder, Masse und Dämpfer mit gegebener Eigenfrequenz und Dämpfung.

Der Fußpunkt (Aufhängungspunkt, base, support) des Einmassenschwingers wird durch eine gemessene Beschleunigung angeregt. Diese Beschleunigung wird absolut (gegenüber einer echten Ruhelage) gemessen. Als Folge der Anregung bewegt sich der angeregte Einmassenschwinger.

Wird seine Auslenkung als relativer Weg (relative displacement) bestimmt, ist dieser Weg der (mit null initialisierte) Abstand zum Fußpunkt. Die Beschleunigung hingegen wird als absoluter Wert (absolute acceleration) ermittelt, also gegenüber einer echten Ruhelage, nicht gegenüber dem Fußpunkt.

absolute acceleration model. Das Ergebnis sind die absoluten Beschleunigungswerte, wobei die Beschleunigung genauso skaliert ist wie beim Beschleunigungs-Zeit-Verlauf, typisch in "g" Erdbeschleunigung) oder "m/s^2".

relative displacement model. Das Ergebnis ist die relative Auslenkung (Weg) und erhält als Einheit "m" (Meter). Dazu ist es erforderlich, dass der Beschleunigungs-Zeit-Verlauf in "m/s^2" skaliert ist, nicht in "g".

Die Funktion nimmt an, dass zu Beginn das System in Ruhe bei 0 ist, also Weg=0 und Geschwindigkeit=0.

Das Ergebnis ist über der Eigenfrequenz aufgetragen.

Dann sollte eine passende Länge gewählt werden.

Zusammenhang mit dem Shock Response Spectrum (SRS): Das SRS wird gebildet, indem für jede gewünschte Frequenz SDOF_RResponse() aufgerufen wird und Minimum bzw. Maximum des Ergebnisses bestimmt werden.

Das Ergebnis hat dieselbe Abtastfrequenz wie die Eingangsdaten. Sollen aus dem Ergebnis später Minimal- oder Maximalwerte abgelesen oder eine Klassierung durchgeführt werden, gilt:

Ist die Eigenfrequenz nicht klein gegenüber der Abtastfrequenz, ergeben sich Fehler, weil nicht genau beim Extremwert abgetastet wird. Z.B. ergibt sich ein Fehler von bis zu 1%, wenn das Verhältnis der Frequenzen 23 beträgt. Ggf. kann vorher oder auch nachher interpoliert werden.

Wird AppendSamples = 0 gesetzt, entspricht das Ergebnis dem Verlauf der Auslenkung, der zur Bestimmung des primary SRS, auch initial SRS, benutzt wird.

Vor allem bei niedrigen Eigenfrequenzen kommt es vor, dass innerhalb der (kurzen) Dauer des Beschleunigungs-Zeit-Verlaufs die Systemantwort zwar mit einer Schwingung begonnen hat, aber das Minimum oder Maximum noch gar nicht erreicht hat.

Sollen später nur Maximalwerte bestimmt werden, reicht es i. Allg., eine Länge von einer Periode der Eigenfrequenz anzuhängen.

Soll später eine Klassierung durchgeführt werden, müssen so viele Perioden angehängt werden, bis das Antwort auf einen vernachlässigbaren Wert abgeklungen ist. Bei einer Dämpfung von 0.05 kann das z.B. eine Anzahl von 30 Perioden sein.

Die Eingangsdaten dürfen Events und Segmente haben.

Vorzeichen, Richtungen

Die Auslenkung als relative displacement z ist der Abstand zwischen der Masse und dem Aufhängungspunkt. Der Aufhängungspunkt selbst bewegt sich mit der als Parameter angegebenen absoluten Beschleunigung. Alle Größen werden in dieselbe Richtung gemessen, positiv ist weg vom Aufhängungspunkt.

Sowohl Masse als auch Aufhängungspunkt seien anfangs in Ruhe. Nun werde der Aufhängungspunkt konstant positiv beschleunigt:

Die Masse bewegt sich im ersten Augenblick noch nicht, die Feder wird gestaucht. Dann kommt langsam die Masse in Bewegung. Sie bewegt sich erst einmal in dieselbe Richtung wie der Aufhängungspunkt. Ihre absolute Beschleunigung ist auch positiv.

Die Masse bewegt sich dann immer schneller werdend in dieselbe Richtung wie der Aufhängungspunkt. Aber der Aufhängungspunkt beschleunigt vorerst noch stärker als die Masse. Für den Abstand z zwischen der Masse und dem Aufhängungspunkt gilt dabei, dass dieser sich erst einmal verringert. Da für z ein Anfangswert von 0 angenommen wird, wird z dann negativ.

z < 0 heißt also, dass die Masse näher am Aufhängungspunkt ist als beim Start. z > 0 heißt, dass die Masse weiter weg vom Aufhängungspunkt ist als beim Start.

Beispiele:

Der Fußpunkt eines Einmassenschwingers wird mit einer kontinuierlichen Beschleunigung angeregt. Die absolute Beschleunigung des Einmassenschwingers soll ermittelt werden. Eigenfrequenz=1000Hz, Dämpfung=0.05

```
SDOF_Acc = SDOF_Response ( Acceleration, 1000, 0.05, 0, 0)
```

Der Fußpunkt eines Einmassenschwingers wird mit einer kurzen Beschleunigung (in m/s^2) angeregt. Die relative Auslenkung des Einmassenschwingers (in m) soll ermittelt werden. Eigenfrequenz=100Hz, Dämpfung=0.05. Dabei wird das Ergebnis um 30 Perioden der Eigenfrequenz verlängert, um das Schwingungen nach Ende der Anregung ebenfalls zu erhalten.

```
f0 = 100 ; Hz
append_samples = floor ( 30 / (xdel?(Acceleration) * f0 ))
RelativeDisplacement = SDOF_Response ( Acceleration, f0, 0.05, 1, append_samples)
```

Eine ausgewählte Linie des Schockantwortspektrums soll nachvollzogen werden

```
SRS = ShockResponseSpectrum ( Acceleration, 5.0, 500.0, 100, 0.05, 14, 0 )
B = SDOF_Response(Acceleration, SRS[1].x, 0.05, 0, 1+1 / (SRS[1].x*xdel?(Acceleration)) )
B_max = abs ( max ( B )) ; == SRS[1].y
```

Untersuchung der Vorzeichen bei kurzer konstanter Beschleunigung

Dabei ist zu sehen, wie abs_a langsam steigt. Aber a ist die ganze Zeit schon groß. z beginnt festlegungsgemäß mit 0. Dann wird z immer kleiner. Kleiner als 0, weil sich Masse und Aufhängungspunkt näher kommen. Erst später schiebt dann die Feder die Masse schneller vor sich her, z wird dann größer.

```
a = ramp(0,1e-4,100)*0+1
abs_a = SDOF_Response(a,100.0,0.0,0,0)
z = SDOF_Response(a,100.0,0.0,1,0)
```

Siehe auch:

ShockResponseSpectrum

SearchLevel

Suche nach vorgebbaren Pegel- u. Anstiegsbedingungen in einem Datensatz.
Alternativer Name: **SuchePegel**

Deklaration:

```
SearchLevel ( Daten, EwPegelBedingung, EwPegel1, EwPegel2, EwAnstiegsBedingung, EwAnstieg1, EwAnstieg2, EwOption ) -> XYGefunden
```

Parameter:

Daten	Zu durchsuchender Datensatz. Erlaubte Typen: [ND],[XY]
EwPegelBedingung	Bedingung für Pegel (Amplitude)
	0 : Pegel beliebig
	1 : Kleiner oder gleich..
	2 : Größer oder gleich....
	3 : Im Intervall von.. bis..
	4 : Außerhalb Intervall von.. bis..
EwPegel1	Pegelwert, Bedeutung abhängig von der Pegel-Bedingung. Für Option 1 und 2 das Limit, für 3 und 4 die untere Intervallgrenze, sonst auf 0 zu setzen.
EwPegel2	Pegelwert, Bedeutung abhängig von der Pegel-Bedingung. Für Option 3 und 4 die obere Intervallgrenze, sonst auf 0 zu setzen.
EwAnstiegsBedingung	Bedingung für Anstieg
	0 : Anstieg beliebig
	1 : Kleiner oder gleich..
	2 : Größer oder gleich....
	3 : Im Intervall von.. bis..
	4 : Außerhalb Intervall von.. bis..
EwAnstieg1	Anstiegswert, Bedeutung abhängig von der Anstiegs-Bedingung. Für Option 1 und 2 das Limit, für 3 und 4 die untere Intervallgrenze, sonst auf 0 zu setzen.
EwAnstieg2	Anstiegswert, Bedeutung abhängig von der Anstiegs-Bedingung. Für Option 3 und 4 die obere Intervallgrenze, sonst auf 0 zu setzen.
EwOption	Option für Verknüpfung von Pegel- und Anstiegsbedingung
	0 : Die Funktion ermittelt alle XY-Paare, bei denen beide Bedingungen gleichzeitig erfüllt sind. Es findet also eine Suche nach bestimmten Zuständen statt.
	1 : Die Funktion sucht alle Übergänge, bei denen die Pegelbedingung erfüllt wird, also von falsch auf wahr umspringt. Zusätzlich muss die Anstiegsbedingung erfüllt sein. Die X-Position wird am Übergang linear interpoliert. Hier werden also keine Zustände, sondern Übergänge im Datensatz gesucht.
XYGefunden	XY-Datensatz mit den Paaren x, y, die obige Bedingungen erfüllen.

Beschreibung:

Die Funktion sucht Punkte im Datensatz, die bestimmte Pegel- und Anstiegsbedingungen erfüllen.

Zuerst kann eine Bedingung für den y-Wert (Pegel, Amplitude) angeben werden.

EwPegelbedingung	
0	Pegel egal
1	Pegel <=EwPegel1
2	Pegel >= EwPegel1
3	EwPegel1 <= Pegel <= EwPegel2 (Wert im Intervall von .. bis..)
4	Pegel <=EwPegel1 ODER Pegel >=EwPegel2 (Wert nicht im Intervall von.. bis..)

Für den Anstieg in einem Punkt kann ebenfalls eine Bedingung angegeben werden.

EwAnstiegsbedingung	
0	Anstieg egal

1	Anstieg <=EwAnstieg1
2	Anstieg >= EwAnstieg1
3	EwAnstieg1 <= Anstieg <= EwAnstieg2 (Anstieg im Intervall von .. bis..)
4	Anstieg <=EwAnstieg1 ODER Anstieg >=EwAnstieg2 (Anstieg nicht im Intervall von.. bis..)

Der letzte Parameter [EwOption] schließlich gibt an, wie die zuvor formulierten Bedingungen zu verknüpfen sind. Die Wirkung diese Optionsparameters verdeutlicht folgendes Beispiel:

Der zu untersuchende Datensatz habe die Werte 0, 2, 6, 7, 9 V bei einer Abtastzeit von 1s. Gesucht wird nach einem Pegel > 4V mit der Formel

```
result = SearchLevel(data, 2, 4, 0, 0, 0, 0, 0)
```

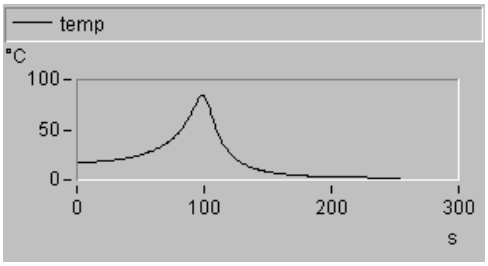
Die Funktion liefert einen XY-Datensatz mit den Wertepaaren (6V, 2s), (7V, 3s), (9V, 4s). Wird die Option dagegen auf Übergangssuche gesetzt:

```
result = SearchLevel(data, 2, 4, 0, 0, 0, 0, 1)
```

liefert die Funktion nur ein Wertepaar, nämlich (4V, 1.5s). Hier wurde die Zeit bestimmt, bei der (bei linearer Interpolation) die Bedingung wahr wurde.

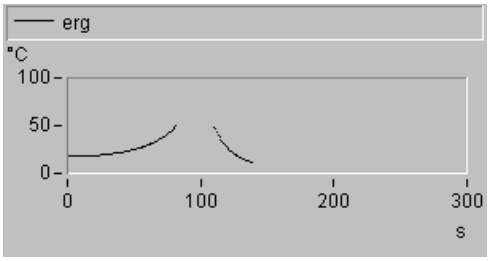
Beispiele:

Zu untersuchender Datensatz:



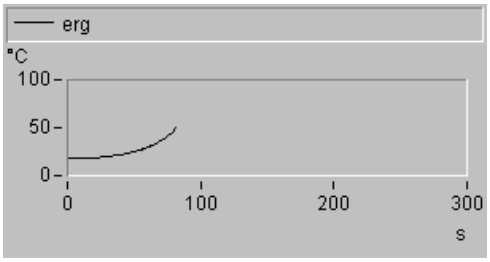
Suche nach allen Werten im Bereich von 10-50 °C, Anstieg egal:

```
result = SearchLevel(temp, 3, 10, 50, 0, 0, 0, 0)
```



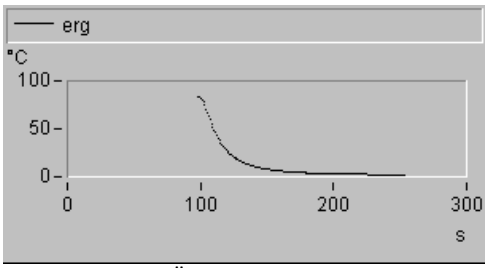
Suche nach allen Werten im Bereich von 10-50 °C bei positivem Anstieg:

```
result = SearchLevel(temp, 3, 10, 50, 2, 0, 0, 0)
```



Suche nach allen Werten mit negativem Anstieg, Pegel egal:

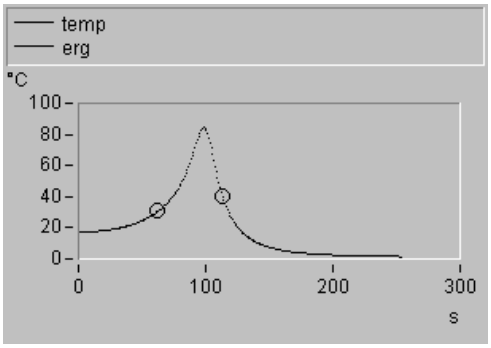
```
result = SearchLevel(temp, 0, 0, 0, 1, 0, 0, 0)
```



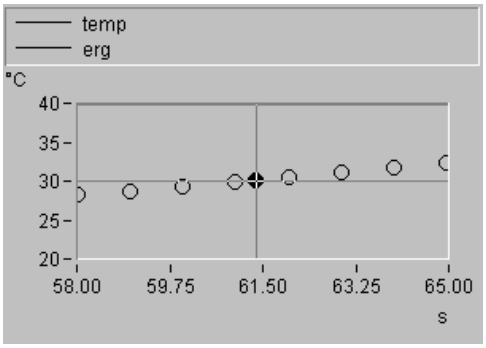
Suche nach allen Übergängen, wo die Temperatur in das Intervall 30- 40°C von oben oder unten einläuft:

```
result = SearchLevel(temp, 3, 30, 40, 0, 0, 0, 1)
```

Der Ergebnisdatensatz besteht aus 2 Punkten (unten als Kreise dargestellt).



Stark gezoomt, wird die Interpolation an der 30°C -Grenze deutlich:



Siehe auch:

Top, All0, xMax, PosiEx2

SegLen?

Die Segmentlänge eines Datensatzes wird ermittelt.
Alternativer Name: **SegLang?**

Deklaration:

```
SegLen? ( Daten ) -> EwSegLang
```

Parameter:

Daten	Datensatz, dessen Segmentlänge ermittelt werden soll.
EwSegLang	Segmentlänge, 0 bedeutet keine Segmentierung.

Beschreibung:

Die Funktion liefert die Segmentlänge des Datensatzes zurück. Eine Länge von 0 bedeutet keine Segmentierung.

Beispiele:

```
IF SegLen?(data) > 0
   SetSegLen(data, 0)
END
```

Eine eventuelle Segmentierung des Datensatzes wird aufgehoben.

Siehe auch:

SetSegLen, MatrixInfo

SelBuildVarName

Aus Kanal- und Messungsbezeichner wird ein kompletter FAMOS-Variablenname zusammengebaut. Kanal und Messung können sowohl durch einen festen Namen als auch über ihren Index in der Kanal- bzw. Messungsliste des Datenselektors angegeben werden.

Deklaration:

```
SelBuildVarName ( Messung, Kanal, EwOption ) -> TxVarName
```

Parameter:

Messung	Name der Messung oder Index (1,2,..) des Eintrags in der Messungsliste.
Kanal	Name des Kanals oder Index (1,2,..) des Eintrags in der Kanalliste.
EwOption	Optionsparameter
	0 : Es wird geprüft, ob die entsprechende Variable in FAMOS existiert. Falls nicht, wird ein leerer String zurückgegeben.
	1 : Der Name wird ungeprüft zurückgegeben.
TxVarName	Kompletter Variablename für FAMOS.

Beschreibung:

Der erzeugte Variablenname hat die folgende Struktur:

```
KanalName@MessungsName
```

Falls Kanal- oder Messungsname nicht den gültigen Regeln für Variablennamen in Sequenzen gehorcht (z.B. Sonderzeichen oder Leerzeichen enthält), wird der jeweilige Bezeichner in **{...}** eingeklammert, wodurch der zurückgegebene Name dann trotzdem innerhalb von Sequenzen verwendet werden kann.

Um den zurückgebenen Variablenbezeichner anschließend in Sequenzen verwenden zu können, verwenden Sie den Text-Indirektionsoperator **<...>**. Sie sollten außerdem vor der Verwendung testen, ob eine Variable dieses Namens in FAMOS überhaupt vorhanden ist (vgl. Beispiel2).

Der Inhalt der Messungs- und Kanalliste des Datenselektors wird durch die manuelle Selektion in der Variablenliste/Messungsansicht und/oder die Funktionen SelMeasListSetName() bzw. SelChanListSetName() bestimmt.

Beispiele:

Kanalname	Messungsname	Ergebnis
"Channel1"	"Test1"	"Channel1@Test1"
"Channel1"	Leer	"Channel1"
"CH**1"	"Test1"	"{CH**1}@Test1"
"CH**1"	"Test[1]"	"{CH**1}@{Test[1]}"

Es werden nacheinander alle Einträge des Datenselektors aufgezählt. Wenn der jeweilige Eintrag auf eine existierende Variable verweist, wird deren X-Offset geändert.

```
MCount = SelMeasListSize?()
FOR M = 1 TO MCount
   CCount = SelChanListSize?()
   FOR C = 1 TO CCount
      TxVarName = SelBuildVarName(M, C, 0)
      ; Option 0: Leerer Text, wenn Variable nicht existiert!
      IF TxVarName <> ""
         XOFFSET <TxVarName> 10
      END
   END
END
```

Falls die Messung #1 im Datenselektor die Kanäle "Voltage" and "Current" enthält, wird die Leistung berechnet und in einem separaten Kurvenfenster angezeigt.

```
TxUName = SelBuildVarName(1, "Voltage", 0)
TxIName = SelBuildVarName(1, "Current", 0)
; Option 0: Leerer Text, wenn Variable nicht existiert!
IF (TxUName <> "") AND (TxIName <> "")
   Power = <TxUName> * <TxIName>
   SHOW Power
END
```

Siehe auch:

SelMeasListName?, SelMeasListSize?, SelChanListName?, SelChanListSize?, SelMeasListSetName, SelChanListSetName

SelChanListName?

Ein Eintrag der Kanalliste im Datenselektor wird zurückgegeben.

Deklaration:

```
SelChanListName? ( EwIndex ) -> TxName
```

Parameter:

EwIndex	Index des gewünschten Eintrages.
TxName	Name des Kanals.

Beschreibung:

Die Funktion wird selten verwendet. Um aus Messungs- und Kanalliste einen vollständigen FAMOS-Variablennamen zu erzeugen, verwenden Sie die Funktion "SelBuildVarName".

Der Inhalt der Messungs- und Kanalliste des Datenselektors wird durch die manuelle Selektion in der Variablenliste/Messungsansicht und/oder die Funktionen SelMeasListSetName() bzw. SelChanListSetName() bestimmt.

Beispiele:

Es werden alle Einträge der Kanalliste des Datenselektors aufgezählt und angezeigt.

```
count = SelChanListSize?()
c = 1
WHILE c <= count
   TxMeas = SelChanListName?(c)
   BoxOutput(TxMeas, EMPTY, "", 1)
   c = c + 1
END
```

Siehe auch:

SelMeasListName?, SelChanListSize?, SelBuildVarName, SelMeasListSetName, SelChanListSetName

SelChanListSetName

Ein Eintrag in der Kanalliste des Datenselektors wird gesetzt.

Deklaration:

```
SelChanListSetName ( EwIndex, TxName, GruppenName [, EwOption] )
```

Parameter:

EwIndex	Index des zu setzenden Eintrags in der Kanalliste des Datenselektors. Der erste Eintrag hat den Index 1.
TxName	Name des Kanals. Ein leerer Text löscht den bisherigen Eintrag.
GruppenName	Name der übergeordneten Gruppe oder leerer Text
EwOption	Option (optional , Standardwert: 0)
	0 : Die Änderung wird sofort wirksam (Standard).
	1 : Änderung wird erst beim nächsten Aufruf von SelListControl(0) wirksam.

Beschreibung:

Die Funktion setzt einen Eintrag in der Kanalliste des Datenselektors, weist also einem Kanalnamen eine Nummer zu.

Die Funktion stellt somit die Automatisierung des manuellen Selektierens eines Kanals in der Variablenliste/Messungsansicht dar.

Um einen Kanal anzugeben, der zu einer Gruppe gehört, können entweder Kanal- und Gruppenname im 2. und 3. Parameter separat oder im 2. Parameter in der üblichen Schreibweise 'Gruppenname:Kanalname' zusammen angegeben werden. Im letzteren Fall ist der 3. Parameter leer zu lassen.

Der Parameter [Option] steuert, ob die Änderung unverzüglich oder erst durch einen expliziten Aufruf von SelListControl(0) wirksam wird. Dies betrifft die Aktualisierung von Anzeige-Elementen, die den Datenselektor verwenden (z.B. ein Kurvenfenster mit einer Verknüpfung zu einer Messungs- oder Kanalnummer), die Anzeige der entsprechenden Nummerierung in der Variablenliste/Messungsansicht und die Auslösung des Panel-Ereignisses 'Datenselektion geändert'. Geben Sie eine 1 an, wenn sie nacheinander mehrere Einträge im Datenselektor setzen möchten und rufen Sie abschließend SelListControl(0) auf.

Es wird nicht geprüft, ob ein Kanal mit dem angegebenen Namen zum Zeitpunkt des Aufrufs tatsächlich vorhanden ist.

Multithreading: Die Funktion wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

Ein Panel enthält ein Kurvenfenster, welches zur Anzeige des 1. Kanals der 1. selektierten Messung konfiguriert ist ('Kanal #1 @ Messung #1') und einen Knopf 'Zeigen'. Der Anwender lädt zunächst manuell alle gewünschten Messungen. Auf Knopfdruck werden dann für jede Messung alle Kanäle, die mit 'U_' anfangen, für jeweils für 2 Sekunden angezeigt.

Ereignis-Sequenz 'Knopf gedrückt (Zeigen)':

```
measurements = MeasNames?("*")
FOREACH ELEMENT name in measurements
   SelMeasListSetName(1, name, 1)
   channels = MeasChanNames?(name, "U_*")
   FOREACH ELEMENT chan in channels
      SelChanListSetName(1, chan, "", 1)
      SelListControl(0) ; jetzt Kurvenfenster aktualisieren
      Sleep(2)
   END
END
```

Siehe auch:

SelChanListName?, SelMeasListSetName, SetMeasurementName, SelBuildVarName, MeasChanNames?

SelChanListSize?

Die Anzahl der fortlaufend nummerierten Einträge in der Kanalliste des Datenselektor wird zurückgegeben.

Deklaration:

```
SelChanListSize? ( ) -> EwAnzahl
```

Parameter:

EwAnzahl	Anzahl der Einträge in der Kanalliste.

Beschreibung:

Es werden nur die Kanäle mit einer fortlaufenden Nummerierung ab 1 berücksichtigt, bei dem seltenen Fall einer "Lücke" in der Kanalliste wird die Zählung abgebrochen. Wenn beispielsweise die Kanäle #1 bis #3 selektiert sind, wird eine 3 zurückgegeben. Wenn dagegen die Kanäle #1, #2 und #4 selektiert sind, der Kanal #3 aber nicht belegt ist, ist das Ergebnis eine 2.

Der Inhalt der Messungs- und Kanalliste des Datenselektors wird durch die manuelle Selektion in der Variablenliste/Messungsansicht und/oder die Funktionen SelMeasListSetName() bzw. SelChanListSetName() bestimmt.

Beispiele:

Es werden nacheinander alle Einträge des Datenselektors aufgezählt. Wenn der jeweilige Eintrag auf eine existierende Variable verweist, wird deren X-Offset geändert.

```
MCount = SelMeasListSize?()
M = 1
WHILE M <= MCount
   CCount = SelChanListSize?()
   C = 1
   WHILE C <= CCount
      TxVarName = SelBuildVarName(M, C, 0)
      ; Option 0: Leerer Text, wenn Variable nicht existiert!
      IF TxVarName <> ""
         XOFFSET <TxVarName> 10
      END
      C = C+1
   END
   M = M+1
END
```

Siehe auch:

SelMeasListName?, SelChanListName?, SelBuildVarName, SelMeasListSetName, SelChanListSetName

SelListControl

Steuerung des Datenselektors

Deklaration:

```
SelListControl ( EwAuftrag )
```

Parameter:

EwAuftrag	Auftrag
	0 : Anzeigen (Kurvenfenster, Panel-Elemente, Variablenliste/Messungsansicht) werden aktualisiert. GGf. wird das Panel-Ereignis 'Datenselektion geändert' ausgelöst.
	1 : Messungs- und Kanalliste werden komplett gelöscht. Die Änderung wird sofort wirksam.
	2 : Messungs- und Kanalliste werden komplett gelöscht. Die Änderung wird erst beim nächsten Aufruf von SelListControl(0) wirksam. Verwenden Sie diesen Wert, wenn Sie anschließend die Liste neu füllen möchten.

Beschreibung:

Mit [Auftrag] = 0 werden vorherige Änderungen des Datenselektors wirksam gemacht. Dies betrifft die Aktualisierung von Anzeige-Elementen, die den Datenselektor verwenden (z.B. ein Kurvenfenster mit einer Verknüpfung zu einer Messungs- oder Kanalnummer), die Anzeige der entsprechenden Nummerierung in der Variablenliste/Messungsansicht und die Auslösung des Panel-Ereignisses 'Datenselektion geändert'.

Beispiele:

Die Einträge des Datenselektors werden auf feste Namen gesetzt:

```
SelListControl(2) ; aktuellen Inhalt löschen
SelMeasListSetName(1, "Messung_12_04_2015", 1)
SelMeasListSetName(2, "Messung_12_05_2015", 1)
SelChanListSetName(1, "channel1", "", 1)
SelChanListSetName(2, "channel2", "", 1)
SelListControl(0) ; Anzeigen aktualisieren
```

Siehe auch:

SelMeasListSetName, SelChanListSetName, MeasNames?, SetMeasurementName

SelMeasListName?

Ein Eintrag der Messungsliste im Datenselektor wird zurückgegeben.

Deklaration:

```
SelMeasListName? ( EwIndex ) -> TxName
```

Parameter:

EwIndex	Index des gewünschten Eintrages.
TxName	Name der Messung.

Beschreibung:

Die Funktion wird selten verwendet. Um aus Messungs- und Kanalliste einen vollständigen FAMOS-Variablennamen zu erzeugen, verwenden Sie die Funktion "SelBuildVarName".

Der Inhalt der Messungs- und Kanalliste des Datenselektors wird durch die manuelle Selektion in der Variablenliste/Messungsansicht und/oder die Funktionen SelMeasListSetName() bzw. SelChanListSetName() bestimmt.

Beispiele:

Es werden alle Einträge der Messungsliste des Datenselektors aufgezählt und angezeigt.

```
MCount = SelMeasListSize?()
FOR M = 1 TO MCount
   TxMeas = SelMeasListName?(M)
   BoxOutput(TxMeas, EMPTY, "", 1)
END
```

Alle Kanäle, die zur Messung #1 im Datenselektor gehören, werden gelöscht.

```
measurement = SelMeasListName?(1)
DELETE *@{<measurement>}
```

Siehe auch:

SelMeasListSize?, SelChanListSize?, SelBuildVarName, SelMeasListSetName, SelChanListSetName

SelMeasListSetName

Ein Eintrag in der Messungsliste des Datenselektors wird gesetzt.

Deklaration:

```
SelMeasListSetName ( EwIndex, TxName [, EwOption] )
```

Parameter:

EwIndex	Index des zu setzenden Eintrags in der Messungsliste. Der erste Eintrag hat den Index 1.
TxName	Name der Messung. Ein leerer Text löscht den bisherigen Eintrag. "." steht für [Keine Messung].
EwOption	Option (optional , Standardwert: 0)
	0 : Die Änderung wird sofort wirksam (Standard).
	1 : Änderung wird erst beim nächsten Aufruf von SelListControl(0) wirksam.

Beschreibung:

Die Funktion setzt einen Eintrag in der Messungsliste des Datenselektors, weist also einem Messungsnamen eine Nummer zu.

Die Funktion stellt somit die Automatisierung des manuellen Selektierens einer Messung in der Variablenliste/Messungsansicht dar.

Der Parameter [Option] steuert, ob die Änderung unverzüglich oder erst durch einen expliziten Aufruf von SelListControl(0) wirksam wird. Dies betrifft die Aktualisierung von Anzeige-Elementen, die den Datenselektor verwenden (z.B. ein Kurvenfenster mit einer Verknüpfung zu einer Messungs- oder Kanalnummer), die Anzeige der entsprechenden Nummerierung in der Variablenliste/Messungsansicht und die Auslösung des Panel-Ereignisses 'Datenselektion geändert'. Geben Sie eine 1 an, wenn sie nacheinander mehrere Einträge im Datenselektor setzen möchten und rufen Sie abschließend SelListControl(0) auf.

Es wird nicht geprüft, ob eine Messung mit dem angegebenen Namen zum Zeitpunkt des Aufrufs tatsächlich vorhanden ist.

Multithreading: Die Funktion wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

Ein Panel enthält ein Kurvenfenster, welches zur Anzeige des 1. Kanals der 1. selektierten Messung konfiguriert ist ('Kanal #1 @ Messung #1') und einen Knopf 'Zeigen'. Der Anwender lädt zunächst manuell alle gewünschten Messungen und selektiert einen Kanalnamen (bekommt die Nummer #1) in der Variablenliste/Messungsansicht. Auf Knopfdruck werden dann alle aktuell vorhandenen Messungen aufgezählt und das entsprechende Kurvenbild für den gewählten Kanal jeweils für 2 Sekunden angezeigt.

Ereignis-Sequenz 'Knopf gedrückt (Zeigen)':

```
measurements = MeasNames?("*")
FOREACH ELEMENT name IN measurements
   SelMeasListSetName(1, name)
   Sleep(2)
END
```

Selbe Situation wie oben, es werden aber nacheinander pro Messung alle Kanäle, die mit 'U_' anfangen, angezeigt.

```
measurements = MeasNames?("*")
FOREACH ELEMENT name in measurements
   SelMeasListSetName(1, name, 1)
   channels = MeasChanNames?(name, "U_*")
   FOREACH ELEMENT chan in channels
      SelChanListSetName(1, chan, "", 1)
      SelListControl(0) ; jetzt Kurvenfenster aktualisieren
      Sleep(2)
   END
END
```

Eine Verzeichnisstruktur enthält eine Reihe von Unterverzeichnissen, die jeweils eine Messung repräsentieren und die Kanäle 'Voltage' and 'Current' enthalten. Ein Panel enthält 3 Kurvenfenster, die für die Anzeige der Kanäle 'Voltage', 'Current' und 'Power' der ersten selektierten Messung konfiguriert sind. Beim Laden eines Verzeichnisses per Datenquellen-Browser wird die neue Messung automatisch als 1. Messung selektiert, die Variable 'Power' berechnet und im selben Verzeichnis wie die Quelldaten abgespeichert. Die Anzeigen im Panel aktualisieren sich entsprechend.

Ereignis-Sequenz 'Messung verfügbar':

```
IF PA2 = 0 ; neue Messung erkannt
   SelMeasListSetName(1, PA1)  ; neue Messung erhält die Nummer 1
   SelUseMeasurement(1)
   Power = Voltage * Current
   SetMeasurementName(Power, PA1)
   fileName = FsSplitPath(FileName?(Voltage), 8) + "\power"
```

```
   FileSave(fileName, "", 0, Power)
END
```

Siehe auch:

SelMeasListName?, SelChanListSetName, SelBuildVarName, SetMeasurementName, MeasNames?

SelMeasListSize?

Die Anzahl der fortlaufend nummerierten Einträge in der Messungsliste des Datenselektors wird zurückgegeben.

Deklaration:

```
SelMeasListSize? ( ) -> EwAnzahl
```

Parameter:

EwAnzahl	Anzahl der Einträge in der Messungsliste.

Beschreibung:

Es werden nur die Messungen mit einer fortlaufenden Nummerierung ab 1 berücksichtigt, bei dem seltenen Fall einer "Lücke" in der Messungsliste wird die Zählung abgebrochen. Wenn beispielsweise die Messungen #1 bis #3 selektiert sind, wird eine 3 zurückgegeben. Wenn dagegen die Messungen #1, #2 und #4 selektiert sind, die Messung #3 aber nicht belegt ist, ist das Ergebnis eine 2.

Der Inhalt der Messungs- und Kanalliste des Datenselektors wird durch die manuelle Selektion in der Variablenliste/Messungsansicht und/oder die Funktionen SelMeasListSetName() bzw. SelChanListSetName() bestimmt.

Beispiele:

Es werden nacheinander alle Einträge des Datenselektors aufgezählt. Wenn der jeweilige Eintrag auf eine existierende Variable verweist, wird deren X-Offset geändert.

```
MCount = SelMeasListSize?()
FOR M = 1 TO MCount
   CCount = SelChanListSize?()
   FOR C = 1 TO CCount
      TxVarName = SelBuildVarName(M, C, 0)
      ; Option 0: Leerer Text, wenn Variable nicht existiert!
      IF TxVarName <> ""
         XOFFSET <TxVarName> 10
      END
   END
END
```

Siehe auch:

SelMeasListName?, SelChanListSize?, SelBuildVarName, SelMeasListSetName, SelChanListSetName

SelUseMeasurement

Formelinterpreter: Legt fest, aus welcher Messung Variablen verwendet werden.

Deklaration:

```
SelUseMeasurement ( MessungsnummerOderName ) -> EwStatus
```

Parameter:

MessungsnummerOderName	Zu verwendende Messung. Entweder Zahl oder Text. Die Zahl bezieht sich dann auf den entsprechenden Eintrag im Datenselektor (Variablenliste/Messungsansicht). Eine 1 z.B. bedeutet, dass die erste selektierte Messung verwendet wird. Alternativ kann auch direkt der Name der Messung angegeben werden, dies funktioniert immer unabhängig von der aktuellen Zuordnung im Datenselektor.
EwStatus	Ergebnis der Funktion
	0 : Der angegebenen Messungsnummer ist momentan keine Messung zugeordnet.
	1 : OK

Beschreibung:

Die hier angegebene Messung wird durch den Formelinterpreter beim Auffinden von Variablen verwendet. Der Variablenname wird dann ggf. automatisch um diese Messung ergänzt. Dies ist sehr nützlich, wenn Sequenzen auf feste Kanalnamen aus unterschiedlichen Messungen angewendet werden sollen.

Der Inhalt der Messungs- und Kanalliste des Datenselektors wird durch die manuelle Selektion in der Variablenliste/Messungsansicht und/oder die Funktionen SelMeasListSetName() bzw. SelChanListSetName() bestimmt.

Die Funktion bietet eine einfache Alternative zum Zusammenbau des kompletten Variablenbezeichners Kanalname@Messungsname mittels der Funktion SelBuildVarName().

Eine 0 oder leerer Text für den Parameter bedeutet, dass keine Standard-Messung verwendet wird.

Nachdem mit SelUseMeasurement() eine Standardmessung festgelegt wurde, kann auch bei der Erzeugung neuer Variablen durch Zuweisung die vereinfachte Schreibweise

```
NeueVariable@ = ...
```

verwendet werden. Die neue Variable wird dann automatisch der aktuellen Standardmessung zugeordnet.

Beispiel: Die aktuell bearbeiteten Messungen enthalten den Kanal 'current'. Der zur Messung #1 gehörige Kanal soll mit einem festen Offset versehen werden:

Variante 1:

```
SelUseMeasurement(1)
current = current + 10
```

Variante 2:

```
TxVarName = SelBuildVarName(1, "current", 0)
<TxVarName> = <TxVarName> + 10
```

Die hier festgelegte Messung bleibt gültig bis zum:

- nächsten Aufruf der Funktion SelUseMeasurement()
- nächsten Neustart einer Top-Level-Sequenz
- Menübefehl 'Neubeginn'.

Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Hinweis: Wenn bei einer Variablen keine Messung explizit angegeben ist, sucht der Formelinterpreter zunächst unter den Variablen mit der hier ausgewählten Messungszuordnung. Erst bei Misserfolg wird die Suche unter den Variablen ohne Messungszugehörigkeit fortgeführt.

Beispiele:

Die aktuell bearbeiteten Messungen enthalten die Kanäle 'Voltage' und 'Current'. Zur aktuell selektierten Messung #1 wird die Leistung berechnet und in einem separaten Kurvenfenster angezeigt.

```
IF SelUseMeasurement(1) = 0
   EXITSEQUENCE
END
Power = Voltage * Current
; oder: Power@ = Voltage * Current ;=> die Variable 'Power' wird der aktuellen Messung zugeordnet.
SHOW Power
```

Wenn Sie obenstehende Sequenz robuster gestalten wollen, können Sie mit der Funktion VarExist?() zusätzlich prüfen, ob die erwarteten Kanäle zur Laufzeit tatsächlich vorhanden sind:

```
IF SelUseMeasurement(1)
   IF VarExist?("Voltage") AND VarExist?("Current")
      Power = Voltage * Current
      SHOW Power
   END
END
```

Ein Panel dient zur Visualisierung von Messdaten. Die zu untersuchenden Messungen enthalten jeweils einen Kanal "channel1", der vor der Anzeige und weiteren Auswertung geglättet werden soll. Das Glätten wird in der Panel-Ereignissequenz "Messung verfügbar" ausgeführt:

```
IF PA2 = 0  ; neue Messung?
   SelUseMeasurement(PA1)  ; PA1: Name der geladenen Messung
   channel1 = Smo(channel1, 2)
END
```

Siehe auch:

SelMeasListName?, SelChanListName?, SelBuildVarName, SelMeasListSetName, SelChanListSetName

SEQUENCE

Sequenz ausführen
Alternativer Name: **SEQUENZ**

Deklaration:

```
SEQUENCE Dateiname Par1 Par2 Par3 Par4 Par5 Par6 Par7 Par8 Par9 ...
```

Parameter:

Dateiname	Dateiname der Sequenz-Datei
Par1	1. Parameter
Par2	2. Parameter
Par3	3. Parameter
Par4	4. Parameter
Par5	5. Parameter
Par6	6. Parameter
Par7	7. Parameter
Par8	8. Parameter
Par9	9. Parameter
...	Weitere Parameter (maximal 20)

Beschreibung:

Eine Sequenz wird ausgeführt. Der Name der Sequenz ist ein Dateiname und wird als erster Parameter übergeben. Wenn die auszuführende Sequenz Parameter erwartet, müssen diese als weitere Parameter übergeben werden. Die übergebenen Parameter werden innerhalb der gerufenen Sequenz über die symbolischen Bezeichner PA1, PA2 etc. adressiert

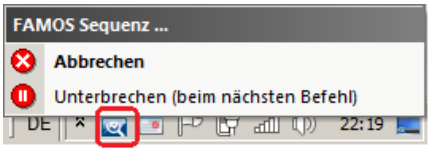
Der Name der Sequenz darf auch in Anführungszeichen angegeben werden. Dies ist dann zwingend notwendig, wenn Leerzeichen im Pfad enthalten sind.

Wenn bei dem Dateinamen kein kompletter Pfad angegeben ist, wird die Sequenzdatei nacheinander in den folgenden Verzeichnissen gesucht:

- Projektverzeichnis: Wenn ein Projekt geladen ist, wird danach im aktuellen Projektverzeichnis gesucht.
- Aktuelles Arbeitsverzeichnis: Dieses steht beim Aufstart von imc FAMOS zunächst auf dem unter "Optionen"/ "Verzeichnisse" vereinbarten Verzeichnis. Es kann mit dem Befehl MDIR oder der Funktion SetOption() umgesetzt werden. Ansonsten das Verzeichnis, aus dem die aufrufende Sequenz geladen wurde.

Nähere Informationen zum Arbeiten mit Sequenzen, insbesondere über Sequenzen mit Parametern, finden Sie im Bedienerhandbuch im Kapitel 'Sequenzen'.

- Die Zahl der Parameter ist auf maximal 20 beschränkt.
- Die Zahl der tatsächlich übergebenen Parameter kann innerhalb der Sequenz mit der Funktion ParametersPassed?() bestimmt werden.
- Wenn eine Sequenz aktiv ist, finden Sie im Info-Bereich der Windows-Taskleiste ein zusätzliches Symbol. Mit einem Rechtsklick erhalten Sie ein Kontextmenü, mit dem Sie die Ausführung abbrechen können.
- Zum Unterbrechen können Sie auch die Tastenkombination "Strg" + "UNTBR" verwenden.



Beispiele:

```
SEQUENCE Folge
```

Die Sequenz mit dem Dateinamen Folge.seq wird ausgeführt. Diese Sequenz erwartet keine Parameter.

```
SEQUENCE Summe Daten1 Daten2
```

Eine Sequenz mit dem Namen "Summe" wird ausgeführt. Sie berechnet die Summe aus zwei Datensätzen. Die beiden zu addierenden Datensätze werden als Parameter übergeben.

```
SEQUENCE c:\imc\seq\Ladeall.seq d*.dat
```

Eine Sequenz, mit der eine Datei geladen wird, wird ausgeführt für alle vorhandenen Dateien, die mit "D" beginnen und die Erweiterung ".DAT" haben.

```
SEQUENCE "c:\imc\Meine Sequenzen\Auswert.dat" Daten1
```

Der Pfadname für die Sequenz enthält Leerzeichen und muss darum in Anführungszeichen angegeben werden.

Siehe auch:

Dialog, SetOption, MDIR, ParametersPassed?

Set

Setzt Punkte eines Datensatzes an gegebenen x-Position(en) auf neue y-Wert(e).
Alternativer Name: **Setze**

Deklaration:

```
Set ( Daten, XPositionen, YWerte ) -> ErgebnisDaten
```

Parameter:

Daten	Zu ändernder Datensatz. Erlaubte Typen: [ND].
XPositionen	X-Koordinaten, zu denen die zugehörigen Y-Werte geändert werden sollen.
YWerte	Y-Werte, auf die der Datensatz an den angegebenen X-Koordinaten gesetzt werden soll.
ErgebnisDaten	Geänderter Datensatz mit entsprechend neuen Werten

Beschreibung:

Ein Zahlenwert eines Datensatzes kann mit dieser Funktion gezielt auf einen neuen Wert gesetzt werden.

Es sind anzugeben:

- der Datensatz, in dem ein Wert neu zu setzen ist,
- die x-Koordinate(n), an die der neue Wert einzutragen ist,
- die neue y-Koordinate(n), d. h. der/die neue Zahlenwert(e) selbst.

Es wird ein neuer Datensatz erzeugt, bei dem ein Reihe von Werten gegenüber dem übergebenen Datensatz verändert ist. Der 2. und 3. Parameter müssen die gleiche Länge aufweisen.

Die Einheiten des Datensatzes werden übernommen. Der 2. Parameter sollte sinnvollerweise die x-Einheit, der 3. die y-Einheit des Datensatzes aufweisen.

Die x-Position des zu setzenden Wertes wird nicht als Index, sondern als x-Koordinate angegeben.

Liegt die x-Position außerhalb der x-Ausdehnung des Datensatzes, wird kein Wert verändert.

Sie können alternativ mit der Funktion SetIndex() arbeiten, wenn Sie die zu ändernden Punkte über ihren Index im Datensatz angeben wollen. Diese Funktion ist auch für XY-Datensätze geeignet.

Wenn Sie Zahlenwerte von Hand in einem Datensatz verändern möchten, ohne den Vorgang automatisieren zu wollen, ist der Datensatz-Editor von imc FAMOS geeigneter.

Möchten Sie einzelnen Wert ändern, können Sie dies auch direkt über eine Zuweisung lösen:

```
NDData[ Index] = newSingleValue
```

Beispiele:

In einem Datensatz der Ausdehnung 0s ... 20s wird der Wert an der Stelle 5s auf 4.2A gesetzt:

```
NDdata = Set(NDdata, 5 's', 4.2 'A')
```

Der siebente Punkt im Datensatz wird auf den Wert 5 gesetzt. Die x-Koordinate des Punktes wird aus dem Index des Punktes innerhalb des Datensatzes und der x-Skalierung berechnet.

```
NDnew = Set(NDold, xOff?(NDold) + (7 - 1) * xDel?(NDold), 5)
; oder viel einfacher:
NDNew[7] = 5
```

Siehe auch:

SetIndex, Value2, ValueIndex, MatrixSet, MatrixFromLine, Repl, ReplIndex

SetBoxPos

Position für Ein- und Ausgabeboxen setzen

Deklaration:

```
SetBoxPos ( TxFktName, EwLinks, EwOben, EwBreite, EwHöhe, Null )
```

Parameter:

TxFktName	Name der Fenster-erzeugenden Funktion.
	"BoxValue?" : BoxValue?
	"BoxText?" : BoxText?
	"BoxOutput" : BoxOutput
	"BoxMessage" : BoxMessage
	"BoxVarSelector" : BoxVarSelector
	"DlgFileName" : DlgFileName
	"FsDlgSelectDirectory" : FsDlgSelectDirectory
EwLinks	x-Position der linken oberen Ecke der Box
EwOben	y-Position der linken oberen Ecke der Box
EwBreite	Breite der Box
EwHöhe	Höhe der Box
Null	Reservierter Parameter. Immer auf 0 zu setzen.

Beschreibung:

Setzt die Anzeigeposition für Boxen, die mit darauf folgenden Aufrufen von BoxValue?(), BoxText?(), BoxOutput(), BoxMessage() oder DlgFileName() erzeugt werden.

Die Positionsangaben beziehen sich auf Bildschirmpunkte. Bei einer Auflösung von 800x600 hat die linke obere Ecke des Bildschirms die Koordinaten (0,0) und die rechte untere Ecke die Koordinaten (799,599).

Boxen, die mittels BoxMessage() oder DlgFileName() erzeugt werden, haben immer eine feste, automatisch bestimmte Größe. [Höhe] und [Breite] werden hier ignoriert.

Alle Boxen besitzen eine Standardhöhe und -breite. Kleinere Werte von [Breite] und [Höhe] werden ignoriert.

Ggf. werden die Koordinaten so korrigiert, dass die Box komplett im sichtbaren Bereich liegt.

Wenn [Links] auf -1 gesetzt wird, wird die Position beim nächsten Aufruf wieder automatisch bestimmt.

Multithreading: Die Funktion wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

```
SetBoxPos("BoxMessage", 0, 0, 0, 0, 0)
BoxMessage("Result: ", res, "f20", 0)
```

Ausgabe links oben in der Bildschirmecke mit Standardbreite und -höhe.

```
SetBoxPos("BoxText?", 600, 450, 200, 150, 0)
TxName = BoxText?("Name of the variable?","var1",0)
```

Ausgabe rechts unten (bei Bildschirmauflösung von 800x600 Punkten).

Siehe auch:

BoxOutput, BoxMessage, BoxValue?, BoxText?, DlgFileName

SetColor

Zuweisen einer Farbe für die Kurvendarstellung eines Datensatzes
Alternativer Name: **SetFarbe**

Deklaration:

```
SetColor ( Daten, EwFarbWert )
```

Parameter:

Daten	Datensatz, dessen Farbattribut gesetzt werden soll.
EwFarbWert	Farbwert, -1 bedeutet automatische Farbgebung.

Beschreibung:

Normalerweise werden Datensätze im Kurvenfenster in der Farbe angezeigt, die am Kurvenfenster für die Darstellung eingestellt wurde. Sie können aber auch eine feste Farbe zuweisen, mit der der Datensatz dann in jedem Kurvenfenster unabhängig von aktuellen Einstellungen angezeigt wird.

Der Farbwert ist ein so genannter RGB-Wert, in dem die Anteile der 3 Grundfarben Rot, Grün und Blau vermerkt sind.

Zum Bilden eines Farbwertes aus den Farbanteilen können Sie die Funktion RGB() benutzen.

Beispiele:

```
green = RGB(0, 255, 0)
clr = Color?(data)
IF clr = -1
   SetColor(data, green)
END
```

Falls dem Datensatz bisher keine feste Farbe zugewiesen worden ist, wird er in Zukunft immer Grün dargestellt.

Siehe auch:

Color?, RGB

SetComm

Setzen des Kommentars zu einem Datensatz, Text oder Datengruppe.
Alternativer Name: **SetKomm**

Deklaration:

```
SetComm ( Datenobjekt, TxKommentar )
```

Parameter:

Datenobjekt	Datensatz, Text oder Datengruppe, dessen Kommentar gesetzt werden soll.
TxKommentar	Neuer Kommentar

Beschreibung:

Beispiele:

Der Kommentar eines Datensatzes wird abgefragt. Falls er nicht gesetzt ist (Länge 0), wird der Nutzer aufgefordert, einen Kommentar einzugeben. Dieser wird dem Datensatz dann zugewiesen.

```
txComment = Comm?(data)
IF TLeng(txComment) = 0
   txComment = BoxText?("Kommentar eingeben:", "",0)
   SetComm(data, txComment)
END
```

Siehe auch:

Comm?

SetDataFormat

Ein Datensatz wird in ein anderes Datenformat konvertiert.
Alternativer Name: **SetDatFormat**

Deklaration:

```
SetDataFormat ( Daten, EwFormatCode [, EwSkalMin] [, EwSkalMax] )
```

Parameter:

Daten	Datensatz, dessen Datenformat gesetzt werden soll
EwFormatCode	Formatangabe
	0 : 4 Byte reell (float)
	1 : 8 Byte reell (double)
	2 : 1 Byte ganzzahlig
	3 : 2 Byte ganzzahlig
	4 : 4 Byte ganzzahlig
	5 : 1 Byte ganzzahlig ohne Vorzeichen
	6 : 2 Byte ganzzahlig ohne Vorzeichen
	7 : 4 Byte ganzzahlig ohne Vorzeichen
	8 : Digital
	10 : 6 Byte ganzzahlig ohne Vorzeichen
	12 : 8 Byte ganzzahlig
	13 : 8 Byte ganzzahlig ohne Vorzeichen
EwSkalMin	Bei ganzzahligen Datenformaten die untere Grenze des Wertebereiches, sonst egal. (optional , Standardwert: 0)
EwSkalMax	Bei ganzzahligen Datenformaten die obere Grenze des Wertebereiches, sonst egal. Wenn [EwSkalMin] und [EwSkalMax] identisch sind, wird automatisch skaliert. Wenn die optionalen Parameter [EwSkalMin] und [EwSkalMax] beide nicht angegeben sind, wird der komplette Zahlenbereich des ganzzahligen Datenformats angenommen. (optional , Standardwert: 0)

Beschreibung:

Die Funktion konvertiert einen Datensatz in ein neues Datenformat.

Das Datenformat gibt an, wie die einzelnen Werte im Speicher bzw. auf dem Datenträger abgelegt werden. Der Speicherplatzbedarf eines Datensatzes, der Wertebereich und die erreichbare Genauigkeit werden durch das Datenformat bestimmt.

Wenn dieser Funktion ein XY-Datensatz übergeben wird, wird das Datenformat der Y-Komponente gesetzt. Wenn ein komplexer Datensatz übergeben wird, wird das Datenformates des Betrages bzw. des Realteiles gesetzt. Um die jeweils andere Komponente zu konvertieren, können die Komponentenkennungen benutzt werden:

```
SetDataFormat(BetragPhase.P, 0, 0, 0)
SetDataFormat(XYSatz.X, 5, 0, 255)
```

Das mit dieser Funktion eingestellte Datenformat kann durch nachfolgende Berechnungen mit diesem Datensatz wieder geändert werden. Wenn Sie dies verhindern und das Format dauerhaft festschreiben möchten, müssen Sie die "Datenformat fixiert"-Eigenschaft mit der Funktion SetFlag() einschalten.

Besonderheiten bei ganzzahligen Datenformaten:

Für ganzzahlige Datenformate werden intern zusätzlich ein Skalierungsfaktor und ein Offset vermerkt, um aus den ganzzahligen Rohdaten die physikalischen Werte zu bestimmen:

```
PhysicalValue = IntegerValue * ScaleFactor + Offset
```

Für vorzeichenbehaftete Formate mit der Bitbreite [BitCount] werden die Kenngrößen [ScaleFactor] und [Offset] wie folgt berechnet:

```
Offset = (EwSkalMin+EwSkalMax)/2
ScaleFactor = (EwSkalMax-Offset) / (2^(BitCount-1)-1)
```

Für eine 1:1-Abbildung des gesamten möglichen Zahlenbereichs ist also [EwSkalMin] = -2^(BitCount-1)-1 und [EwSkalMax] = 2^(BitCount-1)-1 anzugeben.

Für Formate ohne Vorzeichen gilt:

```
Offset = EwSkalMin
```

```
ScaleFactor = (EwSkalMax-Offset) / (2^BitCount-1)
```

Für eine 1:1-Abbildung des gesamten möglichen Zahlenbereichs ist also [EwSkalMin] = 0 und [EwSkalMax] = 2^BitCount-1 anzugeben.

Beispiel:

Format = 2 (1Byte mit VZ), EwSkalMin= -127, EwSkalMax = 127 ==> Offset = 0, ScaleFactor = 1

Format = 5 (1Byte ohne VZ), EwSkalMin= -128, EwSkalMax = 127 ==> Offset = -128, ScaleFactor = 1

Wenn die optionalen Parameter [EwSkalMin] und [EwSkalMax] weggelassen werden, wird bei ganzzahligen Formaten eine 1:1 Abbildung (ScaleFactor=1, Offset=0) realisiert.

Beispiele:

```
format = DataFormat?(myData)
IF format <> 0
   SetDataFormat(myData, 1)
   SetFlag(myData, 0, 1)
END
```

Falls der Datensatz nicht bereits im reellen 4-Byte-Format vorliegt, wird er in dieses konvertiert. Anschließend wird dieses fixiert, um sicherzustellen, dass es durch nachfolgende Berechnungen nicht geändert wird.

Beispiele für Skalierung bei ganzzahligen Formaten:

```
SetDataFormat(myData, 2)            ;Wertebereich -128..127 (Skalierungsfaktor = 1, Offset = 0)
SetDataFormat(myData, 2, -127, 127) ;Wertebereich -128..127 (Skalierungsfaktor = 1, Offset = 0)
SetDataFormat(myData, 2, 0, 10)     ;Wertebereich 0..10     (Skalierungsfaktor = 5/127, Offset = 5)
SetDataFormat(myData, 5)            ;Wertebereich 0..255    (Skalierungsfaktor = 1, Offset = 0)
SetDataFormat(myData, 5, 0, 10)     ;Wertebereich 0..10     (Skalierungsfaktor = 10/255, Offset = 0)
```

Siehe auch:

DataFormat?, GetScale, SetFlag

SetDisplayY

Zuweisung einer festen Y-Skalierung für die Anzeige im Kurvenfenster
Alternativer Name: **SetAnzeigeY**

Deklaration:

```
SetDisplayY ( Daten, EwMin, EwMax )
```

Parameter:

Daten	Datensatz, dessen Y-Skalierung gesetzt werden soll.
EwMin	Unterer Skalenwert
EwMax	Oberer Skalenwert

Beschreibung:

Einem Datensatz kann als zusätzliche Eigenschaft eine feste Skalierung der y-Achse zugewiesen werden. Diese wird dann bei der Anzeige des Datensatzes in einem Kurvenfenster benutzt, wenn die Einstellung "Automatische Skalierung der Y-Achse" am Kurvenfenster gesetzt ist.

Mit EwMin = EwMax = 0 wird die feste Skalierung wieder aufgehoben. Der Datensatz wird dann wieder automatisch entsprechend seinem Wertebereich skaliert.

Beispiele:

```
yMin = DisplayY?(data, 0)
IF yMin > 0
   SetDisplayY(data, 0, DisplayY?(data, 1))
ELSE
   yMax = DisplayY?(data, 1)
   IF yMax < 0
      SetDisplayY(data, DisplayY?(data, 0),0)
   END
END
```

Falls dem Datensatz eine feste Skalierung zugewiesen ist und die Null-Linie nicht enthalten ist, wird der untere oder obere Skalenwert auf 0 korrigiert.

Siehe auch:

DisplayY?, SetColor

SetFlag

Die Funktion schaltet spezielle Datensatz-Attribute an bzw. aus.

Deklaration:

```
SetFlag ( Daten, EwFlag, EwAnOderAus )
```

Parameter:

Daten	Datensatz, dessen Attribut geändert werden soll.
EwFlag	Auswahl des Attributes
	0 : Das aktuelle Datenformat des Datensatzes wird fixiert, d.h. bei nachfolgenden Berechnungen wird nach Möglichkeit das Datenformat (und bei Integer-Formaten die Skalierungsinformation) beibehalten.
	1 : Die Werte des Datensatzes können als Farbinformation für 1 Pixel eines Bildes interpretiert werden. Nur erlaubt für die Datenformate '4 Byte ohne Vorzeichen' (Farbinformation ist im RGB-Format codiert) oder '1 Byte ohne Vorzeichen' (Farbinformation ist als Graustufenwert in Bereich 0..255 codiert). Das Attribut wird standardmäßig nur von Importfiltern für Bilddateien oder Spezialfunktionen wie VpGetImages() gesetzt. Es wird vom Kurvenfenster verwendet, um die Anzeige von Bilddaten zu optimieren.
EwAnOderAus	Neuer Wert
	0 : Aus
	1 : An

Beschreibung:

Die Funktion setzt gewisse Datensatz-Attribute, die lediglich die boolschen Werte [An] (bzw. "Wahr") oder [Aus] (bzw. "Falsch") annehmen können.

Das Einschalten der Option [EwFlag] = 1 (als Bild-Daten interpretieren) ist nur für Datensätze mit dem Datenformat "4 Byte ganzzahlig ohne Vorzeichen" oder "1 Byte ganzzahlig ohne Vorzeichen" erlaubt. Bei segmentierten Datensätzen entspricht jedes Segment einer Bildzeile, das erste Sample des Datensatzes entspricht dem Bildpunkt links unten.

Beispiele:

Das Datenformat des Datensatzes [Signal] wird auf 1-Byte Integer (ohne Vorzeichen, Wertebereich 0..255) eingestellt und fixiert. Nachfolgende Berechnungen (siehe Zeile [*]) ändern dann das Datenformat nicht. Ohne den SetFlag()-Aufruf würde das Datenformat von [Signal] nach Zeile [*] wieder in 4- oder 8-Byte reell gewandelt werden (abhängig von der globalen Voreinstellung für das Ergebnisdatenformat mathematischer Funktionen).

```
signal = ...
SetDataFormat(signal, 5, 0, 255)
SetFlag(signal, 0, 1)
;...
signal = signal/2  ;[*]
```

Siehe auch:

Flag?, RGB, VpGetImages, SetDataFormat

SetIndex

Setzt Punkte in einem Datensatz an gegebenen Indizes auf neue y-Werte
Alternativer Name: **SetzeIndex**

Deklaration:

```
SetIndex ( Daten, Indizes, YWerte ) -> ErgebnisDaten
```

Parameter:

Daten	Zu ändernder Datensatz. Erlaubte Typen: [ND],[XY].
Indizes	Indizes, zu denen die zugehörigen Y-Werte geändert werden sollen..
YWerte	Y-Werte, auf die der Datensatz an den angegebenen Positionen gesetzt werden soll.
ErgebnisDaten	Geänderter Datensatz mit entsprechend neuen Werten

Beschreibung:

Setzt einen oder mehrere Punkte eines Datensatzes an vorgegeben Indizes auf neue y-Werte.

Die in [Indizes] angegebenen Positionen (Indizes) müssen zwischen 1 und der Datensatzlänge von [Daten] liegen.

Außerhalb dieses Bereichs liegende Indizes werden ignoriert (Warnung wird erzeugt).

Es wird ein neuer Datensatz erzeugt, bei dem ein Reihe von Werten gegenüber dem übergebenen Datensatz verändert ist. Der 2. und 3. Parameter müssen die gleiche Länge aufweisen.

Der erste Punkt in einem Datensatz hat den Index 1, der Index des letzten Punktes entspricht der Datensatzlänge.

Alternativ kann die Funktion Set() verwendet werden, bei der die Positionen durch die x-Koordinaten der Punkte definiert werden.

Wenn Sie Zahlenwerte von Hand in einem Datensatz verändern möchten, ohne den Vorgang automatisieren zu wollen, ist der Datensatz-Editor von imc FAMOS geeigneter.

Möchten Sie in FAMOS einen einzelnen Wert ändern, können Sie dies auch direkt über eine Zuweisung lösen:

```
NdDaten[ Index] = EwNeuerWert
```

Beispiele:

Der zweite Wert des Datensatzes wird auf den Wert 10 gesetzt:

```
data = SetIndex(data, 2 , 10)
; äquivalent zu:
data[2] = 10
```

Jeder zweite Y-Wert des Datensatzes wird verdoppelt:

```
Indizes = Ramp(1, 2, Leng?(Data)/2) ;Indizes = 1,3,5..
NewY= ValueIndex(Data, Indizes) * 2
DataNew = SetIndex(Data, Indizes, NewY)
```

Siehe auch:

Set, Value2, ValueIndex, MatrixSet, MatrixFromLine, Repl, ReplIndex

SetMeasurementName

Setzen des Namens der Messung, der die Variable zugeordnet ist.

Deklaration:

```
SetMeasurementName ( Variable, TxMessungsName )
```

Parameter:

Variable	Zu ändernde Variable.
TxMessungsName	Name der Messung, der die Variable zugeordnet werden soll.

Beschreibung:

Ein leerer Messungsname bedeutet, dass eine etwaige Messungszugehörigkeit gelöscht wird.

Das Konzept der Messungszugehörigkeit einer Variablen findet bisher hauptsächlich im Datenquellen-Browser Anwendung. Dort wird beim Laden einer Messwert-Datei jeder erzeugten Variablen automatisch ein Messungsname zugeordnet, um gleichnamige Variablen, die aus verschiedenen Datenquellen stammen, bequem unterscheiden zu können. Das Ändern der Messungszugehörigkeit einer Variablen löst automatisch die Aktualisierung von Messungs- und Kanalliste in der Variablenliste/Messungsansicht aus.

Seit FAMOS 7.1 kann bei der Erzeugung einer Variablen mittels Zuweisung der gewünschte Messungsname auch direkt angegeben werden (siehe Beispiele).

Beispiele:

Nachdem mit dem Datenquellen-Browser eine neue Messung angelegt wurde, wird geprüft, ob diese einen Kanal mit dem Namen "speed" enthält. Falls ja, wird das Maximum dieses Kanals berechnet und das Ergebnis mit dem aktuellen Messungsnamen markiert.

Ereignis-Sequenz 'Messung verfügbar'

```
TxVarName = SelBuildVarName(PA1, "speed", 0)
IF TxVarName <> ""
   MaxSpeed = max(<TxVarName>)
   SetMeasurementName(MaxSpeed, PA1)
END
Oder einfacher:
```

```
...
IF TxVarName <> ""
   MaxSpeed@<PA1> = max(<TxVarName>)
END
```

Die aktuell bearbeiteten Messungen enthalten die Kanäle 'Voltage' und 'Current'. Zur aktuell selektierten Messung #1 wird die Leistung berechnet und mit dem entsprechenden Messungsnamen versehen. Damit wird die berechnete Variable automatisch auch in der Variablenliste/Messungsansicht in der Kanalliste eingeblendet.

```
IF SelUseMeasurement(1) = 0
   EXITSEQUENCE
END
TxMeasName = MeasurementName?(Voltage)
Power = Voltage * Current
SetMeasurementName(Power, TxMeasName)
```

Oder einfacher:

```
IF SelUseMeasurement(1) = 0
   EXITSEQUENCE
END
Power@ = Voltage * Current
```

Siehe auch:

MeasurementName?

SetOption

Festlegen diverser Voreinstellungen , wie z.B. Standardverzeichnisse und Vorgaben für mathematische Funktionen.

Deklaration:

```
SetOption ( TxOptionsName, TxNeueEinstellung ) -> AlteEinstellung
```

Parameter:

TxOptionsName	Bezeichnung der Option
	"Dir.DataFiles" : Standardverzeichnis zum Laden und Speichern von Meßwert-Dateien. Verwendet von den Befehlen LOAD, SAVE u.ä. sowie den Funktionen FileOpenDSF() und FileOpenFAS().
	"Dir.Sequences" : Standardverzeichnis zum Laden von Sequenzen. Verwendet vom dem Befehl SEQUENCE.
	"Func.WarnLevel" : Legt fest, welche Warnungen, die bei der Ausführung von Funktionen auftreten, angezeigt werden sollen
	"Func.NoInfoMessages" : Einige Funktionen (Stat, LFit) geben standardmäßig ihre Ergebnisse im Ausgabefenster aus. Mit dieser Option kann diese Ausgabe unterbunden werden.
	"Func.FFT.Window" : Legt die FFT Fensterfunktion fest. Verwendet von den Funktionen FFT(), Spec() u.ä.
	"Func.FFT.Mode" : Der implementierte FFT-Algorithmus verlangt, dass die Länge des Ausgangsdatensatzes eine 2er-Potenz ist. Diese Option legt fest, wie bei anderen Datensatzlängen verfahren wird.
	"Func.ResultFormat" : Legt das Datenformat fest, in dem Funktionen standardmäßig Ihre Ergebnisse zurückliefern.
	"Func.ErrorBoxes" : Einige Funktionen können im Fehlerfall wahlweise entweder eine Fehlerbox erzeugen oder den Fehler (still) durch Ihren Rückgabewert signalisieren. Betrifft bisher nur die Gruppe der Dateifunktionen FileOpenDSF() u.ä.
	"DLLImport.DefinitionFile" : Dateiname mit Definitionen für externe DLL-Funktionen. Die hier gelisteten Funktionen werden über die allgemeine DLL-Schnittstelle in FAMOS importiert und können in Sequenzen verwendet werden. Die Datei muss über den Dialog 'Optionen'/'DLL-Funktionen registrieren' erzeugt worden sein. Wenn kein kompletter Pfad angegeben ist, wird die Datei nacheinander in den folgenden Verzeichnissen gesucht: Projektverzeichnis (bei aktivem Projekt) - aktuelles Sequenzverzeichnis - Standardverzeichnis für Definitions-Dateien (siehe 'Optionen'/'Verzeichnisse'). Wenn keine Dateierweiterung angegeben ist, wird ".def" angenommen.
	"Display.DecimalSeparator" : Legt den Dezimaltrenner für die Anzeige reeller Zahlen fest.
	"DDE.Text.NumFormat" : Legt das Zahlenformat beim DDE-Senden im Text-Format fest.
	"DDE.Text.Delimeter" : Legt das (die) Trennzeichen fest, welche(s) beim DDE-Senden im Text-Format zwischen 2 Zahlen verwendet wird.
	"DDE.TimeOut" : Legt die maximale Zeit fest, die FAMOS bei der DDE-Kommunikation mit einer anderen Applikation auf eine Antwort wartet.
	"Units.Ctrl.Compatible" : Einheiten: Kompatibilität
	"Units.Display.Greek" : Einheiten: Schriftart für griechische Buchstaben
	"Units.Display.Ohm" : Einheiten: Ohm
	"Units.Create.Delim" : Einheiten: Trennzeichen zwischen Einheiten, z.B. V*A
	"Units.Create.Nm" : Einheiten: Trennzeichen bei Nm u.ä.
	"Units.Create.Pow.1/2" : Einheiten: Sonderzeichen für 1/2 nutzen
	"Units.Create.Pow.2" : Einheiten: Sonderzeichen für ^2 nutzen
	"Units.Create.Pow.3" : Einheiten: Sonderzeichen für ^3 nutzen
	"Units.Create.Pow.Neg" : Einheiten: Negative Potenzen im Nenner
	"Units.Create.u" : Einheiten: u anstelle von µ erzeugen (Präfix 10^-6)
	"Units.Create.Num.Space" : Einheiten: Leerzeichen zwischen automatisch ergänzter 10er-Potenz und Einheit
	"Units.Create.1e3" : Einheiten: 10, 100, 1000
	"Units.Create.1e-3" : Einheiten: 0.1, 0.01, 0.001
	"Units.Create./s" : Einheiten: Einheit mit Zähler = 1
	"Units.Read.cal" : Einheiten: Einheitenzeichen cal
	"Units.Read.Exp" : Einheiten: Zahl als Exponent

	"Units.Read.g" : Einheiten: g
	"Units.Read.Gs" : Einheiten: Einheitenzeichen für Gauss
	"Units.Read.hp" : Einheiten: Einheitenzeichen hp
	"Units.Read.kt" : Einheiten: Einheitenzeichen kt
	"Units.Read.L" : Einheiten: Einheitenzeichen L
	"Units.Read.lb" : Einheiten: Einheitenzeichen lb
	"Units.Read.oz" : Einheiten: Einheitenzeichen oz
	"Units.Read.pt" : Einheiten: Einheitenzeichen pt
	"Units.Read.quot" : Einheiten: Einheiten in Anführungszeichen, z.B. "Clocks"
	"Units.Read.s" : Einheiten: Mehrzahl s erlaubt
	"Units.Read.ton" : Einheiten: Einheitenzeichen ton
	"Units.Read.u" : Einheiten: Einheitenzeichen u
	"Units.Reduce.C" : Einheiten: C (Coulomb) erzeugen
	"Units.Reduce.F" : Einheiten: F (Farad) erzeugen
	"Units.Reduce.H" : Einheiten: H (Henry) erzeugen
	"Units.Reduce.J" : Einheiten: J (Joule) erzeugen
	"Units.Reduce.Ohm" : Einheiten: Ohm erzeugen
	"Units.Reduce.Pa" : Einheiten: Pa (Pascal) erzeugen
	"Units.Reduce.S" : Einheiten: S (Siemens) erzeugen
	"Units.Reduce.T" : Einheiten: T (Tesla) erzeugen
	"Units.Reduce.Wb" : Einheiten: Wb (Weber) erzeugen
TxNeueEinstellung	[TxOptionsName] bestimmt die möglichen Vorgaben:
	"Func.FFT.Window" : FFT Fensterfunktion
	"Rectangle" Rechteckfenster. **"Hamming"** Hamming Fenster **"Hanning"** Hanning Fenster **"Blackman"** Blackman Fenster **"Blackman_Harris"** Blackman/Harris Fenster **"Flat_Top"** Flat-Top Fenster
	"Func.FFT.Mode" : FFT Modus festlegen (wenn Datensatzlänge ungleich 2er-Potenz)
	"Truncate" Der Datensatz wird auf die nächstkleinere 2er-Potenz eingekürzt. **"AppendZeroes"** Der Datensatz wird auf die nächsthöhere 2er-Potenz mit Nullen aufgefüllt.
	"Func.ResultFormat" : Standard für Ergebnis-Datenformat festlegen
	"Auto" Automatisch **"Float"** 4 Byte reell **"Double"** 8 Byte reell
	"Func.WarnLevel" : Welche Warnungen anzeigen?
	"None" Keine Warnungen **"Important"** Wichtige Warnungen **"All"** Alle Warnungen
	"Func.ErrorBoxes" : Fehlerboxen (Dateifunktionen)

	"Yes" Der Fehler wird angezeigt und die Sequenzabarbeitung abgebrochen. **"No"** Wenn möglich, wird ein Fehlercode zurückgegeben. Der Aufrufer ist für die Auswerung des Rückgabewertes selbst verantwortlich.
	"Func.NoInfoMessages" : Informations-Ausgaben
	"No" Die Ergebnisse von Funktionen wie Stat() oder LFit() werden im Ausgabefenster angezeigt. **"Yes"** Keine Ausgabe von Ergebnissen.
	"Display.DecimalSeparator" : Dezimaltrenner
	"." Punkt **","** Komma **"region"** Windows-Einstellung (Systemsteuerung / Region).
	"Units.Ctrl.Compatible" : Kompatibilität
	"no" nein **"7.0"** Wie imc FAMOS 7.0 und Vorgänger
	"Units.Display.Greek" : Schriftart für griechische Buchstaben
	"keep" Griechischer Zeichensatz bei gewählter Schriftart **"symbol"** Schriftart 'Symbol' nutzen
	"Units.Display.Ohm" : Ohm
	"Omega" Griech. Buchstaben Omega bevorzugen **"Ohm"** Stets als Zeichenfolge Ohm anzeigen
	"Units.Create.Delim" : Trennzeichen zwischen Einheiten, z.B. V*A
	"dot" Punkt **"*"** * **"blank"** Leerzeichen
	"Units.Create.Nm" : Trennzeichen bei Nm u.ä.
	"no" nein (z.B. Nm) **"yes"** ja (z.B. N*m)
	"Units.Create.Pow.1/2" : Sonderzeichen für 1/2 nutzen
	"yes" ja **"no"** nein, stets ^1/2 ausschreiben
	"Units.Create.Pow.2" : Sonderzeichen für ^2 nutzen
	"yes" ja **"no"** nein, stets ^2 ausschreiben
	"Units.Create.Pow.3" : Sonderzeichen für ^3 nutzen
	"yes" ja **"no"** nein, stets ^3 ausschreiben
	"Units.Create.Pow.Neg" : Negative Potenzen im Nenner
	"1/m" ja (z.B. 1/m) **"m^-1"** nein (z.B. m^-1)
	"Units.Create.u" : u anstelle von µ erzeugen (Präfix 10^-6)

	"no" nein **"yes"** ja
	"Units.Create.Num.Space" : Leerzeichen zwischen automatisch ergänzter 10er-Potenz und Einheit
	"yes" ja, z.B. 10^3 V **"no"** nein, z.B. 10^3V
	"Units.Create.1e3" : 10, 100, 1000
	"e3" 10^3 etc. bevorzugen **"1000"** 1000 etc. bevorzugen
	"Units.Create.1e-3" : 0.1, 0.01, 0.001
	"e-3" 10^-3 etc. bevorzugen **".001"** 0.001 etc. bevorzugen
	"Units.Create./s" : Einheit mit Zähler = 1
	"default" regulär (z.B. 1/m) **"no1"** 1 weglassen (z.B. /m)
	"Units.Read.cal" : Einheitenzeichen cal
	"internat" International 1 cal = 4.1868 J **"thermo"** Thermochemisch 1 cal = 4.184 J
	"Units.Read.Exp" : Zahl als Exponent
	"no" Nein, z.B. "g CO2" erhalten **"try"** In klaren Anordnungen deuten, z.B. "m2/s2" als "m^2/s^2"
	"Units.Read.g" : g
	"gram" Gramm **"gravity"** Erdbeschleuninigung 9.81 m/s^2
	"Units.Read.Gs" : Einheitenzeichen für Gauss
	"Gs" Gs (Standard) **"G"** G
	"Units.Read.hp" : Einheitenzeichen hp
	"NIST" mechanical horsepower, NIST, 745.6999 W **"metric"** metric horsepower, 735.49875 W = 1 PS **"electric"** electric horsepower, 746 W **"hydraulic"** hydraulic horsepower, 745.69988145 W
	"Units.Read.kt" : Einheitenzeichen kt
	"kiloton" Kilotonne **"knot"** Knoten
	"Units.Read.L" : Einheitenzeichen L
	"Liter" Liter **"Lambert"** Lambert
	"Units.Read.lb" : Einheitenzeichen lb

	"Force" Kraft, 1 lb = 1 lbs = 4.4 N = 1 lbf **"Mass"** Masse, 1 lb = 1 lbs = 0.45 kg = 1 lbm
	"Units.Read.oz" : Einheitenzeichen oz
	"Force" Kraft, 1 oz = 0.28 N **"Mass"** Masse, 1 oz = 0.028 kg
	"Units.Read.pt" : Einheitenzeichen pt
	"pintUS" United States liquid pint, 1 pt = 473.1765 ml **"pintUK"** Imperial pint, UK, 1 pt = 568.26125 ml **"pintDry"** United States dry pint, 1 pt = 550.6104713575 ml **"point"** printer point, 1 pt = 0.35146e mm
	"Units.Read.quot" : Einheiten in Anführungszeichen, z.B. "Clocks"
	"no" Inhalt nicht deuten **"try"** Inhalt versuchen zu deuten, z.B. "N" als N
	"Units.Read.s" : Mehrzahl s erlaubt
	"no" nein (z.B. Volts = V * s) **"yes"** ja (z.B. Volts = V)
	"Units.Read.ton" : Einheitenzeichen ton
	"short" ton (short) (2000 lb), US **"long"** ton (long) (2240 lb), UK
	"Units.Read.u" : Einheitenzeichen u
	"atom" Atomare Masseneinheit **"mju"** µ (Präfix 10^-6)
	"Units.Reduce.C" : C (Coulomb) erzeugen
	"no" nein **"yes"** ja
	"Units.Reduce.F" : F (Farad) erzeugen
	"no" nein **"yes"** ja
	"Units.Reduce.H" : H (Henry) erzeugen
	"no" nein **"yes"** ja
	"Units.Reduce.J" : J (Joule) erzeugen
	"no" nein **"yes"** ja
	"Units.Reduce.Ohm" : Ohm erzeugen
	"yes" ja **"no"** nein
	"Units.Reduce.Pa" : Pa (Pascal) erzeugen

	"no" nein "yes" ja
	"Units.Reduce.S" : S (Siemens) erzeugen
	"no" nein "yes" ja
	"Units.Reduce.T" : T (Tesla) erzeugen
	"no" nein "yes" ja
	"Units.Reduce.Wb" : Wb (Weber) erzeugen
	"no" nein "yes" ja
AlteEinstellung	Vorheriger Wert der geänderten Option.

Beschreibung:

Diese Prozedur gestattet das Ändern diverser Voreinstellungen, die bei der Ausführung von Befehlen und mathematischen Funktionen berücksichtigt werden.

Alle hier einstellbaren Optionen sind nach dem FAMOS-Start mit den Vorgaben initialisiert, wie sie durch die Dialoge "Optionen/Verzeichnisse", "Optionen/DDE" bzw. "Optionen/Funktionen" eingestellt sind.

Diese Voreinstellung kann durch den Aufruf dieser Funktion innerhalb von Sequenzen geändert werden.

Die hier gesetzte Option bleibt gültig, bis entweder FAMOS neu gestartet wurde oder der entsprechende Options-Dialog aufgerufen und mit [OK] verlassen worden ist, was ein Rücksetzen aller Einstellungen auf die aktuellen Dialogwerte bewirkt.

Diese Funktion ist seit Version 4.0 enthalten. Einige der hier änderbaren Einstellungen können auch durch 'alte' Befehle wie FFTOPTION, SDIR, MDIR.. beeinflusst werden. Die Funktion SetOption() hat jedoch, abgesehen von Ihrer leichteren Handhabbarkeit (insbesondere bei Verwendung des Funktions-Assistenten), den Vorteil, dass der alten Zustand zurückgeliefert wird. Damit ist es leicht möglich, durchgeführte Änderungen am Ende einer Sequenz wieder rückgängig zu machen. Dies ist eine sehr empfehlenswerte Vorgehensweise, da dadurch Seiteneffekte vermieden werden. Für neu zu erstellende Sequenzen ist darum im allgemeinen SetOption() vorzuziehen.

Zusätzlich zu den oben genannten Werten für den 2.Parameter kann auch "Reset" angegeben werden. Mit dieser Anweisung wird die FAMOS-Standardvorgabe (wie im Dialog "Optionen" vorgegeben) für die jeweilige Option (wieder) aktiviert.

Multithreading: Die Funktion wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

Das folgende Sequenzfragment restauriert zuerst das in den Voreinstellungen (Dialog) vorgegebene Standard-Sequenzverzeichnis. Dann wird das Unterverzeichnis "Test1" des voreingestellten Standard-Datenverzeichnisses als neues Datenverzeichnis festgelegt. Danach wird in einer Schleife versucht, die Dateien "experim1.dat" bis "experim10.dat" zu laden. Falls eine Datei nicht vorhanden ist, wird keine Fehlermeldung erzeugt (Fehlerboxen wurden vorher ausgeschaltet) und mit der nächsten Datei fortgesetzt. Nach dem erfolgreichen Laden wird eine Sequenz aus dem Standard-Sequenzverzeichnis zur Auswertung aufgerufen.

```
SetOption("Dir.Sequences", "Reset")
StdFolder = GetOption("Dir.DataFiles")
SetOption("Dir.DataFiles", StdFolder + "\test1")
SetOption("Func.ErrorBoxes", "No")
i = 1
WHILE i <= 10
   fh = FileOpenDSF("experim"+ TForm(i, ""), 0)
   IF fh > 0
      data = FileObjRead(fh, 1)
      SHOW data
      SEQUENCE Auswertung.seq data
      ; ... weitere Auswertungen
      err = FileClose(fh)
   END
   i = i+1
END
```

Die folgende Untersequenz führt eine FFT auf den übergebenen Datensatz aus. Der Datensatz wird dabei auf die vorhergehende 2er-Potenz abgeschnitten, als Fensterung wird [Blackman] verwendet. Das Ergebnis wird im Double-Format (8 Byte reell) erzeugt. Warnungen werden unterdrückt. Am Ende der Sequenz werden die alten Einstellungen restauriert.

```
; Sequenz DoFFT
; Aufruf mit "SEQUENCE DoFFT Datensatz"
oldWindow = SetOption("Func.FFT.Window", "Blackman")
oldMode   = SetOption("Func.FFT.Mode", "Truncate")
oldFormat = SetOption("Func.ResultFormat", "Double")
oldLevel  = SetOption("Func.WarnLevel", "None")
Result = FFT(PA1)
...; Weitere Berechnungen
SetOption("Func.FFT.Window", oldWindow)
SetOption("Func.FFT.Mode", oldMode)
SetOption("Func.ResultFormat", oldFormat)
SetOption("Func.WarnLevel", oldLevel)
```

Daten werden nach EXCEL mittels DDE übertragen. Das Zahlenformat wird auf Festkommadarstellung mit Dezimalkomma und 6 Nachkommastellen festgelegt. Zuerst wird eine Tabellenspalte gefüllt (Trennzeichen ASCII-13+ ASCII-10 = Zeilenumbruch). Anschließend wird das Trennzeichen auf Tabulator gesetzt (ASCII-9) und eine Tabellenzeile gefüllt.

```
ColData = ...
RowData = ...
SetOption("DDE.Text.NumFormat", "f1.6")
SetOption("DDE.Text.Delimeter", "~013~010")
result = DDESet("EXCEL", "Mappe1", "Z2S1:Z100S1", ColData, 1)
SetOption("DDE.Text.Delimeter", "~009")
result = DDESet("EXCEL", "Mappe1", "Z1S1:Z1S100", RowData, 1)
```

Siehe auch:

GetOption, FFTOPTION, MDIR, LDIR, SDIR

SetSegLen

Die Segmentlänge eines Datensatzes wird gesetzt.
Alternativer Name: **SetSegLang**

Deklaration:

```
SetSegLen ( Daten, EwSegLang )
```

Parameter:

Daten	Datensatz, dessen Segmentlänge gesetzt werden soll.
EwSegLang	Neue Segmentlänge, 0 bedeutet keine Segmentierung.

Beschreibung:

Die Segmentlänge für einen Datensatz wird gesetzt. Der Datensatz wird auf ein Vielfaches der Segmentlänge mit Nullen aufgefüllt (bei skalierbaren ganzzahligen Datenformaten wird der unskalierte Zahlenwert auf 0 gesetzt).

Beispiele:

```
IF SegLen?(data) > 0
   SetSegLen(data, 0)
END
```

Eine eventuelle Segmentierung des Datensatzes wird aufgehoben.

Siehe auch:

SegLen?, MatrixInit, MatrixChangeDim

SetTime

Die Triggerzeit eines Datensatzes wird gesetzt.
Alternativer Name: **SetZeit**

Deklaration:

```
SetTime ( Daten, EwZeit )
```

Parameter:

Daten	Datensatz, dessen Triggerzeit zu setzen ist.
EwZeit	Neue Triggerzeit

Beschreibung:

Die Triggerzeit eines Datensatzes wird gesetzt. Der übergebene Zeitwert muss im imc FAMOS-Zeitformat vorliegen, wie ihn z. B. Funktionen wie Time?(), TimeSystem?() und TimeJoin() erzeugen.

Beispiele:

Einem Datensatz wird die Triggerzeit eines anderen Datensatzes zugewiesen:

```
time = Time?(dataA)
SetTime(dataB, time)
```

Siehe auch:

Time?, TimeJoin, TimeSystem?, TimeSplit

SetUnit

Eine Einheit eines Datensatzes wird gesetzt.
Alternativer Name: **SetEinheit**

Deklaration:

```
SetUnit ( Daten, TxEinheit, EwCode )
```

Parameter:

Daten	Datensatz, in dem eine Einheit verändert werden soll.
TxEinheit	Neue Einheit
EwCode	Angabe, welche Einheit gesetzt werden soll.
	0 : X-Einheit bei einkomponentigen Daten. Einheit der X-Komponente bei XY-Daten. Einheit der Phase bzw. des Imaginärteils bei komplexen Daten.
	1 : Y-Einheit bei einkomponentigen Daten. Einheit der Y-Komponente bei XY-Daten. Einheit des Betrages bzw. des Realteils bei komplexen Daten.
	2 : Z-Einheit
	3 : Einheit des Parameters bei 2-komponentigen Daten

Beschreibung:

Beispiele:

Die Y-Einheit eines Datensatzes wird abgefragt. Falls die Einheit "W" ist, wird sie auf "VA" umgesetzt:

```
unitY = Unit?(data, 1)
cmp = TComp(unitY, "W")
IF cmp = 0
   SetUnit(data, "VA", 1)
END
```

Siehe auch:

Unit?, ConvertUnit, XUNIT, YUNIT

SetZDel

Das Inkrement in z-Richtung (Delta-Z) wird gesetzt.

Deklaration:

```
SetZDel ( Daten, EwZDelta )
```

Parameter:

Daten	Datensatz, dessen z-Inkrement gesetzt werden soll.
EwZDelta	Neues Delta-Z (>0)

Beschreibung:

Das Inkrement in z-Richtung wird gesetzt. Dieser Wert wird u.a. für die Skalierung der z-Achse bei 3D-Darstellungen von segmentierten Daten verwendet.

Siehe auch:

ZDel?, ZOff?, SetZOff

SetZOff

Der Offset in z-Richtung wird gesetzt.

Deklaration:

```
SetZOff ( Daten, EwZOffset )
```

Parameter:

Daten	Datensatz, dessen z-Offset gesetzt werden soll.
EwZOffset	Neuer z-Offset

Beschreibung:

Der Startwert in z-Richtung wird gesetzt. Dieser Wert wird u.a. für die Skalierung der z-Achse bei 3D-Darstellungen von segmentierten Daten verwendet.

Siehe auch:

ZDel?, ZOff?, SetZDel

Sharpness

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Die Schärfe [acum] wird aus der spezifischen Lautheit [sone/Bark] ermittelt.

Deklaration:

```
Sharpness ( Specific_Loudness, Verfahren ) -> Ergebnis
```

Parameter:

Specific_Loudness	Die spezifische Lautheit über einer Bark-Skala, das Lautheits-Tonheits-Muster, das z.B. mit LoudnessSpectrum() ermittelt worden ist. Seine y-Einheit ist sone/Bark, seine x-Einheit ist Bark.
Verfahren	Verfahren zur Berechnung der Schärfe
	0 : DIN 45692:2009-08
	1 : Aures
	2 : von Bismarck
Ergebnis	Das Ergebnis erhält als Einheit "acum".

Beschreibung:

Die Berechnung erfolgt aus dem Lautheits-Tonheits-Muster, das die Darstellung der spezifischen Lautheit N' über einer Bark-Skala ist.

Ist das Lautheits-Tonheits-Muster ein normaler Datensatz, ist die berechnete Schärfe ein Einzelwert.

Ist das Lautheits-Tonheits-Muster ein segmentierter Datensatz, ist das Ergebnis ein normaler Datensatz, der pro Segment der Eingangsdaten einen Schärfewert enthält.

Die x-Achse des Lautheits-Tonheits-Musters liegt typisch Bereich 0 bis 24 Bark vor. Was darüber hinausgeht, wird nicht berücksichtigt.

Die Berechnung der Schärfe basiert auf dem Lautheits-Tonheits-Muster, das wiederum auf dem Terzspektrum basiert. Für die Gesamtgenauigkeit ist die Genauigkeit der Terzfilter eine bedeutende Einflussgröße. Trotz Klasse 1 und hoher Güte können kleine Abweichungen bereits spürbare Abweichungen der Schärfe bedeuten: So bedeuten z.B. 0.4dB bereits 5%, wobei Terzabweichungen sich nicht linear in Abweichungen der Schärfe ausdrücken.

Beispiele:

Schärfe aus einem Mikrofonsignal mic, gemessen in Pa über der Zeit. Aufnahme im Freien

```
Thd = SpecThirds_1( mic, 25, 12500, 0)
Thd_dB = dB ( Thd / 2e-5)
N = LoudnessSpectrum( Thd_dB, 2, 0 )
S = Sharpness( N, 0 )
```

Zeitverlauf der Schärfe aus einem Mikrofonsignal mic, gemessen in Pa über der Zeit. Das Geräusch ist quasi statisch, ändert sich also nur langsam.

```
Thd = SpecThirds( mic, 25, 12500, 0, -2, 1)
Thd_dB = dB ( Thd / 2e-5)
N = LoudnessSpectrum( Thd_dB, 2, 0 )
S = Sharpness( N, 0 )
```

Schärfe aus einem Mikrofonsignal mic, gemessen in Pa über der Zeit. Aufnahme im Fahrzeuginnern (Kabine). Berechnung nach Aures.

```
Thd = SpecThirds_1( mic, 25, 12500, 0)
Thd_dB = dB ( Thd / 2e-5)
N = LoudnessSpectrum( Thd_dB,3, 0 )
S = Sharpness( N, 1 )
```

Terzspektrum mit Schmalbandrauschen bei 1kHz und Schärfe von ca. 1.0 acum

```
Thd_dB = [-60,-60,-60,-60,-60,-60,-60,-60,-60,-60,-60,-60,0,20,40,60,40,20,0,-60,-60,-60,-60,-60,-60,-60,-60]
xoffset Thd_dB 14
N = LoudnessSpectrum( Thd_dB,2, 0 )
S = Sharpness( N, 0 )
```

Siehe auch:

LoudnessSpectrum, SpecThirds_1

ShockResponseSpectrum

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Das Schockantwortspektrum (Shock Response Spectrum, SRS) wird bestimmt.

Deklaration:

```
ShockResponseSpectrum ( Beschleunigung, f_Start, f_Stop, Frequenzlinien, Dämpfung, Modell, MiniMaxi ) -> Ergebnis
```

Parameter:

Beschleunigung	Der Beschleunigungs-Zeit-Verlauf, die Zeit in Sekunden skaliert.
f_Start	Die Startfrequenz für das SRS, in Hz. Ab dieser Frequenz beginnt die Auswertung
f_Stop	Die Stoppfrequenz für das SRS, in Hz. Bei dieser Frequenz endet die Auswertung. f_Start < f_Stop. Die Stoppfrequenz sollte weit unterhalb der halben Abtastfrequenz liegen, am besten einen Faktor 5 weit weg. Bei sehr hohen Stoppfrequenzen treten verfahrensbedingt größere Ungenauigkeiten auf.
Frequenzlinien	So viele Frequenzlinien werden berechnet. >= 1. Die Frequenzlinien werden logarithmisch gleichmäßig über den Frequenzbereich von der Start- bis zur Stoppfrequenz verteilt. Die Anzahl sollte so groß gewählt werden, dass mindestens 6 Linien pro Oktave bestimmt werden.
Dämpfung	0 <= Dämpfung < 0.9. (Damping ratio) Die relative Dämpfung des Systems. Typisch 0.05 oder 0.01. 0.0 ist ein ungedämpftes System.
Modell	Nach welchem Modell erfolgt die Berechnung?
	14 : absolute acceleration model. Berechnung über alle Werte, primary und secondary
	15 : PVSS: pseudo velocity. Berechnung über alle Werte, primary und secondary
	16 : relative displacement model. Berechnung über alle Werte, primary und secondary
	17 : absolute acceleration model. Nur primary Berechnung, also während des Schocks
	18 : PVSS: pseudo velocity. Nur primary Berechnung, also während des Schocks
	19 : relative displacement model. Nur primary Berechnung, also während des Schocks
	20 : absolute acceleration model. Nur secondary Berechnung, also nur nach dem Schock
	21 : PVSS: pseudo velocity. Nur secondary Berechnung, also nur nach dem Schock
	22 : relative displacement model. Nur secondary Berechnung, also nur nach dem Schock
MiniMaxi	Werden Minimal- oder Maximalwerte berechnet?
	0 : MaxiMax. Das Maximum aus den Absolutbeträgen von Minimum und Maximum der Systemantwort wird bestimmt. Das ist die übliche Wahl.
	4 : positive SRS, auch maximum positive SRS: Der größte positive Wert wird ermittelt. Null, falls keine positiven Werte vorhanden sind.
	5 : negative SRS, auch maximum negative SRS: Der betragsmäßig größte negative Wert wird ermittelt. Sein Vorzeichen wird weggelassen. Null, falls nur positive Werte vorhanden sind.
Ergebnis	Das SRS Spektrum

Beschreibung:

absolute acceleration model. Das Ergebnis ist ein Spektrum mit Beschleunigungswerten, wobei die Beschleunigung genauso skaliert ist wie beim Beschleunigungs-Zeit-Verlauf, typisch in "g" Erdbeschleunigung) oder "m/s^2" Dieses Modell ist die übliche Wahl.

relative displacement model. Das Ergebnis ist die relative Auslenkung (Weg) und erhält als Einheit "m" (Meter). Dazu ist es erforderlich, dass der Beschleunigungs-Zeit-Verlauf in "m/s^2" skaliert ist, nicht in "g".

primary SRS, auch initial SRS. Analyse nur während des Schocks, nicht danach. Also Suche nach Minimum und Maximum nur während der Dauer des übergebenen Beschleunigungs-Zeit-Verlaufs.

secondary SRS, auch residual SRS: Analyse nur nach dem Schock, nicht während des Schocks. Also Suche nach Minimum und Maximum nur hinter dem Ende des übergebenen Beschleunigungs-Zeit-Verlaufs.

I. Allg. wird eine MaxiMax Berechnung durchgeführt. Positive und negative SRS werden nur benötigt, wenn die Symmetrie untersucht werden soll.

PVSS: pseudo velocity. Wird ermittelt aus dem relative displacement shock spectrum, das mit 2*PI*f multipliziert wird. Dazu ist es erforderlich, dass der Beschleunigungs-Zeit-Verlauf in "m/s^2" skaliert ist, nicht in "g".

Die Funktion nimmt an, dass zu Beginn das System in Ruhe bei 0 ist, also Weg=0 und Geschwindigkeit=0.

Die Funktion ermittelt die maximale Auslenkung eines Systems von Feder, Masse und Dämpfer mit gegebener Eigenfrequenz und Dämpfung.

Das Ergebnis ist über der Eigenfrequenz aufgetragen.

Vor allem bei niedrigen Startfrequenzen kommt es vor, dass innerhalb der (kurzen) Dauer des Beschleunigungs-Zeit-Verlaufs die Systemantwort zwar mit einer Schwingung begonnen hat, aber das Minimum oder Maximum noch gar nicht erreicht hat.

Dann sollte eine Berechnung über alle Werte, primary und secondary, gewählt werden.

Die Funktion arbeitet bei der Bestimmung der Extremwerte mit der vorliegenden Abtastfrequenz.

Ist die Eigenfrequenz nicht klein gegenüber der Abtastfrequenz, ergeben sich Fehler, weil nicht genau beim Extremwert abgetastet wird. Z.B. ergibt sich ein Fehler von bis zu 1%, wenn das Verhältnis der Frequenzen 23 beträgt. Ggf. kann vorher oder auch nachher interpoliert werden.

Nicht mehr unterstützte Parameter:

Modell=0: absolute acceleration model

Modell=1: relative displacement model

Modell=2: absolute acceleration model ohne Verlängerung

Modell=3: relative displacement model ohne Verlängerung

Modell=4: absolute acceleration model, andere Berechnung

Modell=5: PVSS: pseudo velocity

Modell=6: relative displacement model, andere Berechnung

Modell=7: absolute acceleration model, ohne Verlängerung, andere Berechnung

Modell=8: PVSS: pseudo velocity, ohne Verlängerung

Modell=9: relative displacement model, ohne Verlängerung, andere Berechnung

Modell=10: absolute acceleration model. Nur secondary

Modell=11: PVSS: pseudo velocity. Nur secondary

Modell=12: relative displacement model. Nur secondary

MiniMaxi=1: Der Absolutbetrag des Maximums der Systemantwort wird bestimmt.

MiniMaxi=2: Das Minimum aus den Absolutbeträgen von Minimum und Maximum der Systemantwort wird bestimmt.

MiniMaxi=3: Der Absolutbetrag des Minimums der Systemantwort wird bestimmt.

Beispiele:

Das Schockantwortspektrum wird bestimmt. Der Beschleunigungs-Zeit-Verlauf hat eine Abtastzeit von 0.1ms und enthält 5000 Messwerte. Seine y-Einheit ist "m/s^2".Zwischen 5.0Hz und 500Hz wird das SRS mit 100 Stützstellen bestimmt. Die Dämpfung beträgt 0.05. Das Maximax-Spektrum wird als absolute Beschleunigung ermittelt. Das Ergebnis wird später im Kurvenfenster i. Allg. doppelt logarithmisch dargestellt.

```
SRS = ShockResponseSpectrum ( Acceleration, 5.0, 500.0, 100, 0.05, 14, 0 )
```

Hier wird die Frequenz so umskaliert, dass sie der Konvention für die Terzskalierung entspricht, also (1Hz: 0, 10Hz: 10, 100Hz: 20, ...). Bei der graphischen Darstellung muss dann am Kurvenfenster die x-Achse mit der Terz/ Oktavbeschriftung eingestellt werden.

```
SRS = ShockResponseSpectrum ( Acceleration, 5.0, 500.0, 100, 0.05, 14, 0 )
SRS_Terz = XYof ( lg ( SRS.x ) * 10, SRS.y )
```

Zahlenbeispiel, Half sine pulse, 1ms, 1m/s^2. Das ist ein theoretisches Signal. Nach dem Puls liegt eine konstante Geschwindigkeit vor. Dieses Signal kann nicht auf einem Shaker erzeugt werden.

```
Acceleration = sin ( ramp( 0, 1e-5, 101 ) * PI2 * 500 )
yunit Acceleration m/s^2
xunit Acceleration s
SRS = ShockResponseSpectrum ( Acceleration, 0.1, 10000.0, 1000, 0.0, 14, 0 )
```

Zahlenbeispiel, Half sine pulse, 1ms, 1m/s^2. Aber beginnend mit einer Phase konstanter Geschwindigkeit. Am Ende dieses Verlaufs ist die Geschwindigkeit wieder 0. Das pseudo velocity Shock Sepctrum wird ermittelt.

```
in1 = sin ( ramp ( 0, 1e-5, 101 ) * PI2 * 500 )
insum = sum( in1 )
in2 = xdel ( leng ( 0, 50000 ), xdel?(in1))
Acceleration = join ( in2-insum/leng?(in2), in1)
yunit Acceleration m/s^2
xunit Acceleration s
PVSS = ShockResponseSpectrum ( Acceleration, 0.1, 20000.0, 1000, 0.0, 15, 0 )
```

Siehe auch:

SDOF_Response

SHOW

Eine Variable wird in einem freischwebenden Fenster angezeigt.
Alternativer Name: **ZEIGEN**

Deklaration:

```
SHOW Variablename
```

Parameter:

Variablename	Anzuzeigende Variable

Beschreibung:

Die als Parameter angegebene Variable wird angezeigt. Ist die Variable ein Datensatz oder eine Datengruppe, so wird ein freischwebendes Kurvenfenster geöffnet.

Ist die Variable vom Typ Text, so wird sie in einem speziellen Textfenster angezeigt.

Existierte für diese Variable bereits ein Fenster, kommt es in den Vordergrund.

Sie können im Variablennamen Jokerzeichen (Wildcards) angeben, um eine Reihe von Variablen anzuzeigen. Das Jokerzeichen '?' steht dabei für genau ein beliebiges Zeichen, das Jokerzeichen '*' für eine unbestimmte Anzahl von beliebigen Zeichen.

```
SHOW *
```

Alle Variablen werden angezeigt.

```
SHOW ??
```

Alle Variablen, deren Name genau 2 Zeichen lang ist, werden angezeigt.

```
SHOW *1a*
```

Alle Variablen, in denen die Zeichenfolge '1a' (auch zu Beginn oder am Ende) vorkommt, werden angezeigt.

```
SHOW a*:kanal?
```

Aus allen Datengruppen, deren Name mit einem 'a' beginnt, werden all jene Kanäle angezeigt, die den Namen 'Kanal', gefolgt von einem beliebigen Zeichen, besitzen.

Beispiele:

Ein Datensatz wird durch Spline-Interpolation vergrößert und angezeigt. Das Maximum dieses Datensatzes wird berechnet und ebenfalls angezeigt:

```
Daten = IPol(Daten, 3)
SHOW Daten
x = Max(Daten)
SHOW x
```

Siehe auch:

CwLoadCCV, CwNewWindow

signum

Verfügbar ab: Professional Edition

Signumfunktion bzw. Vorzeichenfunktion

Deklaration:

```
signum ( x ) -> Ergebnis
```

Parameter:

x	x
Ergebnis	Ergebnis

Beschreibung:

Die Funktion liefert 1, falls x > 0; liefert -1, falls x < 0; liefert 0, falls x = 0.

Die Funktion vergleicht nur mit exakt 0.0. Ist ungefähr null gemeint, kann das Signal vorher bearbeitet werden, z.B. mit round().

Die Eingangsdaten können Events und Segmente haben. Äquidistante und XY-Daten werden unterstützt, bei XY-Daten wird die Y-Komponente verrechnet.

Die Funktion ist nur für reelle Argumente (nicht komplexe) implementiert.

Beispiele:

```
sign = signum ( x )
```

sin

Sinus, trigometrische Funktion

Deklaration:

```
sin ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Eingangsdaten (Winkel). Erlaubte Typen: [ND],[XY].
Ergebnis	Sinus des Parameters

Beschreibung:

Es wird die trigonometrische Funktion sin berechnet, wobei im Bogen- oder Gradmaß entsprechend der Einheit des übergebenen Parameters gearbeitet wird.

Die x-Koordinate(n) von Ergebnis und Parameter sind gleich.

Anmerkungen

- Der Parameter der Funktion sollte keine Einheit oder die Einheiten 'Rad', 'Grad', '° ' haben. Bei anderen Einheiten wird eine Warnung erzeugt und die Einheit ins Ergebnis übernommen.
- Der Parameter darf strukturiert sein (Events/ Segmente).
- Die zugehörige Umkehrfunktion heißt asin().

Beispiele:

Bei entsprechender Einheit wird der Parameter im Gradmaß verstanden:

```
one = sin(90 '°')
```

Zwei Möglichkeiten zur Erzeugung eines Datensatzes, der genau die erste Halbwelle der Sinusfunktion zeigt:

```
NDhalfWave = sin(Ramp(0, PI/100, 100))
NDhalbWave = sin(Ramp(0, 1, 180) * 1 'Grad')
```

Siehe auch:

cos, tan, asin

SlClip

Steilheitsbegrenzung eines Datensatzes
Alternativer Name: **StGren**

Deklaration:

```
SlClip ( Daten, EwMaxSteigung ) -> Filtrat
```

Parameter:

Daten	Zu filternder Datensatz [ND].
EwMaxSteigung	Maximal erlaubte Steigung dy/dx
Filtrat	Gefilteter Datensatz.

Beschreibung:

Steilheits-Begrenzer, der maximale Anstieg dy/dx zwischen zwei benachbarten Werten eines Datensatzes wird begrenzt. Die Funktion kann also benutzt werden, um Messwerte in einem Datensatz herauszufiltern, die aufgrund der physikalisch begrenzten Änderungsgeschwindigkeit des Signals nicht gültig sein können.

Die Steigung in einem Punkt ist dabei definiert als Steigung zum nächsten Messwert, wird also berechnet nach:

```
Steigung[i] = (y[i+1]-y[i]) / dx
```

Wenn die derart berechnete Steigung in einem Punkt größer als der als Parameter übergebene maximale Anstieg ist, wird der nächste Datenpunkt mit diesem maximalen Anstieg berechnet und der Algorithmus mit diesem neuem Wert fortgesetzt.

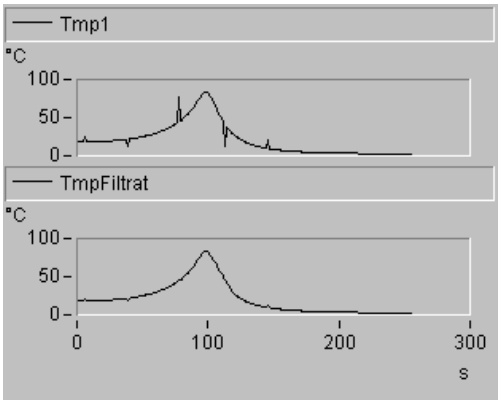
```
y[i+1] = y[i] + EwMaxSteigung * dx
```

Der erste Wert des Datensatzes bleibt unverändert, da er als Startwert für den Algorithmus benutzt wird.

Beispiele:

Die Messwerte eines Temperatursensors sind auf Grund von Störungen bei der Datenübertragung verfälscht worden. Gelegentlich treten zufällig belegte Werte auf, die herausgefiltert werden sollen. Temperaturänderungen von mehr als 4° Celsius pro Sekunde sind an der Messstelle aus physikalischer Sicht nicht möglich.

```
TempFilt = SlClip(Temp, 4)
```



Siehe auch:

Smo, Hyst, SearchLevel

Sleep

Pause, Sequenzausführung wird für eine vorgebbare Zeit unterbrochen.

Deklaration:

```
Sleep ( EwWartezeit )
```

Parameter:

EwWartezeit	Wartezeit (in Sekunden)

Beschreibung:

Die Sequenzausführung wird für die angegebene Zeit unterbrochen, die Funktion kehrt erst zurück, wenn die eingestellte Zeit abgelaufen ist.

Während die Funktion wartet, ist FAMOS nicht bedienbar (Ausnahme: Kurvenfenster). Allerdings kann eine laufende Sequenz auf die übliche Weise ("Sequenz"-Symbol mit Kontextmenü im Infobereich der Windows-Taskleiste oder [Strg+Break]) abgebrochen werden.

Während der Wartezeit ist FAMOS per DDE teilweise erreichbar: Daten können empfangen, aber keine Anweisungen ausgeführt werden.

Beispiele:

In einer Schleife werden alle Kanäle aus einer Meßwertdatei geladen. Jeder Kanal wird für 5s angezeigt, dann wieder entfernt und der nächste Kanal geladen.

```
fh = FileOpenDSF("c:\tmp\test.dat", 0)
IF fh > 0
   count = FileObjNum?(fh)
   n = 1
   WHILE n <= count
      data = FileObjRead(fh, n)
      SHOW data
      Sleep(5)
      n = n+1
   END
END
```

Siehe auch:

PAUSE

sMax

Stat()-Funktion: Maximum eines Datensatzes

Beschreibung:

sMax ist eine vordefinierte Variable und wird durch einen vorherigen Aufruf der Statistikfunktion Stat() belegt.

Alternativ kann die Funktion Max() verwendet werden.

Beispiele:

Spitze-Spitze-Wert einer sinusförmigen Spannung:

```
Stat(U)
Uss =  sMax - sMin
```

Dies ist äquivalent zu:

```
Uss =  Max(U) - Min(U)
```

Siehe auch:

Stat, Max, sMin

sMaxPos

Stat()-Funktion: Position (x-Wert) des Maximums eines Datensatzes

Beschreibung:

sMaxPos ist eine vordefinierte Variable und wird durch einen vorherigen Aufruf der Statistikfunktion Stat() belegt.

Beispiele:

Bestimmung des Maximums in einem Spannungsverlauf:

```
Stat(U)
u_max = sMax
t_max = sMaxPos
```

Dies ist äquivalent zu:

```
u_max = Max(U))
t_max = Pos(U, u_max)
```

Siehe auch:

Stat, sMax, Pos, PosiEx2

sMean

Stat()-Funktion: Arithmetischer Mittelwert eines Datensatzes
Alternativer Name: **sMittel**

Beschreibung:

sMean ist eine vordefinierte Variable und wird durch einen vorherigen Aufruf der Statistikfunktion Stat() belegt.

Alternativ kann die Funktion Mean() verwendet werden.

Beispiele:

Einen Datensatz mittelwertfrei machen:

```
Stat(U)
U =  U - sMean
```

Dies ist äquivalent zu:

```
U =  U - Mean(U)
```

Siehe auch:

Stat, Mean, sRMS

sMin

Stat()-Funktion: Minimum eines Datensatzes

Beschreibung:

sMin ist eine vordefinierte Variable und wird durch einen vorherigen Aufruf der Statistikfunktion Stat() belegt.

Alternativ kann die Funktion Min() verwendet werden.

Beispiele:

Spitze-Spitze-Wert einer sinusförmigen Spannung:

```
Stat(U)
Uss =  sMax - sMin
```

Dies ist äquivalent zu:

```
Uss =  Max(U) - Min(U)
```

Siehe auch:

Stat, Min, sMax

sMinPos

Stat()-Funktion: Position (x-Wert) des Minimums eines Datensatzes

Beschreibung:

sMinPos ist eine vordefinierte Variable und wird durch einen vorherigen Aufruf der Statistikfunktion Stat() belegt.

Beispiele:

Bestimmung des Minimums in einem Spannungsverlauf:

```
Stat(U)
u_min = sMin
t_min = sMinPos
```

Dies ist äquivalent zu:

```
u_min = Min(U))
t_min = Pos(U, u_min)
```

Siehe auch:

Stat, sMin, Pos, PosiEx2

Smo

Glättung mit vorgebbarer Mittelungszeit
Alternativer Name: **Glatt**

Deklaration:

```
Smo ( Daten, EwBreite ) -> Geglättet
```

Parameter:

Daten	Datensatz, der geglättet werden soll [ND].
EwBreite	Breite des Glättungsintervalls
Geglättet	Geglätteter Datensatz

Beschreibung:

Der übergebene Datensatz wird geglättet, indem über ein gewisses Intervall gewichtet gemittelt wird. Die Breite dieses Intervalls ist durch den zweiten Parameter bestimmt. Je größer die Intervallbreite gewählt wird, desto deutlicher wird der Glättungseffekt.

Wenn auf dem Datensatz periodische Störungen sind, können diese durch Glätten (sogar perfekt) unterdrückt werden, wenn die Intervallbreite gleich der Periodendauer der Störung oder einem ganzzahligen Vielfachen davon gewählt wird.

Die Funktion Smo() ist ein digitales Filter, dessen Zeitkonstante in der Größenordnung der Intervallbreite liegt.

Die Gewichtsfunktion ist dreieckig, wenn sie mehr als 5 Punkte des Datensatzes lang ist. Die Gewichtsfunktion kann nur eine ungerade Anzahl von Punkten breit sein, d. h. die geforderte Intervallbreite wird i. a. nicht exakt eingehalten, sondern der nächstmögliche Wert benutzt.

Die Koeffizienten des unterlegten digitalen Filters (mittelwerttreues Dreieckfilter, Summe der Koeffizienten ist 1) als Parameter der Glättungsbreite in Punkten kann man sich in FAMOS am einfachsten aus der Impulsantwort des Filters berechnen, siehe Beispiel.

- Wird die Intervallbreite so klein gewählt, dass über nur noch je 5 Punkte gemittelt wird, wirkt die Funktion Smo() wie die Funktion Smo5(), bei noch kleinerem Intervall wie die Funktion Smo3().
- Das benutzte Filter ist nicht kausal. Es hat die Phase Null. Damit wird die Lage von Flanken durch ein Glätten nicht verfälscht. Die Länge des Datensatzes wird nicht verändert.
- Wegen der Benutzung eines nichtkausalen Filters gibt es an den Rändern des Datensatzes natürliche Einschwingvorgänge.
- Für eine Filterung in Randnähe wird angenommen, dass die Datensätze mit ihren Randwerten konstant fortgesetzt sind.
- Die Intervallbreite darf nicht größer sein als der übergebene Datensatz lang ist.
- Um allein höherfrequente Störungen zu unterdrücken, ist es besser, die Funktion Smo() mehrere Male mit kurzer Intervallbreite zu benutzen als einmal mit langer.

Für eine Filterbreite von n Punkten lässt sich das digitale Filter durch die folgende Differenzengleichung beschreiben:

$$y_{[k]} = \sum_{i=-\frac{n-1}{2}}^{\frac{n-1}{2}} \frac{2n - 4\,|\,i\,|}{n^2 + 1} * u_{[k+i]}$$

Beispiele:

```
NDsmooth = Smo(NDdata, 1E-3 's')
```

Ein Datensatz mit der Abtastzeit 10-4s wird geglättet mit einem Filter der Breite 10-3s. Damit werden viele höherfrequente Störungen reduziert.

Bestimmung der Filterkoeffizienten in Abhängigkeit der Glättungsbreite (in Punkten) aus der Impulsantwort

```
imp = Ramp(0,1,100)*0
imp[50] = 1
antw = Smo(imp, GLÄTTUNGSBREITE_IN_PUNKTEN)
```

Beispiel: Datensatz mit Abtastzeit 0.15s, Glättungsbreite 1s => GLÄTTUNGSBREITE_IN_PUNKTEN = 7

[antw] enthält die von 0 verschiedenen Werte (0.04, 0.12, 0.20, 0.28, 0.20, 0.12, 0.04)

Also:

```
y[k] = 0.04*u[k-3] +0.12*u[k-2] +0.2*u[k-1] +0.28*u[k] +0.20*u[k+1] +0.12*u[k+2] +0.04*u[k+3]
```

Siehe auch:

Smo3, Smo5, DFilt, Median, FiltLP, FiltLpZ, SavitzkyGolay

Smo3

Glättung über 3 Punkte
Alternativer Name: **Gla3**

Deklaration:

```
Smo3 ( Daten ) -> Geglättet
```

Parameter:

Daten	Datensatz, der geglättet werden soll. [ND]
Geglättet	Geglätteter Datensatz

Beschreibung:

Der Datensatz wird geglättet, indem über je 3 benachbarte Punkte gewichtet gemittelt wird. Das hinterlegte digitale Filter gehorcht folgender Gleichung:

```
y[k] = 0.25 * u[k-1] + 0.5 * u[k] + 0.25 * u[k+1]
```

Dabei ist k ein laufender Index, u ein Wert des übergebenen Datensatzes, y ein Wert des erzeugten Datensatzes.

Dies ist ein nichtkausales Filter, um beim Filtern die Lage von Flanken nicht zu verändern: Das Filter hat nämlich die Phase 0.

Die Länge des Datensatzes wird durch das Glätten nicht verändert.

Für die Filterung in der Nähe der Ränder des Datensatzes wird angenommen, dass der Datensatz mit seinen Randwerten konstant fortgesetzt ist.

Beispiele:

Zweifache Anwendung der Funktion zur gezielten Unterdrückung von hochfrequenten Störungen:

```
NDsmooth = Smo3(Smo3(NDdata))
```

Siehe auch:

Smo5, Smo, DFilt, Median, FiltLP, FiltLpZ, SavitzkyGolay

Smo5

Glättung über 5 Punkte
Alternativer Name: **Gla5**

Deklaration:

```
Smo5 ( Daten ) -> Geglättet
```

Parameter:

Daten	Datensatz, der geglättet werden soll. [ND]
Geglättet	Geglätteter Datensatz

Beschreibung:

Der Datensatz wird geglättet, indem über je 5 benachbarte Punkte gewichtet gemittelt wird. Das hinterlegte digitale Filter gehorcht folgender Gleichung:

```
y[k]=(1/9)*u[k-2]+(2/9)*u[k-1]+(3/9)*u[k]+(2/9)*u[k+1]+(1/9)*u[k+2]
```

Dabei ist k ein laufender Index, u ein Wert des übergebenen Datensatzes, y ein Wert des erzeugten Datensatzes.

Dies ist ein nichtkausales Filter, um beim Filtern die Lage von Flanken nicht zu verändern: Das Filter hat nämlich die Phase 0.

Die Länge des Datensatzes wird durch das Glätten nicht verändert.

Für die Filterung in der Nähe der Ränder des Datensatzes wird angenommen, dass der Datensatz mit seinen Randwerten konstant fortgesetzt ist.

Der Glättungseindruck ist deutlicher als bei der Funktion Smo3().

Beispiele:

Zweifache Anwendung der Funktion zur gezielten Unterdrückung von hochfrequenten Störungen:

```
NDsmooth = Smo5(Smo5(NDdata))
```

Siehe auch:

Smo3, Smo, DFilt, Median, FiltLP, FiltLpZ, SavitzkyGolay

SolveLinEq

Verfügbar ab: Professional Edition

Lösung eines linearen Gleichungssystems

Deklaration:

```
SolveLinEq ( Matrix A, Spaltenvektor b [, Fehlerbehandlung] [, MehrfachLoesung] ) -> Ergebnis
```

Parameter:

Matrix A	Matrix A mit Koeffizienten der linken Seite
Spaltenvektor b	Spaltenvektor b mit Koeffizienten der rechten Seite
Fehlerbehandlung	Wie soll im Fehlerfall reagiert werden? (optional , Standardwert: 0)
	0 : Abbruch mit Fehlermeldung
	1 : Leeren Datensatz zurückgeben
MehrfachLoesung	Verhalten bei unendlich vielen Lösungen (optional , Standardwert: 0)
	0 : Nur eindeutige Lösungen zurückgeben. Unendlich viele Lösungen sind ein Fehler.
	1 : Eindeutige und mehrfache Lösungen zurückgeben. Im Fall von unendlich vielen Lösungen wird dann eine ausgewählt.
Ergebnis	Lösungsvektor

Beschreibung:

Eine Matrix ist ein segmentierter Datensatz. Die Segmente sind die Spalten.

Als Vektor wird in diesem Zusammenhang eine einreihige Matrix verstanden, also eine Matrix mit nur einer Zeile oder mit nur einer Spalte.

Ein Spaltenvektor ist eine Matrix mit nur einer Spalte. Er ist ein Datensatz ohne Segmente.

Ein Spaltenvektor kann auch durch einen segmentierten Datensatz mit genau 1 Segment dargestellt werden.

A * x = b wird gelöst, also x ermittelt. A ist eine quadratische Matrix. x und b sind Spaltenvektoren.

Das Ergebnis der Funktion ist ein Spaltenvektor, also ein Datensatz ohne Segmente.

Bei Aufruf mit falschen Parametern oder nicht ausreichend Speicher wird stets mit der üblichen Fehlermeldung abgebrochen.

Einheiten und Abtastzeiten werden nicht beachtet.

Homogene lineare Gleichungssysteme

Der Parameter MehrfachLoesung steuert das Verhalten.

Falls nur eindeutige Lösungen gewünscht sind, gilt: Die triviale Lösung (alles Nullen) zurückgegeben, falls nur sie existiert und damit eindeutig ist. Unendlich viele Lösungen (nicht-trivial) sind ein Fehler.

Falls auch mehrfache Lösungen gewünscht sind, gilt: Falls vorhanden, wird die nicht-triviale Lösung zurückgegeben. Eine von den unendlich vielen Lösungen wird dabei ausgewählt. Das Ergebnis kann z.B. mit einem beliebigen Faktor multipliziert werden und ist dann immer noch Lösung. Falls die nicht-triviale Lösung nicht vorhanden ist, wird die triviale Lösung (alles Nullen) zurückgegeben.

Ist die Matrix A invertierbar, gibt es nur die triviale Lösung, sonst auch die nicht-triviale.

Beispiele:

Lösung eines linearen Gleichungssystems A * x = b

```
A = leng(0,9) ; test data
setsegLen( A,3)
A[3,1] = 4 ; row 1, column 3
A[1,2] = 1
A[2,3] = 2
; A:
; 0 0 4
; 1 0 0
; 0 2 0
b = [4, 2, -2]
xi = SolveLinEq ( A, b )
; result: xi = [2, -1, 1]
```

Siehe auch:

MatrixInverse

Sort

Sortierung der y-Werte eines Datensatzes

Deklaration:

```
Sort ( Daten, EwOption ) -> Sortiert
```

Parameter:

Daten	Zu sortierender Datensatz. Erlaubte Typen: [ND],[XY].
EwOption	Definiert die Art der Sortierung
	1 : [nur für ND] Ausgabe der aufsteigend sortierten y-Werte
	2 : [ND] Ausgabe der absteigend sortierten y-Werte
	3 : [ND] Aufsteigend sortierte y-Werte; Ausgabe der zugehörigen x-Stellen
	4 : [ND] Absteigend sortierte y-Werte; Ausgabe der zugehörigen x-Stellen
	5 : [nur für XY] Aufsteigend sortiert bezüglich Y
	6 : [XY] Absteigend sortiert bezüglich Y
	7 : [XY] Aufsteigend sortiert bezüglich X
	8 : [XY] Absteigend sortiert bezüglich X
Sortiert	Sortierter Datensatz

Beschreibung:

Die Werte eines Datensatzes werden mit dem Quicksort-Algorithmus sortiert. Wahlweise können die Werte auf- oder absteigend sortiert werden.

Der Datentyp des Resultats ist gleich dem Datentyp des zu sortierenden Datensatzes, das Datenformat wird ebenfalls beibehalten.

Die Optionswerte 1 bis 4 sind nur für äquidistant abgetastete Datensätze (ND) erlaubt:

Bei Eingabe von 1 für den Parameter [EwOption] werden die y-Werte des Datensatzes aufsteigend sortiert und ausgegeben.

Bei Eingabe von 2 für den Parameter [EwOption] werden die y-Werte des Datensatzes absteigend sortiert und ausgegeben.

Bei Eingabe von 3 für den Parameter [EwOption] werden die y-Werte des Datensatzes aufsteigend sortiert. Ausgegeben werden die den y-Werten des (nicht sortierten) Datensatzes zugehörigen x-Stellen. Die x-Stelle des kleinsten y-Werts wird zuerst ausgegeben, die x-Stelle des zweitkleinsten y-Werts wird als zweites ausgegeben usw. Die x-Stelle des größten y-Werts wird zuletzt ausgegeben.

Bei Eingabe von 4 für den Parameter [EwOption] werden die y-Werte des Datensatzes absteigend sortiert. Ausgegeben werden die den y-Werten des (nicht sortierten) Datensatzes zugehörigen x-Stellen. Die x-Stelle des größten y-Werts wird zuerst ausgegeben, die x-Stelle des zweitgrößten y-Werts wird als zweites ausgegeben usw. Die x-Stelle des kleinsten y-Werts wird zuletzt ausgegeben

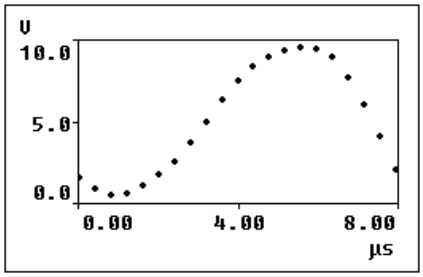
Die Optionswerte 5 bis 8 sind nur für XY-Daten erlaubt:

Bei der Eingabe von 5 oder 6 wird der XY-Datensatz so sortiert, dass die Y-Werte im Ergebnis auf- bzw. absteigend angeordnet sind.

Bei der Eingabe von 7 oder 8 wird der XY-Datensatz so sortiert, dass die X-Werte im Ergebnis auf- bzw. absteigend angeordnet sind. Die Option 7 ist insbesondere nützlich, um einen XY-Datensatz mit nicht-monotoner X-Spur in einen Datensatz mit monotoner X-Spur zu wandeln.

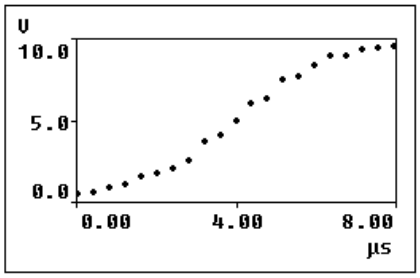
Beispiele:

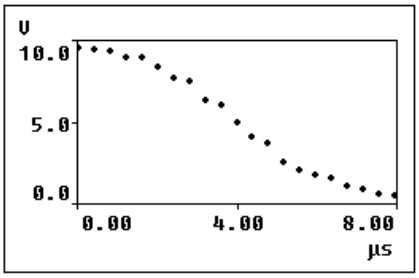
Der zur folgenden Abbildung gehörige Datensatz soll sortiert werden. Der Datensatz besteht aus 21 Werten:




```
NDauf = Sort(NDdaten, 1)
NDab = Sort(NDdaten, 2)
```

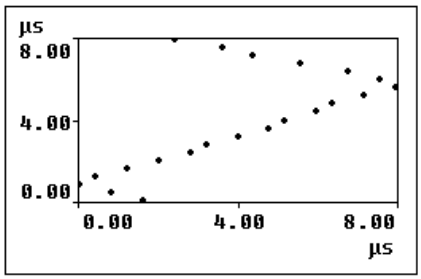
Als Ergebnis der Sortierung wird in imc FAMOS bei Eingabe von 1 für den Parameter [EwOption] (aufsteigende Sortierung der y-Werte) der auf der linken Abbildung dargestellte Datensatz NDauf ausgegeben. Bei Eingabe von 2 für den Parameter [EwOption] (absteigende Sortierung der y-Werte) wird in imc FAMOS als Ergebnis der auf der rechten Abbildung dargestellte Datensatz NDab ausgegeben:

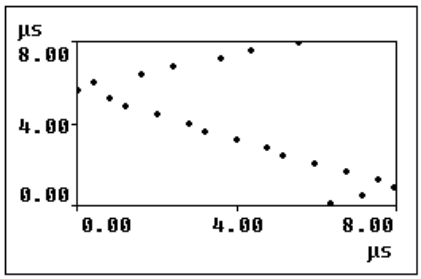





```
NDaufx = Sort(NDdaten, 3)
NDabx = Sort(NDdaten, 4)
```

Als Ergebnis der Sortierung wird in imc FAMOS bei Eingabe von 3 für den Parameter [EwOption] (zugehörige x-Stellen bei aufsteigender Sortierung der y-Werte) der auf der linken Abbildung dargestellte Datensatz NDaufx ausgegeben. Bei Eingabe von 4 für den Parameter [EwOption] (zugehörige x-Stellen bei absteigender Sortierung der y-Werte) wird in imc FAMOS als Ergebnis der auf der rechten Abbildung dargestellte Datensatz NDabx ausgegeben:





Siehe auch:

Mirror, Top

SoundIndex

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Artikulationsindex und andere Kenngrößen eines Schallsignals. Berechnung einer Kenngröße über das gesamte Schallsignal.

Deklaration:

```
SoundIndex ( Schallsignal, Berechnung ) -> Ergebnis
```

Parameter:

Schallsignal	Zeitverlauf des Schallsignals, die Zeit in Sekunden skaliert
Berechnung	Berechnung
	0 : Artikulationsindex (0 .. 100%)
	1 : Offener Artikulationsindex (0 .. 224.78%)
Ergebnis	

Beschreibung:

Um den Artikulationsindex zu berechnen, müssen die Eingangsdaten für die interne dB-Berechnung schon auf den Bezugswert skaliert sein, also z.B. durch 2e-5 Pa geteilt sein.

Der Artikulationsindex (AI) wurde von Leo L. Beranek eingeführt, um die Güte von Telefonanlagen zu beurteilen. Anfang der 70er Jahre änderte B. Braune diese Berechnungsvorschrift für die Fahrzeugakustik so ab, dass diese aus Terzspektren möglich wurde. Das gemessene Spektrum wird mit 2 empirisch ermittelten Referenzkurven verglichen. Für jedes Frequenzband wird ein prozentualer Anteil berechnet.

Der Artikulationsindex ergibt sich aus der Summe der Einzelanteile von 200Hz bis 6300Hz. Die Verständlichkeit ist sehr gut, wenn der AI=100% ist bzw. sehr schlecht, wenn der AI gegen 0% geht.

Die Referenzkurve liegt z.B. auf 44.0 und 74.0 dB(A) bei 1000Hz, auf 23.1 und 53.1 dB(A) bei 200Hz.

Der offene Artikulationsindex unterscheidet sich vom Artikulationsindex nur im unteren Referenzspektrum, das für alle Frequenzen 0 dB ist.

Der Artikulationsindex wird intern über einen Aufruf SpecThirds_1 (..., 1) berechnet.

Beispiele:

```
AI = SoundIndex( vibration / 2e-5'Pa', 0 )
```

Siehe auch:

LoudnessLevel

SoundIntensityThirds

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Schallintensität (pro Terz) abhängig von der Zeit

Deklaration:

```
SoundIntensityThirds ( Schalldruck1, Schalldruck2, f1, f2, Frequenzbewertung, Zeitbewertung, AusgabeIntervall, Spacer, Luftdichte [, Einschwingen] ) -> Ergebnis
```

Parameter:

Schalldruck1	Zeitverlauf des Schalldrucks von Mikrofon 1 in Pa, die Zeit in Sekunden skaliert.
Schalldruck2	Zeitverlauf des Schalldrucks von Mikrofon 2 in Pa, die Zeit in Sekunden skaliert.
f1	Mittenfrequenz der untersten Terz in Hz
f2	Mittenfrequenz der obersten Terz in Hz
Frequenzbewertung	Mit welcher Frequenzbewertung wird das Resultat gewichtet?
	0 : linear
	1 : A-Bewertung
	2 : B-Bewertung
	3 : C-Bewertung
	4 : D-Bewertung
Zeitbewertung	Mittelung, Zeitbewertung der Schallintensität
	-5 : Mittelwert des Intervalls
	-6 : Mittelwert ab Start
AusgabeIntervall	Ausgabe-Intervall der Terz-Spektren; in diesem Zeit-Intervall werden die Terz-Spektren bestimmt. Ganzzahliges Vielfaches der Abtastzeit des Schwingungssignals. Angegeben in Sekunden.
Spacer	Abstand zwischen den Mikrofonen, angegeben in m; > 0; i.a. im Bereich weniger cm.
Luftdichte	Dichte der Luft, angegeben in kg/m^3. Z.B. 1.204 (bei 0°C) oder 1.2038; > 0
Einschwingen	Behandlung des Einschwingens zu Beginn der Berechnung (optional , Standardwert: 0)
	0 : Während des Einschwingens wird das Ergebnis bei jeder noch einschwingenden Terz auf Null gesetzt.
	1 : Das erste gültige Ergebnis jeder Terz wird konstant nach vorn fortgesetzt.
	2 : Der Bereich, in dem noch irgendein Filter einschwingt, wird abgeschnitten. Die Länge verkürzt sich, der Anfangszeitpunkt des Ergebnisses verschiebt sich.
Ergebnis	Schallintensität abhängig von der Zeit

Beschreibung:

Verfahren

Die Schallintensität wird mit der Zweimikrofontechnik bestimmt. Dazu wird eine Intensitätssonde benutzt, bei der zwei Mikrofone in kurzem festen Abstand montiert sind.

Der Verlauf des Schalldrucks für beide Mikrofone liegt als Signal vor.

Die Berechnung erfolgt über eine Produktbildung aus Schalldruck und Schallschnelle.

Als Schalldruck wird der aus beiden Mikrofonen gemittelte Schalldruck benutzt.

Die Schallschnelle wird über eine Integration der Druckdifferenz ermittelt. Die Druckdifferenz ist eine Näherung für die in der Eulergleichung stehende Ableitung des Schalldrucks.

Damit das berechnete Integral bedingt durch Startwerte und (kleine) Offsetfehler nicht wegdriftet und das Ergebnis nicht (stark) verfälscht, wird ein Hochpassfilter eingesetzt.

Zum Zweck der Ausgabe erfolgt eine Mittelwertbildung, für den der Parameter Zeitbewertung den Mittelungsbereich festlegt.

Alle Ergebniswerte sind Intensitäten und in W/m^2 angegeben, wenn die Eingangsdaten in Pa über s skaliert sind.

Vorzeichen

Die errechnete Schallintensität ist ein Mittelwert mit Vorzeichen.

Das Vorzeichen drückt die Richtung aus.

Das Vorzeichen ist positiv, wenn die Energie zuerst auf das erste, danach auf das zweite Mikrofon trifft. Also positiv, wenn Mikrofon 1 zur Quelle

zeigt.

Grenzen des Verfahrens

Der Mikrofonabstand bestimmt den möglichen Frequenzbereich der Auswertung:

Sehr niedrige Frequenzanteile können nicht mehr präzise ausgewertet werden, weil ihr Phasenunterschied kaum oder nicht mehr erkennbar ist.

Sehr hohe Frequenzanteile können nicht mehr präzise ausgewertet werden, weil ihre Wellenlänge in die Größenordnung des Mikrofonabstands kommt.

Die oben genannte Annäherung der Ableitung durch eine Druckdifferenz funktioniert nur gut, wenn die Frequenzanteile deutlich unterhalb 1/4 der Abtastfrequenz liegen.

Terzabhängige Analyse

Die Berechnung erfolgt pro Terz: Zuerst wird die Terzfilterung durchgeführt, danach die Berechnung der Schallintensität.

Die beiden Frequenzgrenzen f1 und f2 sollten als Terzmittenfrequenz angegeben werden, z.B. f1 = 63 Hz und f2 = 5000 Hz. f1 <= f2.

Die oberste Terz muss mit ihrem Frequenzband vollständig innerhalb der halben Abtastfrequenz liegen.

Die oberste Terz sollte deutlich unterhalb 1/4 der Abtastfrequenz liegen.

Das Einschwingen wird bei der 1kHz Terz zu 20ms angenommen.

Diese Dauer ist umgekehrt proportional zur Terzfrequenz.

Bei sehr niedrigen Terzen wird diese Dauer beachtlich.

Eine entsprechend lange dauernde Messung ist dann vorzusehen.

Das Ergebnis ist ein segmentierter Datensatz. Jedes Segment enthält ein Terzspektrum.

Die x-Koordinate des Resultates zählt die Terzen (genauso wie die Famos-Funktion OctA).

Für eine sinnvolle Darstellung im Kurvenfenster ist dort die Terzbeschriftung zu wählen.

Die z-Koordinate des Ergebnisses enthält die Zeit. Der zeitliche Abstand zwischen den Segmenten delta-z entspricht dem Parameter "Ausgabe-Intervall".

Die Terzfilter und Bewertungen entsprechen IEC 651 (Schallpegelmesser), DIN 45652 (Terzfilter für elektroakustische Messungen) und EN61260-1:2014 bzw. IEC61260-1:2014 (Bandfilter für Oktaven und Bruchteile von Oktaven, Filterklasse 1).

Zusätzlich zum Einschwingen der Terzfilter hat das eingesetzte Hochpassfilter auch ein Einschwingen zur Folge.

Wenn f1=f2 ist, dann wird für diese Terz ein normaler Zeitdatensatz erzeugt, kein segmentierter.

Gesamte Schallintensität

Die Schallintensität des Eingangssignals wird bestimmt. Es erfolgt keine Terzfilterung. f2=-1 muss angegeben werden.

f1 ist die Frequenz, die für die intern ablaufende Hochpassfilterung benutzt wird. f1 liegt im Bereich [Abtastfrequenz/1e2] .. [Abtastfrequenz/1e6].

Nur Frequenzbewertung = linear wird unterstützt. Falls z.B. eine A-Bewertung gewünscht ist, muss zuvor ABCRating() für jedes der beiden Mikrofonsignale ausgeführt werden.

Das Einschwingen wird durch das Hochpassfilter bestimmt. Die Einschwingdauer ist umgekehrt proportional zu seiner Grenzfrequenz.

Das Ergebnis ist ein Zeitkanal. Das angegebene Ausgabe-Intervall ist seine Abtastzeit.

dB

Falls eine Umrechnung in dB gewünscht ist, kann sie nachträglich erfolgen.

Das Vorzeichen, das die Richtung ausdrückt, wird vorher bestimmt und ist separat zu behandeln. Siehe Beispiel.

Bei der dB Berechnung ist zu berücksichtigen, dass wie bei Leistung mit 10*log() gerechnet wird.

Bei der dB Berechnung ist zu berücksichtigen, dass der Betrag des Ergebnisses auch < 1E-12 sein kann, was zu negativen dB-Werten führt. Oft ist sinnvoll, diese z.B. mit UpperValue() auf 0 zu begrenzen.

Als Bezugswert für die dB Berechnung in Luft wird 1.a. 1E-12W/m^2 benutzt.

Hinweise

Die Laufzeit zwischen den beiden Mikrofonen ist entscheidender Bestandteil der Analyse. Alle Filter und die Phase beeinflussenden Vorverarbeitungen, die auf die Mikrofonsignale angewendet werden, müssen für beide Kanäle stets identisch ablaufen.

Bedingt durch die eingesetzten Filter gibt es einen Einschwingvorgang.

Während des Einschwingvorgangs werden die Intensitäten zu 0.0 angenommen.

Die Eingangsdaten dürfen Events haben.

Falls ein einziger Wert für die Schallintensität für die gesamte Messung gewünscht ist, wird Zeitbewertung=-6 gewählt. Der letzte Wert des Ergebnisses wird abgelesen.

Beispiele:

Aus 2 Mikrofonsignalen p1 und p2 soll die Schallintensität bestimmt werden. Mikrofonabstand ist 12mm.

Die Abtastfrequenz beträgt 50kHz. Alle 100ms wird ein Ergebnis für die Terzen von 50Hz bis 5kHz mit A-Bewertung gewünscht.

Das Vorzeichen für die Richtung soll separat berechnet werden. Die Schallintensität soll in dB angegeben werden.

```
omega_t = ramp(0,2e-5,50000)*(500*PI2)
p1=sin(omega_t)
```

```
p2=sin(omega_t-0.1)
in10c = SoundIntensityThirds(p1, p2, 50, 5000, 1, -5, 0.1, 0.012, 1.204, 0)
Sign = signum(in10c)
in10c_dB = 0.5 * dB(in10c / 1E-12)
```

P1 ist näher an der Quelle. Also Vorzeichen für die Richtung positiv

Aus 2 Mikrofonsignalen p1 und p2 soll die Schallintensität bestimmt werden. Hochpass bei 3Hz, Signal ab 50Hz interessant.

Anschließend Umrechnung in dB, wobei das Vorzeichen am dB-Wert die Richtung angeben soll.

```
omega_t = ramp(0,1e-5,100000)*(1000*PI2)
p1=sin(omega_t)
p2=sin(omega_t-1*PI2/360)
in10c = SoundIntensityThirds(p1, p2, 3, -1, 0, -5, 0.1, 0.012, 1.204, 0)
in10c_dB_sign = signum(in10c) * UpperValue(0.5 * dB(in10c / 1E-12), 0)
```

Siehe auch:

SpecThirds

SpeakConfig

Konfiguration der Sprachausgabe mit der Funktion SpeakText().

Deklaration:

```
SpeakConfig ( TxStimme [, TxWunschSprache] [, TxWunschGeschlecht] [, TxWunschAlter] [, EwLautstärke] [, EwGeschwindigkeit] ) -> EwErfolg
```

Parameter:

TxStimme	Name der auszuwählenden Sprechstimme. Bei einem leeren Text wird die Sprechstimme entsprechend der nächsten 3 Parameter automatisch bestimmt.
TxWunschSprache	Name der bevorzugten Ausgabesprache. Wird nur berücksichtigt, wenn [TxStimme] leer ist. Syntax des Parameters siehe unten, typische Angaben sind z.B. "en" oder "de" für englisch bzw. deutsch. (optional , Standardwert: "")
TxWunschGeschlecht	Bevorzugtes Geschlechts der Sprechstimme. Wird nur berücksichtigt, wenn [TxStimme] leer ist. (optional , Standardwert: 0)
	0 : egal
	1 : männlich
	2 : weiblich
	3 : neutral
TxWunschAlter	Bevorzugtes Alter der Sprechstimme. Wird nur berücksichtigt, wenn [TxStimme] leer ist. Standardmäßig sind unter Windows nur Stimmen normaler Altersfärbung (Erwachsener) installiert, so dass dieser Parameter im Allgemeinen auf 0 (=egal) belassen werden kann. (optional , Standardwert: 0)
	0 : egal
	1 : normal
	2 : älter
	3 : jugendlich
	4 : kindlich
EwLautstärke	Gibt die gewünschte Lautstärke für die Sprachausgabe an. Die Angabe erfolgt in Prozent der im System eingestellten Lautstärke des Standardgerätes für die Audio-Wiedergabe (erlaubter Wertebereich 0..100). (optional , Standardwert: 100)
EwGeschwindigkeit	Gibt die gewünschte Sprechgeschwindigkeit an. Der erlaubte Wertebereich ist von -10 (sehr langsam) bis 10 (sehr schnell). Der Standardwert ist 0 (normale Sprechgeschwindigkeit). (optional , Standardwert: 0)
EwErfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die Funktion dient zur Konfiguration der Sprachausgabe für nachfolgende Aufrufe der Funktion SpeakText().

Sie können die Sprechstimme wählen sowie die Laustärke und die Wiedergabegeschwindigkeit einstellen.

Standardmäßig werden die Einstellungen der Windows-Desktop-Systemsteuerung für die Text-in-Sprache-Umwandlung verwendet ("Systemsteuerung"->"Spracherkennung"->"Text-zu-Sprache").

Auswahl der Sprechstimme

Für die Auswahl der Sprechstimme haben Sie 2 prinzipielle Möglichkeiten:

Explizite Vorgabe des Namens im Parameter [TxStimme]

Der Name wird unter den installierten Stimmen gesucht, die nächsten 3 Parameter der Funktion werden dann ignoriert. Wenn keine Stimme mit diesem Namen vorhanden ist, bricht die Funktion ab und liefert 0 zurück. Die Liste der vorhandenen Stimmen finden Sie in der Windows-Desktop-Systemsteuerung ("Systemsteuerung"->"Spracherkennung"->"Text-zu-Sprache"). Die Einträge in der Liste "Stimmenauswahl" haben die Form "Name - Sprache". Wenn ein Listeneintrag z.B. "Microsoft Zira Desktop - English (United States)" lautet, so ist der hier anzugebene Name "Microsoft Zira Desktop". Zu Empfehlen ist die Verwendung des FAMOS-Funktions-Assistenten, der die verfügbaren Sprachen auflistet. Geben Sie "Standard" ein, um die in der Systemsteuerung gewählte Standardstimme wieder herzustellen.

Automatische Bestimmung anhand der gewünschten Stimm-Charakteristik

Dazu ist [TxStimme] als leerer Text zu übergeben und die nächsten 3 Parameter entsprechend zu füllen. Höchste Priorität hat dabei die zu verwendende Sprache im Parameter [TxWunschSprache]. Wenn die gewünschte Sprache nicht realisiert werden kann, bleibt die zuvor eingestellte Stimme erhalten. Wenn mehrere Stimmen in der angegebenen Sprachen verfügbar sind, wird versucht, nacheinander die Vorgabe für Geschlecht und Alter der Stimme zu realisieren. Wenn dies nicht möglich ist, wird eine die Standard-Stimme in der gewählten Sprache verwendet.

Format von [TxWunschSprache]: Geben Sie einen leeren String an, um die aktuelle Systemsprache zu wählen. Ansonsten erfolgt die Angabe der

Sprache als Sprachkennung der Form "xx-yy". Die Kennung besteht aus 2 Zeichen für den Sprach-Code nach ISO 639 und (optional) 2 Zeichen für Land/Region- Code nach ISO 3166. Beispiele:

Kennung	Bedeutung
"de"	Deutsch
"en"	Englisch. Region egal.
"en-US"	English, Aussprache USA.
"ja"	Japanisch
"zh-CN"	Chinesisch - Volksrepublik China
"zh-TW"	Chinesisch - Taiwan

Wenn alle Parameter zur Stimmauswahl als leerer Text bzw. 0 übergeben werden, so wird die aktuell eingestellte Stimme nicht verändert und nur Laufstärke und ggf. Sprechgeschwindigkeit geändert.

Die hier gewählten Einstellungen bleiben erhalten bis zum nächsten Neustart einer Top-Level-Sequenz oder dem Menübefehl 'Neubeginn'.

Der Aufruf der Funktion stoppt eventuell noch laufende asynchrone Sprachausgaben, die vorher mittels SpeakText() gestartet wurden.

Als Alternative zu diesem Befehl kann der auszugebende Textes bei SpeakText() im SSML-Format angegeben werden. Mit SSML können Sie Sprachaspekte wie Sprache, Aussprache, Lautstärke und Sprechgeschwindigkeit durch Marken im Text explizit steuern.

Multithreading: Die Funktion wirkt global. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion teilen sich also ein gemeinsames Gedächtnis.

Beispiele:

Bei Texten, die nicht der Systemsprache entsprechen, ist die explizite Vorgabe der Sprache empfehlenswert:

```
; Englische Ausgabe:
SpeakConfig( "Microsoft Zira Desktop")
; oder z.B
; SpeakConfig( "", "en")
SpeakText("Sequence is finished")
; Deutsche Ausgabe:
SpeakConfig( "Microsoft Hedda Desktop")
; oder z.B
; SpeakConfig( "", "de")
SpeakText("Sequenz ist fertig.")
```

Eine Ausgabe in English wird gewünscht. Eine Frauenstimme wird bevorzugt:

```
SpeakConfig( "", "en", 2)
SpeakText("I prefer a female voice")
```

Die Lautstärke wird auf 50% der Systemlautstärke eingestellt, die Wiedergabe etwas langsamer als normal:

```
SpeakConfig( "", "", 0, 0, 50, -5)
SpeakText("Not so loud and slower, please!")
```

Die in der Systemsteuerung eingestellten Standardwerte werden wieder hergestellt:

```
SpeakConfig( "standard")
```

Siehe auch:

SpeakText

SpeakText

Sprachausgabe des angegebenen Textes über das Standardgerät für die Audio-Wiedergabe.

Deklaration:

```
SpeakText ( Text [, Modus] ) -> EwErfolg
```

Parameter:

Text	Der wiederzugebende Text
Modus	Ausgabe-Modus (optional , Standardwert: 0)
	0 : Die Ausgabe erfolgt asynchron, die Funktion kehrt sofort zurück, ohne das Ende der Sprachausgabe abzuwarten. Falls eine asynchrone Ausgabe gerade aktiv ist, wird der Text in eine Warteschlange eingereiht und ausgegeben, wenn die "älteren" Texte abgearbeitet sind.
	1 : Die Funktion wartet, bis die Ausgabe beendet ist. Die Ausgabe beginnt sofort, eventuell noch laufende asynchrone Ausgaben werden abgebrochen.
EwErfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Der angegebene Text wird über das Standardgerät für die Audio-Wiedergabe ausgegeben. Es wird die in Windows eingebaute 'Text-zu-Sprache'-Funktionalität (engl: 'Text to Speech', TTS) verwendet.

Standardmäßig wird die in der Windows-Desktop-Systemsteuerung vorgegebene Sprechstimme verwendet ("Systemsteuerung"->"Spracherkennung"->"Text-zu-Sprache"). Stimme, Lautstärke und Wiedergabegeschwindigkeit kann mit der Funktion SpeakConfig() geändert werden.

Eine hier gestartete asynchrone Wiedergabe wird beendet, wenn:

- SpeakText() mit leerem Text-Parameter aufgerufen wird
- SpeakText() mit Modus='synchron' aufgerufen wird
- SpeakConfig() aufgerufen wird
- eine TopLevel-Sequenz gestartet wird
- der Menüpunkt "Neubeginn" ausgewählt wird
- FAMOS beendet wird

Fehler, die erst verzögert bei der asynchronen Ausgabe auftreten, können naturgemäß nicht über den Rückgabewert der Funktion gemeldet werden.

Der auszugebende Text kann unstrukturiert oder im **SSML-Format** (Speech Synthesis Markup Language, Version 1.0, https://www.w3.org/TR/speech-synthesis) angegeben werden. Mit SSML können Sie Sprachaspekte wie Sprache, Aussprache, Lautstärke und Sprechgeschwindigkeit explizit steuern. Die Funktion erkennt SSML automatisch, dazu wird das Vorhandensein der Zeichenkette '<speak' innerhalb der ersten 100 Zeichen geprüft.

Beispiel für den typischen Kopf einer SSML-Zeichenkette:

```
<?xml version="1.0"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
```

Die 'speak'-Marke muss mindestens die Attribute 'version' und 'xml-lang' enthalten. Die nicht unterstrichenen Teile sind hier optional.

Beispiele:

Das Ende einer längeren Auswertung wird durch eine entsprechende Audio-Meldung mitgeteilt und ein berechnetes Ergebnis vorgelesen:

```
_max = ...
...
SpeakText("Sequence is finished.")
SpeakText("The calculated maximum is "+ TForm( _max,""))
```

Bei Texten, die nicht der Systemsprache entsprechen, ist die explizite Vorgabe der Sprache empfehlenswert:

```
; Englische Ausgabe:
SpeakConfig( "Microsoft Zira Desktop")
; oder z.B
; SpeakConfig( "", "en")
SpeakText("Sequence is finished.")
; Deutsche Ausgabe:
SpeakConfig( "Microsoft Hedda Desktop")
; oder z.B
; SpeakConfig( "", "de")
```

```
SpeakText("Sequenz ist fertig.")
```

Synchrone Ausgabe eines englischen Textes per SSML. Der Test besteht aus 2 Sätzen. Zwischen beiden Sätzen wird eine Pause eingelegt, der 2. Satz wird mit hoher Geschwindigkeit wiedergegeben:

```
; SSML Kopf- und Fußmarke für englische Sprache: <speak version="1.0" xml:lang="en">...</speak>
ssmlHeader = "<speak version=~0341.0~034 xml:lang=~034en~034>"
ssmlFooter = "</speak>"
; Die Marke <p>...</p> markiert einen Absatz im Text:
ssmlText = "<p>This is the first paragraph. There should be a pause after this text is spoken.</p>"
; Die Marke <prosody rate="fast">...</prosody> bewirkt eine hohe Wiedergabegeschwindigkeit:
ssmlText = ssmlText+ "<p><prosody rate=~034fast~034>Speak faster.</prosody></p>"
SpeakText(ssmlHeader + ssmlText + ssmlFooter, 1)
```

Hier wird SSML verwendet, um ein Datum auf englisch in Langform (ausgesprochener Monatsname) auszugeben:

```
ssmlHeader = "<speak version=~0341.0~034 xml:lang=~034en~034>"
ssmlFooter = "</speak>"
; Ansage eines Datums als "January 29, 2009" mittels <say-as type="date"..
ssmlText = "<say-as type=~034date:mdy~034>"+  "1/29/2009" + "</say-as>"
SpeakText(ssmlHeader + ssmlText + ssmlFooter)
```

Siehe auch:

SpeakConfig

Spec

Bestimmt ein aussagekräftiges Spektrum (Spektrallinien als Amplituden)
Alternativer Name: **Spek**

Deklaration:

```
Spec ( Daten [, EwFenster] [, EwModus] ) -> Spektrum
```

Parameter:

Daten	Datensatz, von dem das Spektrum berechnet werden soll [ND, KX].
EwFenster	Fenster-Funktion (optional , Standardwert: -1)
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman/Harris
	5 : Flat-Top
	-1 : Die globale Voreinstellung (Dialog 'Optionen'/'Funktionen', SetOption(), FFTOPTION) wird verwendet.
EwModus	Verfahren (optional , Standardwert: -1)
	0 : Radix 2: Die Länge des Datensatzes wird vor der Berechnung auf die nächstkleinere 2er-Potenz verringert (abgeschnitten).
	1 : Radix 2: Die Länge des Datensatzes wird vor der Berechnung auf die nächstgrößere 2er-Potenz erhöht (mit Nullen aufgefüllt).
	2 : Radix 2: Der Datensatz wird vor der Berechnung so nachabgetastet, dass die neue Datensatzlänge eine Zweierpotenz wird. Entspricht einer vorherigen Anwendung der Funktion Red2().
	3 : Mixed-Radix: Die Länge des Datensatzes muss ein Produkt von Potenzen der Zahlen 2, 3 und 5 sein. Andere Datensatzlängen können nicht verarbeitet werden.
	-1 : Die globale Voreinstellung (Dialog 'Optionen'/'Funktionen', SetOption(), FFTOPTION) wird verwendet. Kompatibel zu älteren FAMOS-Versionen (< V2022).
Spektrum	Diskretes Amplituden-Spektrum des Datensatzes.

Beschreibung:

Die Funktion Spec() bestimmt ein aussagekräftiges Spektrum, bei dem die Spektrallinien als Amplituden zu verstehen sind. Wird ein normaler (reeller) Datensatz übergeben, wird zunächst das Spektrum wie bei der Funktion FFT() bestimmt, anschließend aber gleich automatisch die benötigte Korrektur ausgeführt.

Das Spektrum gibt die Amplituden und Phasen der im Signal enthaltenen Harmonischen an. Die Werte des Spektrums sind nach ihrer Frequenz geordnet. Der allererste Wert gehört dabei zur Frequenz 0, er gibt den Mittelwert des Signals wieder.

Die übrigen Werte kennzeichnen die Grund- und Oberwellen des Signals. Ein cosinusförmiges Signal der Amplitude 1 erzeugt im Spektrum an der Stelle seiner Frequenz eine Amplitude von 1 und Phase von 0°. Dasselbe, aber sinusförmige Signal erzeugt ebenfalls eine Amplitude von 1, aber eine Phase von -90°.

Bei der Überlagerung von mehreren sinusförmigen Signalen sind an den entsprechenden Frequenzen im Spektrum die Amplituden und Phasen der einzelnen Komponenten wieder zu finden.

Wird ein komplexer Datensatz übergeben, wird nur eine Korrektur ausgeführt, indem alle mittleren Spektrallinien um den Faktor 2 angehoben werden. Dabei wird bei allen komplexen Datentypen inhaltlich dasselbe Ergebnis erzeugt. Hier wird allerdings vorausgesetzt, dass die letzte Harmonische des komplexen Datensatzes tatsächlich reell ist, wie es z.B. bei Anwendung der FFT auf Eingangsdaten mit gerader Länge der Fall ist. Bei einer FFT mit ungerader Eingangslänge (möglich beim Mixed-Radix-Verfahren) ist die letzte Harmonische komplex und auch die letzte Spektralline muss verdoppelt werden.

Es ist wichtig zu beachten, dass kein kontinuierliches, sondern nur ein diskretes Spektrum berechnet wird. Das bedeutet, dass das Spektrum nur für einige diskrete Frequenzen berechnet wird. Das sind alle die Frequenzen, bei denen eine ganzzahlige Anzahl von Perioden innerhalb des Berechnungsintervalls liegen. Wird die FFT auf einen Datensatz der Länge 0.5s angewendet, wird das Spektrum an den Frequenzen 0, 2Hz, 4Hz, 8Hz ... bestimmt.

Ein wichtiges Prinzip in diesem Zusammenhang ist, dass ein diskretes Spektrum das Spektrum eines periodischen Signals ist. D. h. nun, dass bei der Berechnung des Spektrums eines Signals das Signal so interpretiert wird, als wäre es viele Male nach vorn und hinten aneinandergereiht. Stellt das Signal einen einzigen Impuls dar, so wird streng genommen nicht das Spektrum dieses einmaligen Impulses, sondern das einer Kette von Impulsen berechnet. Dieses periodische Aneinanderreihen macht sich besonders bemerkbar, wenn eine nichtganze Zahl von Perioden eines Signals vorliegen. Ein Datensatz enthält zum Beispiel 3.5 Perioden eines sinusförmigen Signals. Als Spektrum wird eigentlich eine einzige Harmonische erwartet. Nun aber stelle man sich das Signal periodisch fortgesetzt vor. Das ergibt nun insgesamt keinen sinusförmigen Verlauf, sondern nur eine Reihe sinusförmiger Stücke, die nicht richtig aneinandergereiht zu sein scheinen. Es wird verständlich, dass das Spektrum nicht eine einzige, sondern viele Harmonische enthält, die in der Nähe der wirklichen Frequenz besonders ausgeprägt sind.

Diesen offensichtlichen Nachteil kann man auf zwei Arten beheben. Die erste Möglichkeit ist, vor der Berechnung des Spektrums einen geeigneten Ausschnitt auszuwählen, der eine ganze Anzahl von Perioden des Signals enthält. Das wird man meistens ohnehin bei periodischen Signalen machen. Die andere Möglichkeit ist die Anwendung einer Fensterfunktion. Es können auch beide Möglichkeiten kombiniert werden.

Eine Fensterfunktion gewichtet die Werte eines Datensatzes unterschiedlich stark. Die Randwerte werden besonders schwach, die mittleren Werte besonders stark bewertet.

Sechs verschiedene Fensterfunktionen stehen zur Auswahl:

- Rechteck
- Hanning
- Hamming
- Blackman
- Blackman-Harris.
- Flat Top

Das Rechteckfenster bewertet alle Werte gleich. Es ist die Standardeinstellung, bei der keine Extraberechnung erforderlich ist. Das Flat- Top-Fenster am Ende der Liste ist das andere Extrem. Die Wirkung der Fensterung steigt in dieser Liste von oben nach unten. Der Effekt der Fensterung wird am Beispiel der 3.5-periodigen Sinusfunktion erläutert.

Um die eigentliche Frequenz herum erscheinen im Spektrum weitere Harmonische, deren Amplitude mit zunehmender Entfernung von der eigentlichen Frequenz abnimmt. Das Rechteckfenster erzeugt nun eine deutliche, recht dünne Spitze bei 3 bzw. 4. Aber auch in größerer Entfernung ist noch ein sehr starker verfälschender Einfluss auf das Spektrum vorhanden. Bei Anwendung der anderen Fenster wird die Spitze zunehmend breiter, aber in größerer Entfernung von der eigentlichen Frequenz klingt der Fehler immer schneller ab. Jede Fensterfunktion stellt also nur einen Kompromiss dar.

Um sehr nahe beieinander liegende Harmonische unterscheiden zu können, ist ein Rechteckfenster angemessen. Um mehrere nicht so nahe beieinander liegende Harmonische genau in ihren Amplituden und Phasen zu bestimmen, kann das Blackman-Fenster benutzt werden.

Unterschiede zur Funktion FFT():

Die Funktion FFT() bestimmt ein Spektrum nach dem FFT-Algorithmus, wobei die einzige automatisch ausgeführte Nachverarbeitung das Abschneiden der negativen Seite des Spektrums ist.

Um nun die Spektrallinien nun als Amplituden von Schwingungen interpretieren zu können, führt die Funktion Spec() eine gewisse Korrektur des Spektrums durch. Dazu werden alle komplexen Harmonischen mit dem Faktor 2 multipliziert.

Bei gerader Länge des Eingangsdatensatzes werden alle Spektrallinien, mit Ausnahme der ersten und letzten, mit dem Faktor 2 multipliziert.

Ist die Länge des Eingangsdatensatzes ungerade (nur möglich beim Mixed-Radix-Verfahren), ist die letzte Harmonische ebenfalls komplex, und es werden alle Spektrallinien mit Ausnahme der ersten mit dem Faktor 2 multipliziert.

- Wenn Sie mit einem Spektrum eine weitere Verarbeitung durchführen wollen, sollten Sie i. a. die Funktion FFT() benutzen. Eine Anwendung der inversen FFT oder eine Multiplikation liefern falsche Ergebnisse, wenn Sie Spektren mit der Funktion Spec() erzeugt haben.
- Im **Spektralpaket** (enthalten in Professional/Enterprise-Edition) finden Sie noch weitere, wesentlich leistungsfähigere Funktionen und auf die Bedürfnisse der Signalanalyse zugeschnittene Funktionen zur Berechnung von Spektren. Beispiel: **AmpSpectrumRMS()** berechnet ein Amplitudenspektrum (Harmonische als Effektivwerte), mit gleitendem Fenster und linearer Mittelung.
- Die optionalen Parameter für Fenster-Funktion und Modus werden seit ab FAMOS 2022 unterstützt. Wenn diese Parameter weggelassen werden, gelten wie in früheren Versionen die globalen Einstellungen, die über den Dialog 'Optionen'/'Funktionen', mit dem Kommando FFTOPTION oder der Funktion SetOption() eingestellt wurden.
- Mit der Funktion Spec() können Sie nachträglich jedes mit der Funktion FFT() bestimmte Spektrum bearbeiten, um die Amplituden richtig ablesen zu können.
- Die optionalen Parameter für Fenster-Funktion und Modus werden seit der FAMOS-Version 2022 unterstützt. Wenn diese Parameter weggelassen werden, gelten wie in früheren Versionen die globalen Einstellungen, die über den Dialog 'Optionen'/'Funktionen', mit dem Kommando FFTOPTION oder der Funktion SetOption() eingestellt wurden.
- Wenn die Länge des als Parameter übergebenen Datensatzes 2^27 überschreitet, wird eine entsprechende Fehlermeldung ausgegeben. Es ist dann keine Berechnung möglich. Verkürzen Sie den Datensatz mit den Funktionen Leng() oder Red2(). Die maximal bearbeitbare Punktezahl ist 134.217.728. Dann wird ein komplexer Datensatz der Länge 67.108.865 erzeugt.
- Die Funktion FFT, die im Falle eines übergebenen, normalen Datensatzes von der Funktion Spec aufgerufen wird, benötigt während ihrer Ausführung im Arbeitsspeicher temporären Speicherplatz. Ist (vor allem bei längeren Datensätzen) nicht ausreichend Speicherplatz vorhanden, wird eine Fehlermeldung erzeugt.
- Die hier implementierte Funktion Spec liefert nur die positive Seite des Spektrums. Bei den vorliegenden reellen Datensätzen ist das Spektrum stets konjugiert komplex symmetrisch, so dass die andere Hälfte keine zusätzlichen Informationen enthält. Zudem ist die Darstellung von negativen Frequenzen oder von Frequenzen oberhalb der halben Abtastfrequenz sehr fragwürdig.
- Hat der übergebene Datensatz N Punkte und ist N gerade, so hat das erzeugte Spektrum N/2 + 1 Punkte. Da der erste und letzte Wert (Mittelwert und höchste Harmonische) stets nur reell sind, die anderen Spektrallinien aber komplex, folgt, dass im Spektrum genauso viele signifikante Zahlen enthalten sind wie im Datensatz, also keine Information verloren gegangen ist. Ist N ungerade (nur möglich beim Mixed-Radix-Verfahren), so hat das erzeugte Spektrum (N-1)/2 + 1 Punkte. Die höchste Harmonische ist dann komplex.
- Das **Mixed-Radix-Verfahren** hat den Vorteil, dass man zu Datensätzen mit technisch üblichen Abtastfrequenzen (z.B. 1kHz und Vielfache im 1/2/5-Raster) und geeigneter Wahl der Eingangslänge Spektren mit "runden" Frequenzlinienabständen erhält. Ein mit 1kHz abgetasteter Datensatz liefert z.B. bei einer Länge von 1000 Samples (1000 = 2^3 * 5^3) ein Spektrum mit 1Hz Frequenzlinienabstand, bei einer Länge von 20000 Samples (= 2^5 * 5^4) einen Frequenzlinienabstand von 50mHz.
- Als x-Einheit des Spektrums wird stets Hz gewählt. Die y-Einheit des Betrages entspricht der des Datensatzes. Die Phase wird in Grad angegeben.
- Der x-Offset des übergebenen Datensatzes sollte Null sein. Ist er nicht Null, wird eine entsprechende Warnung erzeugt. Der x-Offset wird stets zu Null angenommenen. Möchten Sie den Einfluss eines x-Offsets dennoch in der Phase berücksichtigt haben, müssen Sie nachträglich

die Phase mit einer linearen Funktion zur Korrektur beaufschlagen.

Beispiele:

```
MPspec = Spec(NDdata, 1, 0)
```

Bestimmung des Spektrums eines Datensatzes, um aus dem Spektrum die Amplituden der Oberwellen des Signals ablesen zu können (Hammingfenster, ggf. abschneiden auf davorliegende 2er-Potenz.)

```
First1000Samples = CutIndex(data1kHz, 1, 1000)
Spec_with_1Hz_distance = Spec(First1000Samples, 0, 3)
```

Das Amplitudenspektrum der ersten 1000 Samples eines mit 1kHz abgetasteten Datensatzes wird berechnet. Das Ergebnis hat 501 Spektrallinien an den Frequenzen 0, 1, 2, ... 500 Hz.

```
; Achtung bei ungerader Eingangslänge:
First125amples = CutIndex(data1kHz, 1, 125)
Spec_with_8Hz_distance = Spec(First125Samples, 0, 3)
```

Das Amplitudenpektrum der ersten 125 Samples eines mit 1kHz abgetasteten Datensatzes wird berechnet. Das Ergebnis hat 63 Spektrallinien an den Frequenzen 0, 8, 16,... 496 Hz. Die letzte Spektrallinie ist komplex und liegt nicht bei der halben Abtastfrequenz.

```
maxValue = Max(Cmpl(Spec(NDdata, 0, 1)))
```

Bestimmung der größten Oberschwingung im Spektrum

```
NDwrong = iFFT( Spec(NDdata))
```

Achtung, grober Fehler! Die inverse FFT darf nur auf nicht amplitudenkorrigierte Spektren angewendet werden, wie sie von der Funktion FFT erzeugt werden.

Siehe auch:

FFT, iFFT, Red2, AmpSpectrumPeak, AmpSpectrumPeak_1, AmpSpectrumRMS_1

SpecThirds

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Das Terz-Spektrum wird aus dem Zeitverlauf eines Schwingungssignals in Abhängigkeit von der Zeit bestimmt.

Deklaration:

```
SpecThirds ( Schwingung, f1, f2, Frequenzbewertung, Zeitbewertung, Ausgabe-Intervall ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals, die Zeit in Sekunden skaliert
f1	Mittenfrequenz der untersten Terz in Hz
f2	Mittenfrequenz der obersten Terz in Hz
Frequenzbewertung	Mit welcher Frequenzbewertung wird das Resultat gewichtet?
	0 : linear
	1 : A-Bewertung
	2 : B-Bewertung
	3 : C-Bewertung
	4 : D-Bewertung
Zeitbewertung	Mittelungsdauer, Zeitbewertung der gefilterten Zeitdaten, wie wird der gleitende Effektivwert gebildet.
	-1 : Fast (0.125s)
	-2 : Slow (1s)
	-3 : Impuls
	-4 : Spitze
	> 0.0 : Frei definierte Mittelungsdauer, angegeben in Sekunden
Ausgabe-Intervall	In diesem zeitlichen Abstand werden die Terz-Spektren bestimmt. Ganzzahliges Vielfaches der Abtastzeit des Schwingungssignals. Angegeben in Sekunden.
Ergebnis	Terzspektrum abhängig von der Zeit

Beschreibung:

Die beiden Frequenzgrenzen f1 und f2 sollten als Terzmittenfrequenz angegeben werden, z.B. f1 = 8 Hz und f2 = 12500 Hz. f1 < f2. Die oberste Terz muss mit ihrem Frequenzband vollständig innerhalb der halben Abtastfrequenz liegen.

Die einzelnen Werte der Terzen sind als Effektivwerte angegeben.

Während zu Beginn der Messung einmalig die einzelnen Terz-Filter (Bandpässe) einschwingen, werden die Werte des Eingangssignals ignoriert. Das Einschwingen wird bei der 1kHz Terz zu 20ms angenommen. Diese Dauer ist umgekehrt proportional zur Terzfrequenz. Bei sehr niedrigen Terzen wird diese Dauer beachtlich. Eine entsprechend lange dauernde Messung ist dann vorzusehen.

Während der Einschwingphase werden die Effektivwerte zu 0.0 angenommen.

Das Ergebnis ist ein segmentierter Datensatz. Jedes Segment enthält ein Terzspektrum. Die x-Koordinate des Resultates zählt die Terzen (genauso wie die Famos-Funktion OctA). Für eine sinnvolle Darstellung im Kurvenfenster ist dort die Terzbeschriftung zu wählen.

Die z-Koordinate des Ergebnisses enthält die Zeit. Das erste Terzspektrum entsteht durch Ablesen der gleitenden Effektivwerte nach Ablauf der Zeit, die im Parameter "Ausgabe-Intervall" angegeben ist.

Die Terzfilter und Bewertungen entsprechen IEC 651 (Schallpegelmesser), DIN 45652 (Terzfilter für elektroakustische Messungen) und EN61260-1:2014 bzw. IEC61260-1:2014 (Bandfilter für Oktaven und Bruchteile von Oktaven, Filterklasse 1).

Zeitbewertung "Impuls": Bei ansteigender Amplitude beträgt die Zeitkonstante 35ms, bei abfallender Amplitude 1.5s. Damit werden impulsförmige Signale schnell erfasst, die Anzeige klingt langsam ab.

Zeitbewertung "Spitze": Extreme Anzeige für ganz kurze Impulse, wobei garantiert der Spitzenwert gezeigt wird. Bei ansteigender Amplitude Zeitkonstante null (ist im Computer exakt machbar, in Analogschaltung nur näherungsweise). Bei abfallender Amplitude 3s.

Beispiele:

Aus dem zeitlichen Verlauf einer Schwingung vib soll das Terz-Spektrum alle 0.1s bestimmt werden. Die Abtastzeit des Schwingungssignals ist 0.025 ms.

```
fEval = 1 ; 0 (linear) 1 (A-Bewertung)
f1 = 10
f2 = 12500
tEval = -1 ; -1 (Fast)
```

```
tInterval = 0.1 ; [s]
Thirds = SpecThirds ( vib, f1, f2, fEval, tEval, tInterval )
```

Siehe auch:

OctA, OtrRpmThirds, SpecThirds_1

SpecThirds_1

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Das gemittelte Terz-Spektrum wird aus dem Zeitverlauf eines Schwingungssignals bestimmt.

Deklaration:

```
SpecThirds_1 ( Schwingung, f1, f2, Frequenzbewertung ) -> Ergebnis
```

Parameter:

Schwingung	Zeitverlauf des Schwingungssignals, die Zeit in Sekunden skaliert
f1	Mittenfrequenz der untersten Terz in Hz
f2	Mittenfrequenz der obersten Terz in Hz
Frequenzbewertung	Mit welcher Frequenzbewertung wird das Resultat gewichtet?
	0 : linear
	1 : A-Bewertung
	2 : B-Bewertung
	3 : C-Bewertung
	4 : D-Bewertung
Ergebnis	Terzspektrum

Beschreibung:

Die beiden Frequenzgrenzen f1 und f2 sollten als Terzmittenfrequenz angegeben werden, z.B. f1 = 8 Hz und f2 = 12500 Hz. f1 < f2. Die oberste Terz muss mit ihrem Frequenzband vollständig innerhalb der halben Abtastfrequenz liegen.

Die einzelnen Werte der Terzen sind als Effektivwerte angegeben.

Während zu Beginn der Messung einmalig die einzelnen Terz-Filter (Bandpässe) einschwingen, werden die Werte des Eingangssignals ignoriert. Das Einschwingen wird bei der 1kHz Terz zu 20ms angenommen. Diese Dauer ist umgekehrt proportional zur Terzfrequenz. Bei sehr niedrigen Terzen wird diese Dauer beachtlich. Eine entsprechend lange dauernde Messung ist dann vorzusehen.

Während der Einschwingphase werden die Effektivwerte zu 0.0 angenommen.

Das Ergebnis ist ein normaler Datensatz. Die x-Koordinate des Resultates zählt die Terzen (genauso wie die Famos-Funktion OctA). Für eine sinnvolle Darstellung im Kurvenfenster ist dort die Terzbeschriftung zu wählen.

Die Terzfilter und Bewertungen entsprechen IEC 651 (Schallpegelmesser), DIN 45652 (Terzfilter für elektroakustische Messungen) und EN61260-1:2014 bzw. IEC61260-1:2014 (Bandfilter für Oktaven und Bruchteile von Oktaven, Filterklasse 1).

Beispiele:

Aus dem zeitlichen Verlauf einer Schwingung vib soll das Terz-Spektrum bestimmt werden. Die Abtastzeit des Schwingungssignals ist 0.025 ms.

```
fEval = 1 ; 0 (linear) 1 (A-Bewertung)
f1 = 10
f2 = 12500
Thirds = SpecThirds_1 ( vib, f1, f2, fEval )
```

Siehe auch:

OctA, OtrRpmThirds, SpecThirds

sqr

Quadrat
Alternativer Name: **Quad**

Deklaration:

```
sqr ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Parameter. Erlaubte Typen: [ND],[XY].
Ergebnis	Quadrat des Parameters

Beschreibung:

Der übergebene Parameter wird quadriert.

Anmerkungen

- Die y-Einheit wird quadriert.
- Die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Die Funktion sqr() arbeitet effektiver als das ausgeschriebene Produkt mit dem Operator * (Multiplikation).
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Die elektr. Leistung an einem ohmschen Widerstand ist das Spannungsquadrat, dividiert durch den Widerstand.

```
power = sqr(voltage) / resistance
```

Siehe auch:

sqrt, ^(hoch)

sqrt

Quadratwurzel
Alternativer Name: **Wurz**

Deklaration:

```
sqrt ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Parameter. Erlaubte Typen: [ND],[XY].
Ergebnis	Quadratwurzel des Parameters

Beschreibung:

Die Quadratwurzel wird berechnet.

Anmerkungen

- Aus der Einheit wird die Wurzel gezogen. Die Einheit muss dazu mindestens quadratische Terme enthalten.
- Die Wurzel ist nur definiert für Zahlen größer gleich Null.
- Die Wurzel kann auch als Potenz berechnet werden: sqrt(x) ist äquivalent zu (x ^ 0.5).
- Die x-Koordinate(n) von Parameter und Ergebnis sind gleich.
- Der Parameter darf strukturiert sein (Events/ Segmente).

Beispiele:

Die Wurzel aus 9 ist 3:

```
three = sqrt(9)
```

Siehe auch:

sqr, ^(hoch)

sRMS

Stat()-Funktion: Effektivwert eines Datensatzes
Alternativer Name: **sEff**

Beschreibung:

sRMS ist eine vordefinierte Variable und wird durch einen vorherigen Aufruf der Statistikfunktion Stat() belegt.

Alternativ kann die Funktion RMS() verwendet werden.

Beispiele:

Anzeige des Effektivwerts eines Spannungsverlaufs:

```
Stat(U)
BoxMessage("Effektivwert", "Ueff = " + TForm(sRMS, ""), "!1")
```

Dies ist äquivalent zu:

```
BoxMessage("Effektivwert", "Ueff = " + TForm(RMS(U), ""), "!1")
```

Siehe auch:

Stat, RMS, sMean

sStDev

Stat()-Funktion: Standardabweichung eines Datensatzes
Alternativer Name: **sStreu**

Beschreibung:

sStDev ist eine vordefinierte Variable und wird durch einen vorherigen Aufruf der Statistikfunktion Stat() belegt.

Alternativ kann die Funktion StDev() verwendet werden.

Beispiele:

Anzeige der Standardabweichung eines Spannungsverlaufs:

```
Stat(U)
BoxMessage("Standardabweichung", "Us = " + TForm(sStDev, ""), "!1")
```

Dies ist äquivalent zu:

```
BoxMessage("Standardabweichung", "Us = " + TForm(StDev(U), ""), "!1")
```

Siehe auch:

Stat, StDev, sMean

Stat

Bestimmt diverse statistische Kenngrößen eines Datensatzes und bewertet die vordefinierten Variablen sMin, sMax, sMean, sRMS, sStDev, sMinPos, sMaxPos.

Deklaration:

```
Stat ( Daten )
```

Parameter:

Daten	Datensatz, von dem statistische Kennwerte ermittelt werden sollen. [ND]

Beschreibung:

Die Statistikfunktion hat die Form einer Prozedur, sie liefert keinen Wert zurück und kann nicht mit einer Zuweisung benutzt werden.

FAMOS enthält eine Reihe von vordefinierten Variablen, die durch den Aufruf der Stat()-Funktion berechnet werden:

sMin	Minimum
sMax	Maximum
sMean	arithmetischer Mittelwert
sStDev	Standardabweichung
sRMS	Effektivwert, Wurzel des Mittels der Quadrate
sMinPos	Position (x-Wert) des Minimums
sMaxPos	Position (x-Wert) des Maximums

Die Statistikfunktion berechnet diese vordefinierten Variablen bei jedem Aufruf neu. Damit haben Sie mit einem Aufruf viele wichtige statistische Kennwerte des Datensatzes verfügbar, die Sie dann in weiteren Formeln weiterverarbeiten können.

- Die statistischen Kenngrößen werden von der Statistikfunktion mit entsprechenden Einheiten erzeugt.
- Die vordefinierten Variablen sMin, sMax usw. dürfen in allen Formeln benutzt werden, sie tauchen aber nicht in der Variablenliste auf. D. h. sie können nicht von Hand oder durch eine Zuweisung verändert werden. Ihnen können allein von der Statistikfunktion Werte zugewiesen werden. Die vordefinierten Variablen sind bei Programmanfang zu Null initialisiert.
- Die vordefinierten Variablen werden bei jedem Aufruf der Statistikfunktion neu berechnet. Damit gehen die alten Werte verloren. Benutzen Sie also stets alle interessierenden Größen, bevor Sie die Statistikfunktion erneut aufrufen.
- Die Statistikfunktion zeigt die berechneten Werte außerdem im Ausgabefeld im imc FAMOS Hauptfenster an.
- Sie können die Ergebnisse, die die Funktion Stat() liefert, auch mit anderen Funktionen einzeln berechnen.
  Multithreading: Die Funktion wirkt lokal auf den Ausführungs-Thread, in dem sie aufgerufen wurde. Der Standard-Ausführungs-Thread und jede mittels BEGIN_PARALLEL in einem eigenen Thread ausgeführte Sequenzfunktion haben also jeweils ein eigenes, unabhängiges Gedächtnis.

Beispiele:

```
Stat(NDdata)
specificValue = (sMax - sMin) / sMean
```

Diese beiden Formeln berechnen eine Kenngröße aus Standardgrößen des Datensatzes, wie Minimum und Maximum. Der Aufruf ist äquivalent zu

```
specificValue = (Max(NDdata) - Min(NDdata)) / Mean(NDdata)
```

```
Stat(NDdata)
delta1 = sMax - sMean
Stat(NDdata * 2)
delta2 = sMax - sMean
```

Beachten Sie, dass delta1 und delta2 unterschiedlich sind, da die Statistikfunktion erneut für einen anderen Datensatz, nämlich einen doppelt so großen, aufgerufen wird.

Siehe auch:

Min, Max, Mean, Pos, PosiEx, RMS, StDev

StDev

Die Standardabweichung der Zahlenwerte eines Datensatzes wird ermittelt.
Alternativer Name: **Streu**

Deklaration:

```
StDev ( Daten ) -> EwStandardabweichung
```

Parameter:

Daten	Daten, von denen die Standardabweichung berechnet werden soll [ND].
EwStandardabweichung	Standardabweichung des Datensatzes

Beschreibung:

Die Standardabweichung ist eine statistische Kenngröße eines Datensatzes. Sie gibt an, wie weit die Daten um ihren arithmetischen Mittelwert gestreut sind. Datensätze mit sehr ähnlichen Zahlenwerten haben also eine kleine Standardabweichung, während stark verschiedene Zahlenwerte eine große Standardabweichung zur Folge haben.

Die Standardabweichung wird nach folgendem Algorithmus berechnet:

Die Abweichung eines jeden Wertes des Datensatzes zum arithmetischen Mittelwert des Datensatzes wird quadriert und für alle Werte des Datensatzes summiert. Dieser Wert wird durch die Anzahl der Werte des Datensatzes minus 1 dividiert. Daraus wird die Quadratwurzel gezogen.

Beispiele:

Ein Datensatz mit den Werten 1, 3, 5 hat die Standardabweichung 2.

```
data = [1, 3, 5]
SD = StDev(data)
```

Siehe auch:

Mean, Min, Max, Stat

STri

Schmitt-Trigger mit Vorgabe des oberen und unteren Schwellwertes

Deklaration:

```
STri ( Daten, EwOben, EwUnten ) -> Ergebnis
```

Parameter:

Daten	Zu formender Datensatz. Erlaubte Typen: [ND].
EwOben	Obere Schwelle
EwUnten	Untere Schwelle
Ergebnis	Rechteckförmiges Ergebnis des Schmitt-Triggers

Beschreibung:

Die Schmitt-Triggerfunktion STri() dient zum Formen von Impulsen. Die Funktion sTri() erzeugt ideal geformte, rechteckförmige Impulse. Sie kann also nur zwei verschiedene Funktionswerte erzeugen, nämlich +1 und -1. Zu ihrer Konfiguration benötigt die Funktion zwei Schwellwerte, einen unteren und einen oberen.

Der Schmitt-Trigger arbeitet nach folgendem Algorithmus:

- Der Schmitt-Trigger liefert als Startwert -1, wenn der erste Wert des Signals kleiner als der obere Schwellwert ist. Ansonsten ist der Startwert +1.
- Wenn der Schmitt-Trigger zuletzt eine -1 geliefert hat und das Signal kleiner als der obere Schwellwert ist, liefert er wieder eine -1, ansonsten eine +1.
- Wenn der Schmitt-Trigger zuletzt eine +1 geliefert hat und das Signal größer als der untere Schwellwert ist, liefert er wieder eine +1, ansonsten eine -1.
- Die beiden letzten Schritte werden wiederholt, bis der ganze Datensatz durchlaufen ist. Es fällt auf, dass die Schmitt-Triggerfunktion ein Gedächtnis hat.

Die Schmitt-Triggerfunktion wird eingesetzt, um verrauschte (mit Störungen behaftete) Impulse für eine Weiterverarbeitung zu formen. Dabei wird alles Rauschen eliminiert. Allerdings geht die Signalform verloren. Je größer die Differenz zwischen den Schwellwerten ist, desto unempfindlicher ist der Schmitt- Trigger gegen Störungen.

Es lohnt sich, den Schmitt-Trigger anzuwenden, wenn es bei einer Impulsfolge auf die Häufigkeit der Impulse ankommt und deren Lage zueinander, nicht aber auf deren Form.

- Das Schmitt-Trigger-Signal hat keine y-Einheit, stimmt aber in der x-Skalierung mit dem übergebenen Datensatz überein.
- Bei der grafischen Darstellung von Schmitt-Trigger-Signalen oder anderen rechteckförmigen Kurven sollten Sie eine Treppenstufendarstellung wählen.
- Kurze, aber sehr hohe Störimpulse können oft mit dem Schmitt-Trigger nicht unterdrückt werden, wenn der Abstand zwischen den Schwellwerten nicht so groß sein kann. Benutzen Sie dann die Funktionen Smo3(), Smo5() oder Smo() zu einem Vorabglätten, oder ändern Sie die einzelnen Zahlenwerte von Hand.
- Wenn Ober- und Untergrenze gleich sind, arbeitet der Schmitt-Trigger als Komparator, d. h. er stellt fest, ob ein Signal größer oder kleiner einem bestimmten Wert ist.
- Der 2. und 3. Parameter dürfen vertauscht werden. Es wird stets der kleinere als Untergrenze angenommen.

Beispiele:

```
peakCount = Peaks(STri(NDdata, 10 'V', 20 'V'))
```

Ein Signal hat i. a. einen Pegel von 3V ... 8V, aber einige wichtige Spitzen, die 25V ... 30V hoch sind. Es ist die Zahl dieser Spitzen festzustellen. Die Schwellwerte werden zwischen 8V und 25V mit gewissen Sicherheiten gesetzt, jedoch wiederum recht groß, um Störungen an den Flanken der Spitzen zu unterdrücken. Die Schmitt-Triggerfunktion idealisiert die Spitzen zu rechteckigen Impulsen, die anschließend gezählt werden.

Siehe auch:

Hyst, Peaks, Scale, Smo, Perio

Sub

Zeit- bzw. x-richtige Subtraktion

Deklaration:

```
Sub ( Minuend, Subtrahend, EwOption ) -> Differenz
```

Parameter:

Minuend	Erster Parameter, Minuend. Erlaubte Typen: [ND],[XY].
Subtrahend	Zweiter Parameter, Subtrahend. Erlaubte Typen: [ND],[XY].
EwOption	Option
	0 : Die Triggerzeit der beiden Summanden wird ignoriert
	1 : Zeitrichtige Überlagerung mit Berücksichtigung der Triggerzeit
Differenz	Differenz, Ergebnis der Subtraktion. [XY]

Beschreibung:

Zwei Datensätze werden zeit- bzw. x-richtig subtrahiert, d. h. es werden immer 2 y-Werte subtrahiert, die die gleiche Zeit bzw. den gleichen x-Wert besitzen.

Das Ergebnis ist nur für den x-Bereich definiert, den die beiden Parameter-Datensätze gemeinsam haben. In diesem Bereich wird überall dort ein Ergebnis-Wert berechnet, wo mindestens einer der beiden Datensätze eine Stützstelle besitzt. Wenn an dieser Stelle der andere Datensatz keine Stützstelle besitzt, wird dieser dort linear interpoliert.

Die x-Spur beider Parameter-Datensätze muss monoton steigend sein, d.h. die x-Koordinaten müssen kontinuierlich wachsen.

Der Operator - (Subtraktion) führt dagegen eine punktweise Subtraktion von Datensätzen aus.

Beispiele:

Zwei Kanäle werden zwischen 11 und 13 Uhr bzw. 12 und 14 Uhr aufgenommen.

```
difference = Sub(voltage11_13h, voltage12_14h, 1)
```

Eine zeitrichtige Subtraktion der beiden Datensätze mit Berücksichtigung der Triggerzeit wird ausgeführt. Das Ergebnis ist zwischen 12 und 13 Uhr definiert.

Siehe auch:

-(Subtraktion), Add, Mult, Div, Append

Sum

Summe über alle Werte eines Datensatzes
Alternativer Name: **Summe**

Deklaration:

```
Sum ( Daten ) -> EwSumme
```

Parameter:

Daten	Zu summierender Datensatz [ND, XY].
EwSumme	Summe aller y-Werte des Datensatzes

Beschreibung:

Es wird die Summe aller y-Werte des Datensatzes gebildet.

Beispiele:

Die beiden Formeln berechnen die Summe aller y-Werte eines äquidistant abgetasteten Datensatzes:

```
dataSum = Sum(data)
dataSum = Int(data) / xdel?(data)
```

Siehe auch:

Int, GInt

SvToChar

Ein Text mit einem beliebigen Zeichen wird erzeugt
Alternativer Name: **TZeichen**

Deklaration:

```
SvToChar ( EwZeichen ) -> TxZeichen
```

Parameter:

EwZeichen	Der ASCII-Code des Zeichens, das als Text darzustellen ist (0..255).
TxZeichen	Text (der Länge 1) mit dem angegebenen Zeichen.

Beschreibung:

Der als Parameter angegebene ASCII-Code eines Zeichens wird benutzt, um einen Text zu erzeugen, der genau dieses eine Zeichen enthält. Hilfreich für Sonderzeichen wie z.B. Tabulator und Zeilenumbrüche.

Zur Angabe von Sonderzeichen in Textkonstanten kann alternativ das Sonderzeichen "~" (Tilde), gefolgt von einer 3-stelligen Dezimalzahl mit dem ASCII-Code, verwendet werden.

- Zulässige ASCII-Codes sind 1..255.
- Wird ein ASCII-Code von 0 angegeben, wird ein leerer Text zurückgegeben.
- Ein Tabulator-Zeichen hat beispielsweise den Code 9, ein Carriage Return 13, ein Line Feed 10.
- Nicht alle ASCII-Codes können unter Windows angezeigt werden. Beachten Sie vor allem, dass der ANSI-Zeichensatz benutzt wird. ASCII-Tabellen, die z.B. für die DOS-Umgebung gemacht sind, stellen die Zeichen im OEM-Zeichensatz dar.

Beispiele:

Ein Text mit der Kombination Carriage Return/ Line Feed wird erzeugt:

```
TxCrlf = SvToChar(13) + SvToChar(10)
```

Das gleiche Ergebnis erhalten Sie mit

```
TxCrlf = "~013" + "~010"
```

oder noch kürzer:

```
TxCrlf = "~013~010"
```

Siehe auch:

CharToSv, TtoSv, TReplace, TConv

SWITCH

Leitet eine Fallunterscheidung (mehrfache Verzweigung) ein. Abhängig vom Ergebnis des hier angegebenen Ausdrucks wird maximal einer von mehreren nachfolgenden alternativen Anweisungsblöcken (CASE/DEFAULT) ausgeführt.

Deklaration:

```
SWITCH Vergleichswert
```

Parameter:

Vergleichswert	Vergleichswert für nachfolgende CASE-Abweisungen. Die Auswertung des hier angegebenen Ausdrucks muss entweder einen Einzelwert oder einen Text ergeben.

Beschreibung:

Nach dem SWITCH muss mindestens ein CASE oder DEFAULT-Befehl folgen, das Ende der Fallunterscheidung wird durch den Befehl END markiert.

Als CASE-Bedingung können Sie entweder einen einzelnen Wert, eine Komma-getrennte Liste von Werten oder einen Bereich (mit dem Schlüsselwort 'TO') angeben.

Das Konstrukt

```
SWITCH value
CASE c1
   ...Anweisungen1
CASE c2,c3
   ...Anweisungen2
CASE c4 TO c5
   ...Anweisungen3
DEFAULT
   ...Anweisungen4
END
```

ist äquivalent zu:

```
IF value = c1
   ...Anweisungen1
ELSEIF value = c2 OR value = c3
   ...Anweisungen2
ELSEIF value >= c4 AND value <= c5
   ...Anweisungen3
ELSE
   ...Anweisungen4
END
```

Beispiele:

Zu einem Wert, der normalerweise im Bereich von 0 bis 100 liegt, wird ein beschreibender Text gebildet.

```
SWITCH Round(value, 1)
CASE 0
   Tx = "Lower limit"
CASE 1 TO 48
    Tx = "Lower half"
CASE 49,50,51
    Tx = "Center"
CASE 52 To 99
   Tx = "Upper half"
CASE 100
   Tx = "Upper limit"
DEFAULT
   Tx = "Invalid Value"
END
```

Der Anwender wird aufgefordert, eine Datei zum Laden auszuwählen. Anhand der Dateierweiterung wird das Dateiformat erkannt und die entsprechende Funktion zum Laden aufgerufen.

```
TxFileName = DlgFileName("", "", "",0)
TxFileExt = FsSplitPath(TxFileName, 3)
SWITCH TxFileExt
CASE ".dat", ".raw"
   ; Load imc data file
   fh = FileOpenDSF(TxFileName, 0)
   ;...
CASE ".xls"
   ; Load EXCEL file
```

```
   fh = FileOpenXLS(TxFileName, 0)
   ;...
DEFAULT
   PAUSE ==> Invalid file format
END
```

Siehe auch:

CASE, DEFAULT, IF, CodeRange

TAdd

Zwei Texte werden aneinandergehängt

Deklaration:

```
TAdd ( TxVorn, TxHinten ) -> TxGesamt
```

Parameter:

TxVorn	Text, an den angehängt werden soll
TxHinten	Text, der angehängt werden soll
TxGesamt	Zurückgegeben wird die Aneinander-Reihung der beiden übergebenen Texte

Beschreibung:

Die beiden übergebenen Texte werden aneinandergehängt.

Alternativ kann auch der **Additions-Operator +** verwendet werden.

Beispiele:

```
Pfad = TAdd(TAdd("c:\", "imc\"), "dat")
; oder:
Pfad = "c:\" + "imc\" + "dat"
```

Nach Ausführung enthält die Variable Pfad den Text "c:\imc\dat".

Siehe auch:

TForm, TConv, TPart, TtoSv

tan

Tangens, trigometrische Funktion

Deklaration:

```
tan ( Parameter ) -> Ergebnis
```

Parameter:

Parameter	Eingangsdaten (Winkel). Erlaubte Typen: [ND],[XY].
Ergebnis	Tangens des Parameters

Beschreibung:

Es wird die trigonometrische Funktion tan berechnet, wobei im Bogen- oder Gradmaß entsprechend der Einheit des übergebenen Parameters gearbeitet wird.

Die x-Koordinate(n) von Ergebnis und Parameter sind gleich.

Anmerkungen

- Der Parameter der Funktion sollte keine Einheit oder die Einheiten 'Rad', 'Grad', '° ' haben. Bei anderen Einheiten wird eine Warnung erzeugt und die Einheit ins Ergebnis übernommen.
- Der Parameter darf strukturiert sein (Events/ Segmente).
- Die zugehörige Umkehrfunktion heißt atan().

Beispiele:

tan und atan kehren sich um:

```
one = tan(atan(1))
```

Zur Korrektur einer Messgröße, die einem Sättigungseffekt unterliegt, kann die tan-Funktion zum Strecken des Sättigungsbereichs eingesetzt werden:

```
NDcorr = 3 * tan(NDdata / 10)
```

Ein Winkel im Gradmaß wird als Parameter übergeben. Es gilt ferner: tan(PI/4) = 1, tan(0) = 0, tan(-PI/4) = -1 bzw. tan(45'Grad') = 1, usw.

```
one = tan(45'°')
```

Siehe auch:

sin, cos, atan2

TComp

Zwei Texte oder Textfelder werden miteinander verglichen
Alternativer Name: **TVerg**

Deklaration:

```
TComp ( TxText1, TxText2 ) -> Ergebnis
```

Parameter:

TxText1	Der erste der vergleichende Parameter
TxText2	Der zweite zu vergleichende Parameter
Ergebnis	Ergebnis des Vergleichs, Einzelwert oder Datensatz mit den folgenden Werten:
	-1 : Der erste Text ist alphabetisch vor dem zweiten Text anzuordnen
	0 : Beide Texte sind gleich
	1 : Der erste Text ist alphabetisch nach dem zweiten Text anzuordnen

Beschreibung:

Die folgenden Parameterkombinationen sind erlaubt:

Text - Text:	Das Ergebnis ist ein Einzelwert mit dem Wert -1, 0 oder 1.
Textfeld - Textfeld:	Es findet ein elementweiser Vergleich der beiden Textfelder statt. Das Ergebnis ist ein Datensatz, der aus -1, 0, 1 besteht und die Länge des kürzeren der beiden Textfelder hat.
Textfeld - Text:	Jedes Element des Textfeldes wird mit dem 2. Parameter verglichen. Das Ergebnis ist ein Datensatz, der aus -1, 0, 1 besteht und die Länge des Textfeldes hat.

- Zwischen Groß- und Kleinschreibung wird nicht unterschieden. Ein Vergleich der Texte "hallo" und "Hallo" liefert demzufolge eine 0 zurück.
- Ein leerer Text wird bei alphabetischer Sortierung vor alle anderen Texte gesetzt.
- Zum Vergleich von Texten und Textfeldern können auch die Operatoren "<>" und "=" verwendet werden (bei Textfeldern liefern diese Operatoren keinen elementweisen Vergleich, sondern prüfen das gesamte Feld auf Gleichheit).

Beispiele:

```
DatName = "wave0001"
WavVergl = TComp(TPart(DatName, 1, 4), "WAVE")
```

WavVergl erhält den Wert 0.

```
Txa1 = ["Channel1", "CHANNEL2", "Channel5"]
Txa2 = ["Channel2", "channel2", "Channel4", "Channel7"]
v = TComp(Txa1, Txa2)
v2 = TComp(Txa1, "channel5")
```

v enthält die Werte [-1, 0, 1].

v2 enthält die Werte [-1, -1, 0].

Siehe auch:

TLike, TForm, TPart, TReplace, TxWhere, TxRegExMatch

TConv

Ein Text kann auf verschiedene Weise konvertiert werden, z. B. Groß-Kleinschreibung.
Alternativer Name: **TKonv**

Deklaration:

```
TConv ( TxText, EwAuftrag ) -> TxKonvertiert
```

Parameter:

TxText	Zu konvertierender Text
EwAuftrag	Angabe, wie konvertiert werden soll
	1 : Alle Großbuchstaben werden in Kleinbuchstaben gewandelt
	2 : Alle Kleinbuchstaben werden in Großbuchstaben gewandelt
	3 : Ein Text wird vom ANSI- in den OEM-Zeichensatz übertragen. Wenn OEM-Texte in Dateien geschrieben werden, werden sie von DOS-Applikationen korrekt gelesen. Windows-Applikationen erwarten die Texte im ANSI-Format.
	4 : Alle Leerzeichen am Beginn des Textes werden entfernt.
	5 : Alle Leerzeichen am Ende des Textes werden entfernt
	6 : Alle Leerzeichen werden entfernt.
	7 : Ein Text wird vom OEM- in den ANSI-Zeichensatz übertragen.
TxKonvertiert	Konvertierter Text

Beschreibung:

Beispiele:

```
TxPfad= TConv(" c:\imc\ dat ",6)
```

TxPfad erhält den Inhalt "c:\imc\dat"

Siehe auch:

TForm, TPart, TComp, TtoSv

TForm

Eine Zahl wird in einen formatierten Text gewandelt.

Deklaration:

```
TForm ( EwZahl, TxFormat ) -> TxFormatiert
```

Parameter:

EwZahl	Zu wandelnde reelle Zahl.
TxFormat	Angabe des gewünschten Formates
TxFormatiert	Der formatierte Text

Beschreibung:

Eine Zahl wird in einen formatierten Text gewandelt, wobei verschiedene Formattypen vorgegeben werden können. Der Parameter [TxFormat] enthält die dazu nötigen Angaben.

Folgende Formate sind verfügbar, wobei die führenden Buchstaben die obligatorische Formatkennung sind, die anderen Buchstaben sind durch die gewünschten Zahlenwerte zu ersetzen:

TxFormat	Beschreibung	Beispiel
"e.N"	Fließkomma-Format N: Anzahl der Nachkommastellen.	3.456 mit "e.2" -> "3.46E+00"
"FV.N"	Festkomma-Format. V: Minimale Anzahl der Vorkommastellen. N: Anzahl der Nachkommastellen. Überzählige Vorkomma-Stellen werden mit Leerzeichen aufgefüllt.	3.456 mit "f2.2" -> " 3.46"
"gV.N"	Festkomma-Format. V: Anzahl der Vorkommastellen. N: Anzahl der Nachkommastellen. Überzählige Vorkomma-Stellen werden mit Nullen aufgefüllt.	3.456 mit "g2.2" -> "03.46"
"a.N"	Automatisch. N: Maximal ausgegebene Stellen. Je nach Zahlenwert wird die kürzere Darstellung gewählt. Abschließende Nullen nach dem Dezimalpunkt werden weggelassen, ggf. auch der Dezimalpunkt selbst. Somit ideal für ganze Zahlen.	3.40056 mit "a.2" -> "3.4"
""	Entspricht "a.6"	
"xG"	Hexadezimal-Format. G: Anzahl der Stellen insgesamt	127 mit "x2" -> "7F"
"x"	Hexadezimal-Format (für ganzzahlige Datenformate) [EwZahl] muss ganzzahlig und unskaliert sein.	127 (4 Byte ganzzahlig ohne Vorzeichen) -> "0000007F" -1 (4 Byte ganzzahlig mit Vorzeichen) -> "FFFFFFFF"
"bG"	Binär-Format. G: Anzahl der Stellen insgesamt	34.56 mit "b4" -> "100011"
"b"	Binär-Format (für ganzzahlige Datenformate). [EwZahl] muss ganzzahlig und unskaliert sein.	127 (1 Byte ganzzahlig ohne Vorzeichen) -> "01111111" -1 (1 Byte ganzzahlig mit Vorzeichen) -> "11111111"

Komma oder Punkt

Wenn in diesen Formatstrings der Punkt durch ein Komma ersetzt wird, erfolgt die Ausgabe mit einem Dezimalkomma statt einem Dezimalpunkt. Ansonsten kann der Punkt auch weggelassen werden (Beispiel: "f23" ist gleichwertig zu "f2.3", "f2,3" dagegen erzwingt ein Dezimalkomma.

Wenn in diesen Formatstrings der Punkt durch ein Semikolon ersetzt wird, erfolgt die Ausgabe entsprechend der globalen Voreinstellung für das bevorzugte Dezimaltrennzeichen für reelle Zahlen ('Extra'/'Optionen'/'Anzeige' bzw. SetOption("Display.DecimalSeparator",...).

Besonderheiten bei Gleitkommadarstellung:

Bei Gleitkomma-Darstellung wird der Exponent mit 2 Stellen angegeben und ein "E" als Zeichen für den Exponenten benutzt, gefolgt vom Vorzeichen.

Besonderheiten bei Festkommadarstellung:

Bei Festkomma-Darstellung wird mit Leerzeichen vor der Zahl aufgefüllt, bis die angegebene Zahl von Vorkomma-Stellen erreicht ist. Wenn sie kein Leerzeichen vor der Zahl wünschen, können sie die Anzahl der Vorkomma-Stellen auf 1 setzen. Hat die Zahl mehr Vorkomma-Stellen, als in [TxFormat] angegeben, werden trotzdem alle Vorkomma-Stellen erzeugt, damit die Zahl nicht verfälscht wird.

Rundung

Der ausgegebene Wert wird gegebenenfalls entsprechend der Anzahl der Nachkommastellen gerundet.

Besonderheiten bei Binär- oder Hexadezimalformat:

Die zu konvertierende Zahl muss im Zahlenbereich -2^63..2^63-1 liegen.

Bei negativen Zahlen wird die Zweierkomplement-Darstellung verwendet.

Für "xG" und "bG" ergibt sich die minimale Ausgabebreite nach:
Hex-Format: Bereich -2147483648..0: 8 Stellen, sonst 16 Stellen.
Binär-Format: Bereich -32768..0: 16 Stellen, -2147483648..-32769: 32 Stellen, sonst 64 Stellen.

"x" und "b" (ohne Breitenangabe) sind zu bevorzugen, wenn das Datenformat der zu formatierenden Zahl ganzzahlig ist. Es findet dann intern keine Datenkonvertierung statt, es können also auch 8 Byte-Zahlen ohne Präzisionsverlust formatiert werden. Die Anzahl der ausgegebenen Stellen wird automatisch aus dem Zahlenformat bestimmt: bei "x" das 2-fache der Byte-Anzahl, bei "b" das 8-fache. Gegebenenfalls wird mit führenden Nullen aufgefüllt.

Beispiele:

```
txWert= TForm(1.2345, "f2,3")
; txWert enthält: " 1,234"

txWert= TForm(31, "f4.2")
; txWert enthält: " 31.00" (2 führende Leerzeichen)

txWert= TForm(31, "g4.0")
; txWert enthält: "0031"

txWert= TForm(31.000, "")
; txWert enthält: "31"

txWert= TForm(1.200, "a.6")
; txWert enthält: "1.2"

txWert= TForm(1.201, "a.6")
; txWert enthält: "1.201"

txWert= TForm(31, "x2")
; txWert enthält: "1F"

value = 128
SetDataFormat(value, 6) ; 2 Byte ganzahlig ohne Vorzeichen
txWert= TForm(value, "x")
; txWert enthält: "0080"
txWert= TForm(value, "b")
; txWert enthält: ""0000000010000000""

value = -2
SetDataFormat(value, 2) ; 1 Byte ganzahlig mit Vorzeichen
txWert= TForm(value, "x")
; txWert enthält: "FE"
txWert= TForm(value, "b")
; txWert enthält: "11111110"
```

Siehe auch:

TConv, TPart, TtoSv

ThrowError

Fehler erzeugen

Deklaration:

```
ThrowError ( TxErrorText )
```

Parameter:

TxErrorText	Auszugebender Fehlertext

Beschreibung:

Die Funktion erzeugt einen anwenderdefinierten Laufzeit-Fehler mit dem angegebenen Fehlertext. Die Anzeige und weitere Behandlung erfolgt in der selben Weise wie bei Fehlern, die zur Laufzeit bei durch den Formelinterpreter gemeldet werden - und ist somit vom aktuell eingestellten Fehlerbehandlungs-Modus abhängig, festgelegt durch OnError().

Wenn die Fehlerbehandlung auf "Standard" gesetzt ist, wird eine Meldungsbox mit dem vorgegebenen Text angezeigt und die Ausführung abgebrochen. Bei den Fehlermodi "Return" und "ResumeEnd" wird die Ausführung entsprechend fortgesetzt, der Fehlertext kann dann mittels GetLastError() später abgefragt werden. "ReturnFail" liefert den Fehler an den Aufrufer zurück. "ResumeNext" ist in Kombination mit ThrowError() im allgemeinen nicht sinnvoll.

Beispiele:

Die in FAMOS eingebaute Funktion zur Wurzel-Berechnung Sqrt() liefert bei negativen Eingangswerten eine 0 und gibt eine entsprechende Warnung aus. Die folgende Sequenzfunktion gibt stattdessen einen Fehler zurück.

```
; Deklaration: !Sqrt_Strict(Par) => Result
OnError("ReturnFail")
IF min(Par) < 0
   ThrowError("Der Parameter enthält negative Werte!")
END
Result = Sqrt(par)
```

Die nachfolgende Sequenzfunktion prüft, ob ein Datensatz komplett oberhalb eines gegebenen Referenzdatensatzes verläuft. Wenn die Funktion wegen unpassender Parameter nicht ausgeführt werden kann, wird dies durch einen speziellen Rückgabewert signalisiert. Der Aufrufer kann dann die Fehlerursache mittels GetLastError() abfragen.

```
; Deklaration: !CheckAbove(TestData, Lower) => Result
; Result = 1:  OK
; Result = 0:  Mindestens 1 Wert von [TestData] ist kleiner als der korrespondierende Wert in [Lower]
; Result = -1: Fehler: Eingangsdaten passen nicht zusammen (bzgl. ihrer x-Achse) oder haben Events/Segmente

OnError("Return", "Unbekannter Fehler", "Result", -1)
IF NOT(VerifyVar(TestData, "!SegEvn")) OR NOT(VerifyVar(Lower, "!SegEvn"))
   ThrowError("Die Parameter dürfen weder Segmente noch Events besitzen!")
END
Verify(Leng?(TestData)=Leng?(Lower) AND xdel?(TestData)=xdel?(Lower) AND xoff?(TestData)=xoff?(Lower), "Parameter: Inkompatible x-Skalierung!")
Result = Max(Lower - TestData) < 0
```

Siehe auch:

OnError, Verify, VerifyVar, BoxMessage, GetLastError

Time?

Die Triggerzeit eines Datensatzes wird abgefragt.
Alternativer Name: **Zeit?**

Deklaration:

```
Time? ( Daten ) -> EwZeit
```

Parameter:

Daten	Datensatz, dessen Triggerzeit ermittelt werden soll.
EwZeit	Triggerzeit

Beschreibung:

Die Funktion liefert die Triggerzeit des Datensatzes in einem speziellen imc FAMOS-Zeitformat, in dem Datum und Uhrzeit enthalten sind.

Der erhaltene Zahlenwert kann mit den weiteren Zeitfunktionen zerlegt, verrechnet und zugewiesen werden.

Beispiele:

Die Triggerzeit eines Datensatzes wird abgefragt und in einer Ausgabebox angezeigt:

```
triggerTime = Time?(data)
Tx = TimeToText(triggerTime, 0)
BoxMessage("Zeit des Datensatzes: ", Tx, "!1")
```

Siehe auch:

SetTime, TimeSplit, TimeJoin, TimeToText, TimeSystem?

TimeAdd

Sekunden zu Zeitwert addieren
Alternativer Name: **ZeitAdd**

Deklaration:

```
TimeAdd ( EwZeit, EwSekunden ) -> EwZeitSumme
```

Parameter:

EwZeit	Zeitangabe im internen Zeitformat.
EwSekunden	Zu addierende Zahl von Sekunden.
EwZeitSumme	Resultierender Zeitwert.

Beschreibung:

Zu einer im imc FAMOS-Zeitformat vorliegenden Zeitangabe wird eine Anzahl von Sekunden addiert.

Das Ergebnis liegt wieder im internen Zeitformat vor.

Beispiele:

Die Triggerzeit eines Datensatzes wird um 1 Minute erhöht:

```
triggerTime = Time?(data)
triggerTime = TimeAdd(triggerTime, 60)
SetTime(data, triggerTime)
```

Siehe auch:

SetTime, TimeDiff, TimeJoin, TimeSplit, TimeToText

TIMECOPY

Triggerzeit kopieren
Alternativer Name: **ZEITKOPIE**

Der Befehl ist veraltet, statt dessen sollten die leistungsfähigeren Funktionen SetTime() und Time?() verwendet werden.

Deklaration:

```
TIMECOPY Quellvariable Zielvariable
```

Parameter:

Quellvariable	Variable, deren Triggerzeit kopiert werden soll.
Zielvariable	Variable, die die neue Triggerzeit erhalten soll.

Beschreibung:

Die Triggerzeit der Zielvariablen wird gleich der Triggerzeit der Quellvariablen gesetzt.

Beispiele:

```
TIMECOPY qdat zdat
```

Die Variable "zdat" erhält die Triggerzeit der Variablen "qdat".

Siehe auch:

Time?, SetTime

TimeDiff

Differenz aus zwei Zeitwerten bilden
Alternativer Name: **ZeitDiff**

Deklaration:

```
TimeDiff ( EwZeit1, EwZeit2 ) -> EwDiffInSekunden
```

Parameter:

EwZeit1	Erste Zeitangabe im internen Zeitformat.
EwZeit2	Zweite Zeitangabe im internen Zeitformat.
EwDiffInSekunden	Resultierende Differenz in Sekunden.

Beschreibung:

Aus zwei im imc FAMOS-Zeitformat vorliegenden Zeitangaben wird die Differenz in Sekunden gebildet.

Beispiele:

Der X-Offset eines Datensatzes wird auf die Differenz der Triggerzeiten von zwei anderen Datensätzen gesetzt:

```
time1 = Time?(dataA)
time2 = Time?(dataB)
sec = TimeDiff(time1, time2)
XOFFSET dataC sec
```

Siehe auch:

SetTime, TimeAdd, TimeJoin, TimeSplit, TimeToText

TimeJoin

Ein Zeitwert im internen Zeitformat wird aus den Komponenten erzeugt.
Alternativer Name: **ZeitBinde**

Deklaration:

```
TimeJoin ( EwTag, EwMonat, EwJahr, EwStunde, EwMinute, EwSekunden ) -> EwZeit
```

Parameter:

EwTag	Tag (1..31)
EwMonat	Monat (1..12)
EwJahr	Jahr (1980..)
EwStunde	Stunde (0..23)
EwMinute	Minute (0..59)
EwSekunden	Sekunden (0..<60)
EwZeit	Resultierender Zeitwert

Beschreibung:

Ein Zeitwert im imc FAMOS- Zeitformat wird aus den einzelnen Bestandteilen zusammengesetzt.

Beispiele:

Die Triggerzeit des Datensatzes wird auf den 2.11.1995, 12:00 Uhr gesetzt:

```
SetTime(data, TimeJoin(2, 11, 1995, 12, 0, 0))
```

Siehe auch:

SetTime, Time?, TimeSplit, TimeToText, TimeSystem?

TIMESET

Triggerzeit setzen
Der Befehl ist veraltet, statt dessen sollte die leistungsfähigere Funktion SetTime() verwendet werden.

Deklaration:

```
TIMESET Zielvariable EwTag EwMonat EwJahr EwStunde EwMinute EwSekunde
```

Parameter:

Zielvariable	Variable, deren Triggerzeit gesetzt werden soll.
EwTag	Tag
EwMonat	Monat
EwJahr	Jahr (4-stellig)
EwStunde	Stunde
EwMinute	Minute
EwSekunde	Sekunde

Beschreibung:

Die Triggerzeit der Zielvariablen wird entsprechend der neu angegebenen Zeit gesetzt.

Beispiele:

```
   TIMESET ZielVar 1 3 1992 12 0 1
```

Die Triggerzeit der Variablen "ZielVar" wird auf '01.03.1992 12:00:01' gesetzt.

Siehe auch:

Time?, SetTime

TimeSplit

Abfrage einer Zeitkomponente aus einer Zeitangabe im imc FAMOS-Zeitformat.
Alternativer Name: **ZeitTeil**

Deklaration:

```
TimeSplit ( EwZeit, Auswahl ) -> EwZeitKomponente
```

Parameter:

EwZeit	Zeitangabe im imc FAMOS-Zeitformat.
Auswahl	Auswahl der Komponente
	0 : Tag (1..31)
	1 : Monat (1..12)
	2 : Jahr (1980..)
	3 : Stunde (0..23)
	4 : Minute (0..59)
	5 : Sekunde (0.. <60)
EwZeitKomponente	Gewünschte Komponente

Beschreibung:

Aus einer im imc FAMOS- Zeitformat vorliegenden Zeitangabe wird eine Komponente abgefragt.

Beispiele:

Das Jahr, im dem der Datensatz erzeugt wurde, wird bestimmt:

```
year = TimeSplit(Time?(data), 2)
```

Siehe auch:

SetTime, Time?, TimeToText, TimeJoin

TimeSystem?

Abfrage der aktuellen Systemzeit.
Alternativer Name: **ZeitSystem?**

Deklaration:

```
TimeSystem? ( ) -> EwZeit
```

Parameter:

EwZeit	Systemzeit im imc FAMOS-Zeitformat.

Beschreibung:

Die Funktion liefert die aktuelle Systemzeit im imc FAMOS-Zeitformat. Der erhaltene Zahlenwert kann mit den anderen Zeitfunktionen weiterverarbeitet werden.

Beispiele:

Die aktuelle Systemzeit wird in einen Text konvertiert und ausgegeben:

```
tx = TimeToText(TimeSystem?(), 0)
BoxMessage("Current time", tx, "!1")
```

Siehe auch:

SetTime, Time?, TimeToText, TimeAdd, TimeJoin, TimeSplit

TimeToText

Ein Zeitwert wird in einen Text konvertiert.
Alternativer Name: **ZeitInText**

Deklaration:

```
TimeToText ( EwZeit, EwCode ) -> TxZeit
```

Parameter:

EwZeit	Zeitwert im internen Zeitformat.
EwCode	Formatierungsvorschrift
	0 : Datum und Zeit, durch Leerzeichen getrennt, Jahr 2-stellig
	1 : Nur Datum, Jahr 2-stellig
	2 : Nur Uhrzeit
	3 : Datum und Zeit, durch Leerzeichen getrennt, Jahr 4-stellig
	4 : Nur Datum, Jahr 4-stellig
TxZeit	Zeitwert als Text.

Beschreibung:

Die Funktion wandelt eine als Zahl vorliegende Zeitangabe im imc FAMOS-Zeitformat in einen Text. Die Windows-Systemeinstellungen bezüglich Zeit- und Datumsformat werden beachtet.

Beispiele:

Die aktuelle Systemzeit wird in einen Text konvertiert und ausgegeben:

```
tx = TimeToText(TimeSystem?(), 0)
tx = TimeToText(TimeSystem?(), 0)
BoxMessage("Current time", tx, "!1")
```

Siehe auch:

SetTime, Time?, TimeSystem?, TimeAdd, TimeJoin, TimeSplit

TLeng

Bestimmt die Länge eines Textes, d.h. die Anzahl seiner Zeichen. Bei einem Textfeld wird die Länge jedes Elements bestimmt.
Alternativer Name: **TLang**

Deklaration:

```
TLeng ( TxText ) -> Länge
```

Parameter:

TxText	Text/Textfeld, dessen Länge zu ermitteln ist
Länge	Einzelwert (bei Text) oder Datensatz (bei Textfeldern) mit der jeweiligen Anzahl der Zeichen der Texte. -1 bei anderen Datentypen.

Beschreibung:

Die Anzahl der Zeichen in einem Text (einschließlich Leerzeichen) wird ermittelt und zurückgegeben.

Bei einem Textfeld wird die Länge der einzelnen Elemente bestimmt und als Datensatz zurückgegeben.

Die Länge eines leeren Textes ist 0.

Wenn der Datentyp des Parameters weder Text noch Textfeld ist, wird eine -1 zurückgegeben.

Beispiele:

```
text = "15 Zeichen lang"
Länge = TLeng(text)
```

Länge erhält den Wert 15.

```
txa = ["current1", "RPM2", "", "rpm1", "Current2"]
Längen = TLeng(txa)
```

Längen enthält die Werte [8, 4, 0, 4, 8].

Siehe auch:

TAdd, TForm, TPart, TComp, TConv

TLike

Ein Text oder Textfeld wird mit einem Muster auf Übereinstimmung verglichen. Das Muster kann Jokerzeichen enthalten.

Deklaration:

```
TLike ( TxText, TxMuster, EwOption ) -> Ergebnis
```

Parameter:

TxText	Zu prüfender Text oder Textfeld.
TxMuster	Zu vergleichendes Muster. Die Jokerzeichen '?' und '*' sind erlaubt.
EwOption	Steuert, ob der Vergleich Groß-/Kleinschreibung berücksichtigen soll.
	0 : Groß- und Kleinschreibung eines Buchstabens werden als identisch betrachtet.
	1 : Groß- und Kleinschreibung eines Buchstabens werden als verschieden betrachtet.
Ergebnis	Ergebnis des Vergleichs, Einzelwert oder Datensatz mit den folgenden Werten:
	0 : Der Text entspricht nicht dem gegebenen Muster.
	1 : Der Text entspricht dem gegebenen Muster.

Beschreibung:

Im Muster (2. Parameter) können die Jokerzeichen "?" und "*" verwendet werden. "?" steht dabei für genau ein beliebiges Zeichen, "*" für beliebig viele beliebige Zeichen.

Wenn der erste Parameter ein Textfeld ist, wird jedes Element des Feldes mit dem Muster verglichen. Das Ergebnis ist ein Datensatz, der nur die Werte [0, 1] enthält und die Länge des Textfeldes hat.

Es werden nur die ersten 259 Zeichen beider Texte berücksichtigt.

Für komplexere Mustervergleiche kann die Funktion TxRegexMatch() verwendet werden.

Beispiele:

Prüfe, ob TxVar mit dem Buchstaben 'a' oder 'A' beginnt:

```
ok = TLike(TxVar, "a*", 0)
```

Prüfe, ob TxVar die Zeichenfolge "voltage" an beliebiger Position enthält. Bezüglich Groß-/Kleinschreibung wird nicht unterschieden:

```
ok = TLike(TxVar, "*voltage*", 0)
```

Prüfe, ob TxVar mit der Zeichenfolge "A0" beginnt und genau 2 weitere Zeichen enthält. Groß-/Kleinschreibung wird unterschieden:

```
ok = TLike(TxVar, "A0??", 1)
```

Prüfe, ob TxVar mit der Zeichenfolge "Channel" beginnt und mit einer "_1" endet. Groß-/Kleinschreibung wird nicht unterschieden:

```
ok = TLike(TxVar, "Channel*_1", 1)
```

```
txa = [ "current1", "RPM2", "rpm1", "Current2", "voltage", "rpm11"]
; Prüfe ob die Feldelemente auf 1 enden:
res = TLike(txa, 0, "*1")
```

res enthält [1, 0, 1, 0, 0, 1]

```
; Prüfe ob die Feldelemente mit einem kleinen 'r' beginnen und genau 4 Zeichen haben:
res = TLike(txa, 1, "r???")
```

res enthält [0, 0, 1, 0, 0, 0]

Siehe auch:

TComp, TxRegExMatch

ToInt

Konvertierung in das Datenformat 'Integer 8 Byte (mit Vorzeichen)'

Deklaration:

```
ToInt ( Parameter ) -> Konvertiert
```

Parameter:

Parameter	Zu konvertierender Einzelwert oder Datensatz
Konvertiert	Konvertierter Einzelwert oder Datensatz

Beschreibung:

Der als Parameter übergebene Einzelwert oder Datensatz wird in das Datenformat "Integer (8 Byte mit Vorzeichen)" konvertiert. Dabei wird jeder Wert auf die nächstliegende ganze Zahl gerundet.

Anmerkungen:

- Der maximale Wertebereich des resultierenden Datenformates ist [-2^63-1 ... 2^63], bei Unter-/Überschreitung wird der Wert auf das jeweilige Bereichsende gesetzt.
- Alle anderen Eigenschaften des Parameters werden übernommen.
- Der Parameter darf segmentiert und eventiert sein.

Der Aufruf

```
integer = ToInt(parameter)
```

ist wirkungsgleich zu

```
integer = parameter
SetDataFormat(integer, 12)
```

Beispiele:

Die Python-Funktion "round" wird verwendet, um PI auf 3 Nachkommastellen zu runden. Ohne den Aufruf der ToInt()-Funktion würde die Konstante als reelle Zahl an Python übergeben werden und einen Laufzeitfehler verursachen, da die Funktion an dieser Stelle ein ganzzahliges Argument erwartet.

```
PI_rounded = PyCallFunction("", "round", PI, ToInt(3))
```

Siehe auch:

SetDataFormat

Unterstützt ab:

Version 2022

Top

Liefert die x-Position aller Punkte, deren y-Werte größer als eine Schwelle sind.
Alternativer Name: **Oben**

Deklaration:

```
Top ( Daten, EwSchwelle ) -> XOben
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND],[XY].
EwSchwelle	Schwelle
XOben	Die ermittelten X-Koordinaten der Werte oberhalb [EwSchwelle].

Beschreibung:

Die x-Koordinaten aller Punkte des Eingangsdatensatzes, die größer als die gegebene Schwelle sind, bilden das Ergebnis.

- Mit der Funktion Value() können Sie die zugehörigen y-Koordinaten ermitteln.
- Wenn kein Wert die Bedingung erfüllt, wird ein leerer Datensatz geliefert.
- Alternativ können Sie die Funktion SearchLevel() verwenden, die ebenfalls für XY-Datensätze geeignet ist.

Beispiele:

```
NDxPeak = Top(NDdata, "5V")
```

Das Ergebnis sind die x-Koordinaten aller Punkte des Datensatzes NDdata, deren Wert größer als 5 Volt ist.

```
NDyPeak = Value(NDdata, Top(NDdata, topValue))
```

Das Ergebnis sind alle y-Koordinaten des Datensatzes NDdata, die größer als topValue sind.

```
meanValue = Mean(Value(NDimpulse, Top(NDimpulse, 0.1)))
```

Gegeben ist ein Datensatz NDimpulse, der eine Anzahl von Impulsen enthält, die auf einem Rauschen von maximal 0.1 liegen. Das Ergebnis ist der Mittelwert aller Impulse.

Siehe auch:

SearchLevel, Value, All0, RangeSet, xMax

TPart

Aus einem Text wird ein Teil herauskopiert.
Alternativer Name: **TTeil**

Deklaration:

```
TPart ( TxText, EwAnfang, EwLänge ) -> TxTeil
```

Parameter:

TxText	Text, aus dem ein Teil herauskopiert werden soll
EwAnfang	Position, ab der herauszukopieren ist
EwLänge	Anzahl der zu kopierenden Zeichen
TxTeil	Der herauskopierte Text

Beschreibung:

Aus dem gegebenen Text wird ein Teil herauskopiert, der durch die Position des ersten zu kopierenden Zeichens und die Anzahl der zu kopierenden Zeichen definiert wird.

- Die Position des ersten Zeichens im Text und damit der kleinstmögliche Wert für [Anfang] ist 1.
- Ist [Anfang] größer als die Textlänge, wird ein leerer Text zurückgegeben.
- Ist [Anfang] + [Länge] größer als die Textlänge, wird der zu kopierende Teil am Textende beendet.

Beispiele:

```
Kennung = TPart("daten.dat", 7, 3)
```

Kennung erhält den Inhalt "dat".

Siehe auch:

TReplace, TAdd, TForm, TLang, TtoSv

TransposeMatrix

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Eine Matrix (ein Datensatz mit Segmenten) wird transponiert. Datei werden Zeilen und Spalten vertauscht. Die Zeilen der Matrix entsprechen den Segmente des Datensatzes. Eingangsdaten und Ergebnis sind segmentiert. Die Eingangsdaten sind einkomponentig.
Diese Funktion ist nur zur Kompatibilität mit imc FAMOS 6.0 und Vorgängern enthalten. Bitte benutzen Sie stattdessen MatrixTranspose().

Deklaration:

```
TransposeMatrix ( Matrix ) -> Ergebnis
```

Parameter:

Matrix	Eingangsmatrix, zu transponieren.
Ergebnis	Transponierte Matrix

Beispiele:

```
Spektren = AmpSpectrumRMS ( Kanal, 1000, 0, 50, 1, 0, 0 )
tm = TransposeMatrix ( Spektren )
```

Die Matrix besteht aus Amplitudenwerten, aufgetragen über Zeit und Frequenz. Mit der Funktion werden Zeit- und Frequenz-Achse vertauscht.

Siehe auch:

MatrixTranspose

TransRec

Datenreduktion nach dem Transitional-Recording-Verfahren

Deklaration:

```
TransRec ( Daten, EwToleranz ) -> Reduziert
```

Parameter:

Daten	Zu reduzierender Datensatz. [ND]
EwToleranz	Zulässige Toleranz (in phys. Einheiten)
Reduziert	Reduzierter Datensatz [XY]

Beschreibung:

Ein äquidistant abgetasteter Datensatz wird in einen XY-Datensatz interpoliert. Die Toleranz (in physikalischen Einheiten) gibt die erlaubte Abweichung zwischen Quelldaten und dem interpolierten Resultat an. Je größer die Toleranz, um so kürzer ist der interpolierte Datensatz.

Bei analogen Quelldaten sollte das Ergebnis mit 'Linien' im Kurvenfenster dargestellt werden (keine 'Treppen').

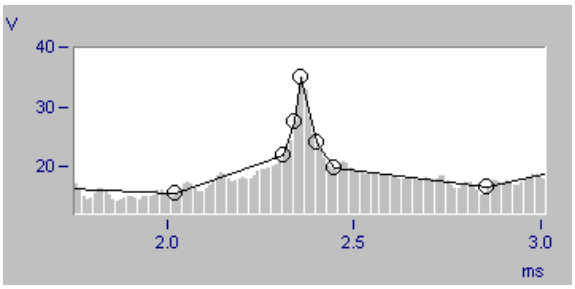
Bei digitalen Quelldaten sollte [EwToleranz] = 0 benutzt werden. Dann werden nur Änderungen des Signalzustandes vermerkt. Für pseudo-analoge Datenreduktion kann [EwToleranz] = 1e-20 benutzt werden.

Die Funktion ändert nicht das Datenformat der Quelldaten (float, double, integer..). Für eine gute Datenreduktion sollte deshalb eine Funktion wie SetDataFormat() verwendet werden, um das Datenformat der Y-Komponente auf 1- oder 2-Byte Integer zu setzen.

Beispiele:

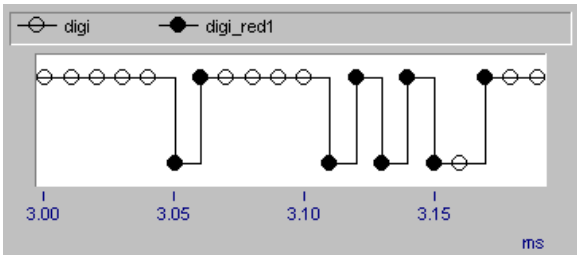
```
anstieg_2 = TransRec(anstieg, 2)
```

Datenreduktion für analoge Daten. Die grauen Balken zeigen das Originalsignal, die schwarzen Geraden und Kreise den Verlauf und die Punkte des reduzierten Signals.




```
digi_red1= TransRec(digi, 0)
```

Datenreduktion für digitale Daten. Die schwarzen Kreise zeigen die Punkte des Originalsignals, die schwarzen Punkte das resultierende reduzierte Signal.



Siehe auch:

SetDataFormat, Red, XYdt

TReplace

In einem Text wird eine ausgewählte Zeichenfolge durch eine andere ersetzt.
Alternativer Name: **TErsetze**

Deklaration:

```
TReplace ( TxGesamt, TxTeil, TxErsatz ) -> TxErgebnis
```

Parameter:

TxGesamt	Der gesamte Text, in dem ein Teil durch einen anderen Text ersetzt werden soll.
TxTeil	Der Text-Ausschnitt, der ersetzt werden soll.
TxErsatz	Der Text, der in [TxGesamt] anstelle von [TxTeil] eingefügt werden soll.
TxErgebnis	Entsprechend veränderter Text

Beschreibung:

In einem Text wird nach einem Teilstück gesucht. Wenn das Teilstück gefunden wird, wird es durch den angegebenen Ersatz ersetzt.

- Wird [TxTeil] in [TxGesamt] nicht gefunden, wird [TxGesamt] unverändert zurückgegeben.
- Auch wenn der zu suchende Text mehrmals auftaucht, wird nur das zuerst gefundene Stück ersetzt.
- Zwischen Groß- und Kleinschreibung wird nicht unterschieden, d.h. "X" und "x" sind gleichbedeutend bei der Suche nach dem Teiltext.

Beispiele:

```
TxFile = "messung.txt"
BoxMessage("Hinweis", TReplace("Datei %s gespeichert!", "%s", TxFile), "!1")
```

Die Meldung "Datei messung.txt gespeichert!" wird angezeigt.

Siehe auch:

TxWhere, TAdd, TPart, TComp, TConv

TsaAppend

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird ein neues Element hinten angehängt.

Deklaration:

```
TsaAppend ( TsaChannel, Data, Time )
```

Parameter:

TsaChannel	TsaChannel
Data	Data
Time	Time

Beschreibung:

Inhalt und Zeit des neuen Elementes werden vorgegeben. Der Inhalt wird als Datensatz vorgegeben. Der Datensatz wird Byte für Byte übernommen, egal in welchem Zahlenformat er vorliegt. Sinnvollerweise sollte der Datensatz das Zahlenformat Byte haben. Die maximale Größe eines Elementes (siehe Zeitstempel-Ascii-Doku) darf nicht überschritten werden. Die Zeit darf nicht kleiner als die des Vorgängers sein. Zeiten müssen monoton wachsen, nicht aber streng monoton. D.h. die Zeiten benachbarter Elemente dürfen gleich sein. Die Zeit wird ggf. gerundet. Denn die Zeit wird intern als x0+i*dx dargestellt, wobei i >= 0 eine ganze Zahl ist. Die Zeit wird als relative Zeit vorgegeben, also als Zeit ab dem Trigger, also ohne die absolute Triggerzeit, also ohne Datum und Uhrzeit des Starts der Messung. Der übergebene Kanal wird dabei verändert. Hat der Datensatz Events, dann wird das Element an das letzte Event angehängt.

Beispiele:

Ein Element wird angehängt, das 10 Werte von 48 bis 57 enthält.

```
Content = rampe ( 48, 1, 10 )
SetDatFormat ( Content, 5, 0, 255) ; Das Zahlenformat Byte erzwingen (0..255).
Time = 20 ;

TsaAppend ( TsaChannel, Content, Time )
```

TsaAppendText

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird ein neues Element hinten angehängt.

Deklaration:

```
TsaAppendText ( TsaChannel, Text, Time )
```

Parameter:

TsaChannel	TsaChannel
Text	Text
Time	Time

Beschreibung:

Inhalt und Zeit des neuen Elementes werden vorgegeben. Der Inhalt wird als Text vorgegeben. Die Zeit darf nicht kleiner als die des Vorgängers sein. Zeiten müssen monoton wachsen, nicht aber streng monoton. D.h. die Zeiten benachbarter Elemente dürfen gleich sein. Die Zeit wird ggf. gerundet. Denn die Zeit wird intern als x0+i*dx dargestellt, wobei i >= 0 eine ganze Zahl ist. Die Zeit wird als relative Zeit vorgegeben, also als Zeit ab dem Trigger, also ohne die absolute Triggerzeit, also ohne Datum und Uhrzeit des Starts der Messung. Der übergebene Kanal wird dabei verändert. Hat der Datensatz Events, dann wird das Element an das letzte Event angehängt.

Beispiele:

Ein Element mit festem Text wird angehängt.

```
Time = 20
Text = "hello"

TsaAppendText ( TsaChannel, Text, Time )
```

TsaCreateEmpty

Anwendungsbereich: Zeitstempel-ASCII-Daten

Ein leerer Kanal im Format Zeitstempel-Ascii wird erzeugt.

Deklaration:

```
TsaCreateEmpty ( x0, dx, xUnit ) -> TsaChannel
```

Parameter:

x0	x0
dx	dx
xUnit	xUnit
TsaChannel	TsaChannel

Beschreibung:

x0: kleinster möglicher Zeitstempel dx: Auflösung der Zeitstempel Alle möglichen Zeiten, die Elemente in diesem Kanal später haben können, sind x = x0+i*dx, i>=0, ganze Zahl Die absolute Zeit (Triggerzeit) wird auf 0.0 gesetzt, also 1.1.1980.

Beispiele:

Ein leerer Kanal wird erzeugt. Die Zeiten, die später eingetragen werden, sind >= 0 und haben eine Auflösung von 1ms.

```
TsaChannel = TsaCreateEmpty ( 0.0, 1e-3, "s" )
```

TsaDataToText

Anwendungsbereich: Zeitstempel-ASCII-Daten

Ein Datensatz Data wird Byte für Byte in einen Text konvertiert.

Deklaration:

```
TsaDataToText ( Data, ZeroReplace ) -> Text
```

Parameter:

Data	Data
ZeroReplace	ZeroReplace
Text	Text

Beschreibung:

Dabei ist das Zahlenformat des Daten egal. Sinnvollerweise sollte der Datensatz das Zahlenformat Byte haben. Alle Bytes mit dem Wert null werden durch ZeroReplace ersetzt. ZeroReplace: Ersatz für Null-Bytes, 0..255. Hat es den Wert 0, wird der Text bei einem 0-Byte beendet.

Beispiele:

Ein Datensatz wird in einen Text konvertiert. Damit der Text nicht abgeschnitten wird, werden alle Null-Bytes durch ein Leerzeichen (Ascii 32) ersetzt.

```
Text = TsaDataToText ( Data, 32 )
```

TsaDecode

Verfügbar ab: Professional Edition

Ein Zeitstempel-Ascii-Kanal wird als Folge von zeitgestempelten Botschaften, z.B. vom CAN-Bus aufgefasst. Aus den Botschaften wird ein Kanal gebildet, indem aus jeder Botschaft an einer bestimmten Position ein Wert ermittelt wird.

Deklaration:

```
TsaDecode ( TsaChannel, Analysis [, Startbyte] [, Startbit] [, BitCount] [, Datenformat] [, ByteOrder] [, Factor] [, Offset] [, Unit] ) -> Channel
```

Parameter:

TsaChannel	TsaChannel
Analysis	Was ist zu analysieren?
	"" : Allgemeine Botschaft in beliebigem Format mit beliebiger Herkunft. Bei CAN-Botschaften gilt: Die ID steht in den ersten 4 Bytes, gefolgt von den Datenbytes.
	"CAN.data" : Daten einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Das Startbyte zählt ab dem Beginn des Datenbereichs der CAN-Botschaft.
	"CAN.ID" : Identifier einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Nur die ID ist gewünscht. Das einen Extended Identifier kennzeichnende MSB (Bit 31) = 1 wird nicht gelesen. Damit ist keine Unterscheidung von Standard und Extended Identifiern mehr möglich.
	"CAN.ID.msb" : Identifier einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Nur die ID ist gewünscht. Das einen Extended Identifier kennzeichnende MSB (Bit 31) = 1 kann enthalten sein.
	"CAN.len" : Länge einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Nur die Anzahl der Datenbytes (ohne ID) ist gewünscht.
	"len" : Länge der Botschaft. Die gesamte Länge der Botschaft wird ermittelt. Bei CAN ist zu beachten, dass die Länge inkl. der 4 Bytes für die ID ermittelt wird.
Startbyte	Ab dieser Byte-Position in der Botschaft wird der Wert gelesen. Startbyte = 0 für das erste Byte. Das Byte zählt ab dem Beginn der Botschaft. Ggf. vorhandene ID muss berücksichtigt werden. Bei "CAN.data" ist Startbyte = 0 der erste Datenwert der CAN-Botschaft. (optional , Standardwert: 0)
Startbit	Startbit. Ab dieser Bitposition steht innerhalb des Startbytes das erste Bit der zu lesenden Zahl. Das LSB ist Bit 0. Das MSB ist Bit 7. 0 <= Startbit <= 7. Bei Zahlen mit mehr als 56 Bit Länge muss Startbit = 0 (Intel) bzw. = 7 (Motorola) sein. (optional , Standardwert: 0)
BitCount	Bitanzahl. So viele Bits ist die zu lesene Zahl lang. 1 <= Bitanzahl <= 64 (optional , Standardwert: 0)
Datenformat	Datenformat der zu lesenden Zahl (optional , Standardwert: "signed")
	"signed" : Ganze Zahl mit Vorzeichen im Zweierkomplement.
	"unsigned" : Ganze Zahl ohne Vorzeichen
	"real" : reell (float, double). Nur 32 oder 64 Bit mit Startbit = 0 (Intel) bzw. = 7 (Motorola) möglich.
ByteOrder	Byteanordnung. Welches (höchstes, niedrigstes) ist zuerst angegeben. (optional , Standardwert: "intel")
	"intel" : Intel, niedrigstwertiges Byte zuerst, Little-Endian
	"motorola" : Motorola, höchstwertiges Byte zuerst, Big-Endian
Factor	Der gelesene Wert wird mit diesem Skalierungsfaktor multipliziert. Nur für ganzzahlige Formate (optional , Standardwert: 1)
Offset	Zu dem mit dem Skalierungsfaktor multiplizierten Wert wird der Skalierungsoffset addiert. Nur für ganzzahlige Formate (optional , Standardwert: 0)
Unit	Einheit. Der skalierte Wert erhält diese Einheit. (optional , Standardwert: "")
Channel	Kanal im XY-Format.

Beschreibung:

Aus jeder Botschaft wird ein Wert im angegebenen Format extrahiert. Ist die Botschaft ausreichend lang, wird erfolgreich extrahiert. Dieser Wert wird ins Ergebnis geschrieben.

Ist die Botschaft zu kurz, kann der Wert nicht (vollständig) gelesen werden. Er wird dann nicht ins Ergebnis geschrieben.

Der Datensatz darf keine Events haben.

Die Festlegung des Zahlenformats ist kompatibel zu imc DEVICES und imc STUDIO.

Falls es sich um CAN-Botschaften handelt, ist bei der CAN-ID zu beachten: Standard-Identifier sind 11 Bit groß, also Wertebereich 0 bis 2047. Extended Identifer sind 29 Bit groß. Zusätzlich ist das MSB der 32 bit Zahl gesetzt. Die CAN-ID steht in den ersten 4 Byte und ist in INTEL Byte

Reihenfolge angegeben. Das gilt, falls die Daten von imc DEVICES, imc STUDIO erzeugt wurden.

Die Funktion filtert eigentlich nicht. Die Absicht ist zu jeder Botschaft einen Wert ins Ergebnis zu schreiben. Wenn das Bestimmen dieses Wertes aber nicht erfolgreich ist, wird nicht ins Ergebnis geschrieben. Diese Situation wird i. Allg. vermieden, indem zuvor passend gefiltert wurde. Zum Filter siehe TsaFilter().

Skalierung

Skalierungsfaktor und Skalierungsoffset gehorchen dieser Formel:

[Physikalischer Wert] = [Ganze Zahl] * Skalierungsfaktor + Skalierungsoffset

Skalierungsfaktor und Skalierungsoffset gibt es nur bei ganzzahligen Formaten.

Bei Defaultwerten für Skalierungsfaktor und Skalierungsoffset wird bevorzugt ein passendes Datenformat im Ergebnis erzeugt.

Beispiel Intel Byte Reihenfolge

Startbit = 3, BitCount = 15, MSB bei jedem Byte zuerst dargestellt, LSB zuletzt. Bit=X gehört zur Zahl. Bit=0 gehört nicht zur Zahl:

XXXXX000 XXXXXXXX 000000XX

Im 1. Byte sind die 5 obersten Bit gesetzt. Darin befinden sich die niedrigstwertigen Bits der Zahl.

Im 3. Byte sind die untersten 2 Bit gesetzt. Darin befinden sich die höchstwertigen Bits der Zahl.

Startbit = 3 bedeutet, dass die 3 untersten Bits (Bits 0, 1, 2) noch nicht zur Zahl gehören.

Beispiel Intel Byte Reihenfolge

Startbit = 0, BitCount = 16, MSB bei jedem Byte zuerst dargestellt, LSB zuletzt. Bit=X gehört zur Zahl. Bit=0 gehört nicht zur Zahl:

XXXXXXXX XXXXXXXX

Im 1. Byte sind alle 8 Bit gesetzt. Darin befinden sich die niedrigstwertigen Bits der Zahl.

Im 2. Byte sind alle 8 Bit gesetzt. Darin befinden sich die höchstwertigen Bits der Zahl.

Beispiel Motorola Byte Reihenfolge

Startbit = 2, BitCount = 15, MSB bei jedem Byte zuerst dargestellt, LSB zuletzt. Bit=X gehört zur Zahl. Bit=0 gehört nicht zur Zahl:

00000XXX XXXXXXXX XXXX0000

Im 1. Byte sind die 3 untersten Bit gesetzt. Darin befinden sich die höchstwertigen Bits der Zahl.

Im 3. Byte sind die obersten 4 Bit gesetzt. Darin befinden sich die niedrigstwertigen Bits der Zahl.

Startbit = 2 bedeutet, dass die 3 untersten Bits (Bit 0, 1, 2) zur Zahl gehören.

Beispiel Motorola Byte Reihenfolge

Startbit = 7, BitCount = 16, MSB bei jedem Byte zuerst dargestellt, LSB zuletzt. Bit=X gehört zur Zahl. Bit=0 gehört nicht zur Zahl:

XXXXXXXX XXXXXXXX

Im 1. Byte sind alle 8 Bit gesetzt. Darin befinden sich die höchstwertigen Bits der Zahl.

Im 2. Byte sind alle 8 Bit gesetzt. Darin befinden sich die niedrigstwertigen Bits der Zahl.

Beispiele:

xy-Kanal mit allen CAN-IDs erzeugen

```
ids = TsaDecode( tsa, "CAN.ID")
```

Standard und Extended IDs können auftreten. Das MSB zur Unterscheidung soll ebenfalls in den Kanal.

```
ids = TsaDecode( tsa, "CAN.ID.msb")
```

Ab dem 3. Byte einer CAN-Botschaft steht eine 16 bit vorzeichenbehaftete ganze Zahl mit dem niederwertigen Byte zuerst. 1 LSB = 5 N.

```
N = TsaDecode( tsa, "CAN.data", 2, 0, 16, "signed", "intel", 5.0, 0.0, "N")
```

Einen Kanal mit dem 1. Byte der Botschaft erzeugen

```
c = TsaDecode( tsa, "", 0, 0, 8, "unsigned", "intel")
```

Ab dem 3. Byte einer CAN-Botschaft mit Standard Identifier 22 Hex steht eine 16 bit vorzeichenlose ganze Zahl mit dem höherwertigen Byte zuerst. 1 LSB = 5 N.

```
tsaFil = TsaFilter ( tsa, "CAN.ID.std", 0x22, 0x22)
N = TsaDecode( tsaFil, "CAN.data", 2, 7, 16, "unsigned", "motorola", 5.0, 0.0, "N")
```

Zahlenbeispiel

```
tsa = TsaCreateEmpty(1, 0.1, "s")
b01 = [0x31, 0x32, 0x33, 0x81, 0x77, 0x55] ; ext id
setdatFormat( b01, 5, 0, 255 )
TsaAppend (tsa, b01, 1.7 )
c = TsaDecode( tsa, "CAN.ID.msb") ; c.y = 0x81333231
c = TsaDecode( tsa, "CAN.ID") ; c.y = 0x01333231
c = TsaDecode( tsa, "CAN.len") ; c.y = 2
c = TsaDecode( tsa, "len") ; c.y = 6
```

```
c = TsaDecode( tsa, "",0,0,8,"unsigned","intel") ; c.y = 0x31
c = TsaDecode( tsa, "",0,0,16,"unsigned","intel") ; c.y = 0x3231
c = TsaDecode( tsa, "",0,7,16,"unsigned","motorola") ; c.y = 0x3132
c = TsaDecode( tsa, "CAN.data",0,0,16,"unsigned","intel") ; c.y = 0x5577
```

Siehe auch:

TsaFilter, TsaGetFirst, SetDatFormat

TsaDelete

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird das Element an der vorgegebenen Position aus dem Kanal entfernt.

Deklaration:

```
TsaDelete ( TsaChannel, Position )
```

Parameter:

TsaChannel	TsaChannel
Position	Position

Beschreibung:

Damit verändern sich auch die Positionen der folgenden Elemente.

Beispiele:

Das 1. Element wird entfernt

```
Position = TsaFindFirst ( TsaChannel )
TsaDelete ( TsaChannel, Position )
```

TsaFilter

Verfügbar ab: Professional Edition

Ein Zeitstempel-Ascii-Kanal wird als Folge von zeitgestempelten Botschaften z.B. vom CAN-Bus aufgefasst. Durch Filterung wird eine Teilmenge dieser Botschaften erstellt. Z.B. alle Botschaften mit einer bestimmten CAN-ID.

Deklaration:

```
TsaFilter ( TsaChannel, Analysis, Min, Max [, Mask] [, Startbyte] [, Startbit] [, BitCount] [, Datenformat] [, ByteOrder] ) -> Gefiltert
```

Parameter:

TsaChannel	TsaChannel
Analysis	Was ist zu analysieren?
	"" : Allgemeine Botschaft in beliebigem Format mit beliebiger Herkunft. Bei CAN-Botschaften gilt: Die ID steht in den ersten 4 Bytes, gefolgt von den Datenbytes.
	"CAN.data" : Daten einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Das Startbyte zählt ab dem Beginn des Datenbereichs der CAN-Botschaft.
	"CAN.ID" : Identifier einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Nur die ID ist gewünscht. Das einen Extended Identifier kennzeichnende MSB (Bit 31) = 1 wird nicht gelesen. Damit ist keine Unterscheidung von Standard und Extended Identifiern mehr möglich.
	"CAN.ID.msb" : Identifier einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Nur die ID ist gewünscht. Das einen Extended Identifier kennzeichnende MSB (Bit 31) = 1 kann enthalten sein.
	"CAN.ID.std" : Standard Identifier einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Nur eine ID, die ein Standard Identifier ist, ist gewünscht. Extended Identifier werden ignoriert.
	"CAN.ID.ext" : Extended Identifier einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Nur eine ID, die ein Extended Identifier ist, ist gewünscht. Das kennzeichnende MSB (Bit 31) wird nicht extrahiert. Standard Identifier werden ignoriert.
	"CAN.len" : Länge einer CAN-Botschaft. Die vorliegenden Botschaften sind CAN-Botschaften. Nur die Anzahl der Datenbytes (ohne ID) ist gewünscht.
	"len" : Länge der Botschaft. Die gesamte Länge der Botschaft wird ermittelt. Bei CAN ist zu beachten, dass die Länge inkl. der 4 Bytes für die ID ermittelt wird.
Min	Minimalwert. Der gelesene Wert wird nach der Anwendung der Bitmaske auf diesen Wert geprüft. Die Filterbedingung kann nur erfüllt sein, wenn Wert >= Min.
Max	Maximalwert. Der gelesene Wert wird nach der Anwendung der Bitmaske auf diesen Wert geprüft. Die Filterbedingung kann nur erfüllt sein, wenn Wert <= Max.
Mask	Der gelesene Wert wird mit dieser Bitmaske UND-verknüpft. Nur falls Bitmaske <> 0 ist. Nur bis BitCount <= 32. (optional , Standardwert: 0)
Startbyte	Ab dieser Byte-Position in der Botschaft wird der Wert gelesen. Startbyte = 0 für das erste Byte. Das Byte zählt ab dem Beginn der Botschaft. Ggf. vorhandene ID muss berücksichtigt werden. Bei "CAN.data" ist Startbyte = 0 der erste Datenwert der CAN-Botschaft. (optional , Standardwert: 0)
Startbit	Startbit. Ab dieser Bitposition steht innerhalb des Startbytes das erste Bit der zu lesenden Zahl. Das LSB ist Bit 0. Das MSB ist Bit 7. 0 <= Startbit <= 7. Bei Zahlen mit mehr als 56 Bit Länge muss Startbit = 0 (Intel) bzw. = 7 (Motorola) sein. (optional , Standardwert: 0)
BitCount	Bitanzahl. So viele Bits ist die zu lesene Zahl lang. 1 <= Bitanzahl <= 64 (optional , Standardwert: 0)
Datenformat	Datenformat der zu lesenden Zahl (optional , Standardwert: "signed")
	"signed" : Ganze Zahl mit Vorzeichen im Zweierkomplement.
	"unsigned" : Ganze Zahl ohne Vorzeichen
	"real" : reell (float, double). Nur 32 oder 64 Bit mit Startbit = 0 (Intel) bzw. = 7 (Motorola) möglich.
ByteOrder	Byteanordnung. Welches (höchstes, niedrigstes) ist zuerst angegeben. (optional , Standardwert: "intel")
	"intel" : Intel, niedrigstwertiges Byte zuerst, Little-Endian
	"motorola" : Motorola, höchstwertiges Byte zuerst, Big-Endian
Gefiltert	Gefilterter Kanal im Zeitstempel-Ascii Format

Beschreibung:

Aus jeder Botschaft wird ein Wert im angegebenen Format extrahiert. Dieser Wert wird der Filterbedingung unterzogen.

Die Filterbedingung besteht aus einer Überprüfung des Wertebereichs nach vorheriger Maskierung.

Wenn die Filterbedingung erfüllt ist, wird die Botschaft ins Ergebnis geschrieben.

Bei der regulären Filterbedingung muss gelten: Wert >= Min UND Wert <= Max. In diesem Fall ist Min <= Max. Sollen z.B. alle Werte mit 7 herausgefiltert werden, ist Min = 7 und Max = 7 mit Wert >= 7 UND Wert <= 7.

Falls aber Min > Max, dann wird folgendermaßen gedeutet: Wert >= Min ODER Wert <= Max. Sollen z.B. alle Werte <> 7 herausgefiltert werden, ist Min = 8 und Max = 6 mit Wert >= 8 ODER Wert <= 6.

Ist die Botschaft zu kurz, kann der Wert nicht (vollständig) gelesen werden. Die Botschaft wird nicht ins Ergebnis geschrieben.

Der Datensatz darf keine Events haben.

Die Festlegung des Zahlenformats ist kompatibel zu imc DEVICES und imc STUDIO.

Falls es sich um CAN-Botschaften handelt, ist bei der CAN-ID zu beachten: Standard-Identifier sind 11 Bit groß, also Wertebereich 0 bis 2047. Extended Identifer sind 29 Bit groß. Zusätzlich ist das MSB der 32 bit Zahl gesetzt. Die CAN-ID steht in den ersten 4 Byte und ist in INTEL Byte Reihenfolge angegeben. Das gilt, falls die Daten von imc DEVICES, imc STUDIO erzeugt wurden.

Wenn eine Botschaft ins Ergebnis geschrieben wird, wird sie unverändert inklusive Originalzeitstempel übernommen.

Die Botschaften werden auch Frame oder Message bezeichnet.

Beispiel Intel Byte Reihenfolge

Startbit = 3, BitCount = 15, MSB bei jedem Byte zuerst dargestellt, LSB zuletzt. Bit=X gehört zur Zahl. Bit=0 gehört nicht zur Zahl:

XXXXX000 XXXXXXXX 000000XX

Im 1. Byte sind die 5 obersten Bit gesetzt. Darin befinden sich die niedrigstwertigen Bits der Zahl.

Im 3. Byte sind die untersten 2 Bit gesetzt. Darin befinden sich die höchstwertigen Bits der Zahl.

Startbit = 3 bedeutet, dass die 3 untersten Bits (Bits 0, 1, 2) noch nicht zur Zahl gehören.

Beispiel Intel Byte Reihenfolge

Startbit = 0, BitCount = 16, MSB bei jedem Byte zuerst dargestellt, LSB zuletzt. Bit=X gehört zur Zahl. Bit=0 gehört nicht zur Zahl:

XXXXXXXX XXXXXXXX

Im 1. Byte sind alle 8 Bit gesetzt. Darin befinden sich die niedrigstwertigen Bits der Zahl.

Im 2. Byte sind alle 8 Bit gesetzt. Darin befinden sich die höchstwertigen Bits der Zahl.

Beispiel Motorola Byte Reihenfolge

Startbit = 2, BitCount = 15, MSB bei jedem Byte zuerst dargestellt, LSB zuletzt. Bit=X gehört zur Zahl. Bit=0 gehört nicht zur Zahl:

00000XXX XXXXXXXX XXXX0000

Im 1. Byte sind die 3 untersten Bit gesetzt. Darin befinden sich die höchstwertigen Bits der Zahl.

Im 3. Byte sind die obersten 4 Bit gesetzt. Darin befinden sich die niedrigstwertigen Bits der Zahl.

Startbit = 2 bedeutet, dass die 3 untersten Bits (Bit 0, 1, 2) zur Zahl gehören.

Beispiel Motorola Byte Reihenfolge

Startbit = 7, BitCount = 16, MSB bei jedem Byte zuerst dargestellt, LSB zuletzt. Bit=X gehört zur Zahl. Bit=0 gehört nicht zur Zahl:

XXXXXXXX XXXXXXXX

Im 1. Byte sind alle 8 Bit gesetzt. Darin befinden sich die höchstwertigen Bits der Zahl.

Im 2. Byte sind alle 8 Bit gesetzt. Darin befinden sich die niedrigstwertigen Bits der Zahl.

Beispiele:

Ab dem 3. Byte einer CAN-Botschaft mit Standard Identifier 22 Hex steht eine 16 bit vorzeichenlose ganze Zahl mit dem höherwertigen Byte zuerst. 1 LSB = 5 N.

```
tsaFil = TsaFilter ( tsa, "CAN.ID.std", 0x22, 0x22)
N = TsaDecode( tsaFil, "CAN.data", 2, 0, 16, "unsigned", "motorola", 5.0, 0.0, "N")
```

Alle CAN Botschaften herausfiltern mit Standard ID 100

```
tsaFil = TsaFilter ( tsa, "CAN.ID.std", 100, 100)
```

Alle CAN Botschaften herausfiltern mit Extended ID 0x374500 nach J1939, also PF=0x37 und PS=0x45. Die untersten 8 Bit sind die "source address" und sollen ignoriert werden. Die obersten 3 Bit der 29 Bit ID sind die "priority" und sollen ebenfalls ignoriert werden.

```
tsaFil = TsaFilter ( tsa, "CAN.ID.ext", 0x374500, 0x374500, 0x3ffff00 )
```

Alle CAN Botschaften herausfiltern mit Standard ID 33 Hex und außerdem in den ersten Datenbytes eine 7 (Intel, ohne Vorzeichen).

```
tsaFil2 = TsaFilter ( tsa, "CAN.ID.std", 0x33, 0x33)
tsaFil = TsaFilter ( tsaFil2, "CAN.data", 7, 7, 0, 0, 0, 16, "unsigned", "intel")
```

Alle CAN Botschaften herausfiltern, deren Standard ID nicht im Bereich 100 bis 110 liegt.

```
tsaFil = TsaFilter ( tsa, "CAN.ID.std", 111, 99)
```

Zahlenbeispiel. Diese Filter übernehmen die angegebene Botschaft ins Ergebnis.

```
tsa = TsaCreateEmpty(1, 0.1, "s")
b01 = [0x31, 0x32, 0x33, 0x81, 0x77, 0x55] ; ext id
setdatFormat( b01, 5, 0, 255 )
TsaAppend (tsa, b01, 1.7 )
tsaFil = TsaFilter ( tsa, "CAN.ID.ext", 0x01333231, 0x01333231)
tsaFil = TsaFilter ( tsa, "CAN.ID.msb", 0x81333231, 0x81333231)
tsaFil = TsaFilter ( tsa, "CAN.ID", 0x01333231, 0x01333231)
tsaFil = TsaFilter ( tsa, "CAN.ID.ext", 0, 0x1fffffff, 0)
tsaFil = TsaFilter ( tsa, "CAN.ID.msb", 0, 0x9fffffff, 0)
tsaFil = TsaFilter ( tsa, "CAN.ID", 0, 0x1fffffff, 0)
tsaFil = TsaFilter ( tsa, "CAN.len", 2, 2)
tsaFil = TsaFilter ( tsa, "CAN.len", 0, 8)
tsaFil = TsaFilter ( tsa, "len", 6, 6)
tsaFil = TsaFilter ( tsa, "CAN.data", 0x5577, 0x5577, 0, 0, 0, 16, "unsigned", "intel")
tsaFil = TsaFilter ( tsa, "CAN.data", 0x5577, 0x5577, 0xffff, 0, 0, 16, "unsigned", "intel")
tsaFil = TsaFilter ( tsa, "CAN.data", 0x0570, 0x0570, 0x0ff0, 0, 0, 16, "unsigned", "intel")
tsaFil = TsaFilter ( tsa, "CAN.data", 0x7755, 0x7755, 0, 0, 7, 16, "unsigned", "motorola")
tsaFil = TsaFilter ( tsa, "", 0x7755, 0x7755, 0, 4, 7, 16, "unsigned", "motorola")

tsa = TsaCreateEmpty(1, 0.1, "s")
b02 = [0x31, 0x01, 0x0, 0x0, 0x77, 0x55] ; std id
setdatFormat( b02, 5, 0, 255 )
TsaAppend (tsa, b02, 1.7 )
tsaFil = TsaFilter ( tsa, "CAN.ID.std", 0x131, 0x131)
tsaFil = TsaFilter ( tsa, "CAN.ID.msb", 0x131, 0x131)
tsaFil = TsaFilter ( tsa, "CAN.ID", 0x131, 0x131)
```

Siehe auch:

TsaDecode, TsaGetFirst, SetDatFormat

TsaFindBefore

Anwendungsbereich: Zeitstempel-ASCII-Daten

Diese Funktion wird zum Fortsetzen einer Aufzählung der Elemente eines Zeitstempel-Ascii-Kanals aufgerufen.

Deklaration:

```
TsaFindBefore ( TsaChannel, Position ) -> ResultPosition
```

Parameter:

TsaChannel	TsaChannel
Position	Position
ResultPosition	ResultPosition

Beschreibung:

Als Parameter wird die gültige Position eines Elementes vorgegeben. Bestimmt wird die Position des vorherigen Elementes innerhalb des Kanals. Rückgabe: Gültige Position, falls >= 0. Sonst ist kein weiteres Element vorhanden.

Beispiele:

```
; Initialisierung
Position = TsaFindLast ( TsaChannel )
while Position >= 0
   ; hier das Element lesen, z.B.
   text = TsaGetText ( TsaChannel, Position )

   ; hier zum vorherigen Element
   Position = TsaFindBefore ( TsaChannel, Position )
end
```

TsaFindFirst

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bestimmt wird die Position des 1. Elementes innerhalb des Kanals.

Deklaration:

```
TsaFindFirst ( TsaChannel ) -> Position
```

Parameter:

TsaChannel	TsaChannel
Position	Position

Beschreibung:

Diese Funktion wird zum Beginn einer Aufzählung der Elemente eines Zeitstempel-Ascii-Kanals aufgerufen. Rückgabe: Gültige Position, falls >= 0. Sonst ist kein Element vorhanden.

Beispiele:

```
; Initialisierung
Position = TsaFindFirst ( TsaChannel )
solange Position >= 0
   ; hier das Element lesen, z.B.
   text = TsaGetText ( TsaChannel, Position )

   ; hier zum nächsten Element
   Position = TsaFindNext ( TsaChannel, Position )
ende
```

TsaFindLast

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bestimmt wird die Position des letzten Elementes innerhalb des Kanals.

Deklaration:

```
TsaFindLast ( TsaChannel ) -> Position
```

Parameter:

TsaChannel	TsaChannel
Position	Position

Beschreibung:

Diese Funktion wird zum Beginn einer Aufzählung der Elemente eines Zeitstempel-Ascii-Kanals aufgerufen. Rückgabe: Gültige Position, falls >= 0. Sonst ist kein Element vorhanden.

Beispiele:

```
; Initialisierung
Position = TsaFindFirstLast ( TsaChannel )
while Position >= 0
   ; hier das Element lesen, z.B.
   text = TsaGetText ( TsaChannel, Position )

   ; hier zum nächsten Element
   Position = TsaFindBefore ( TsaChannel, Position )
end
```

TsaFindNext

Anwendungsbereich: Zeitstempel-ASCII-Daten

Diese Funktion wird zum Fortsetzen einer Aufzählung der Elemente eines Zeitstempel-Ascii-Kanals aufgerufen.

Deklaration:

```
TsaFindNext ( TsaChannel, Position ) -> ResultPosition
```

Parameter:

TsaChannel	TsaChannel
Position	Position
ResultPosition	ResultPosition

Beschreibung:

Als Parameter wird die gültige Position eines Elementes vorgegeben. Bestimmt wird die Position des nächsten Elementes innerhalb des Kanals. Rückgabe: Gültige Position, falls >= 0. Sonst ist kein weiteres Element vorhanden.

Beispiele:

```
; Initialisierung
Position = TsaFindFirst ( TsaChannel )
solange Position >= 0
   ; hier das Element lesen, z.B.
   text = TsaGetText ( TsaChannel, Position )

   ; hier zum nächsten Element
   Position = TsaFindNext ( TsaChannel, Position )
ende
```

TsaFindTime

Anwendungsbereich: Zeitstempel-ASCII-Daten

Für einen Zeitstempel-Ascii-Kanal wird zu einer vorgegebenen Zeit die Position des ersten Elementes gefunden, dessen Zeit größer gleich der vorgegebenen Zeit ist.

Deklaration:

```
TsaFindTime ( TsaChannel, Time ) -> Position
```

Parameter:

TsaChannel	TsaChannel
Time	Time
Position	Position

Beschreibung:

Die Zeit wird als relative Zeit vorgegeben, also als Zeit ab dem Trigger, also ohne die absolute Triggerzeit, also ohne Datum und Uhrzeit des Starts der Messung. Benachbarte Elemente können durchaus dieselbe Zeit haben. Gültige Position, falls >= 0.

Beispiele:

```
Time = 45.0 ; Element bei >= 45s gesucht.
Position = TsaFindTime ( TsaChannel, Time )
; z.B. den Text abfragen
text = TsaGetText ( TsaChannel, Position )
; die wirkliche Zeit abfragen:
Time = TsaGetTime ( TsaChannel, Position )
```

TsaFindValidPos

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird zu einer beliebigen Position die nächste gleiche oder größere gültige Position eines Elementes bestimmt.

Deklaration:

```
TsaFindValidPos ( TsaChannel, Position ) -> ResultPosition
```

Parameter:

TsaChannel	TsaChannel
Position	Position
ResultPosition	ResultPosition

Beschreibung:

Rückgabe: Gültige Position, falls >= 0. Sonst ist kein weiteres Element vorhanden. Normalerweise sollten die gültigen Positionen mit TsaFindFirst() und TsaFindNext() gefunden werden.

Beispiele:

Es soll geschaut werden, welche Zeit ein Sample bei Position >= 100000 hat.

```
Position = TsaFindValidPos ( TsaChannel, 100000 )
wenn Position >= 0
   Time = TsaGetTime ( TsaChannel, Position )
ende
```

TsaGetCount

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird die Anzahl der enthaltenen Elemente bestimmt.

Deklaration:

```
TsaGetCount ( TsaChannel ) -> Data
```

Parameter:

TsaChannel	TsaChannel
Data	Data

Beschreibung:

Beispiele:

```
Count = TsaGetCount ( TsaChannel )
```

TsaGetData

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird zu einer vorgegebenen Position der Inhalt des betreffenden Elementes als Datensatz zurückgegeben.

Deklaration:

```
TsaGetData ( TsaChannel, Position ) -> Data
```

Parameter:

TsaChannel	TsaChannel
Position	Position
Data	Data

Beschreibung:

Die Position muss gültig sein, z.B. mit TsaFindTime(), TsaFindFirst(), TsaFindNext() oder TsaFindValidPos() bestimmt. Der zurückgegebene Datensatz hat das Zahlenformat Byte.

Beispiele:

```
Position = TsaFindFirst ( TsaChannel )
Content = TsaGetData ( TsaChannel, Position )
```

TsaGetText

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird zu einer vorgegebenen Position der Inhalt des betreffenden Elementes als Text zurückgegeben.

Deklaration:

```
TsaGetText ( TsaChannel, Position ) -> Text
```

Parameter:

TsaChannel	TsaChannel
Position	Position
Text	Text

Beschreibung:

Die Position muss gültig sein, z.B. mit TsaFindTime(), TsaFindFirst(), TsaFindNext() oder TsaFindValidPos() bestimmt. TsaGetData() ist vorzuziehen, wenn das Element binäre Daten enthält.

Beispiele:

```
Position = TsaFindFirst ( TsaChannel )
Text = TsaGetText ( TsaChannel, Position )
```

TsaGetTime

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird zu einer vorgegebenen Position die Zeit des betreffenden Elementes zurückgegeben.

Deklaration:

```
TsaGetTime ( TsaChannel, Position ) -> Time
```

Parameter:

TsaChannel	TsaChannel
Position	Position
Time	Time

Beschreibung:

Die Position muss gültig sein, z.B. mit TsaFindTime(), TsaFindFirst(), TsaFindNext() oder TsaFindValidPos() bestimmt. Die zurückgegebene Zeit ist die relative Zeit ab Start der Messung.

Beispiele:

```
Position = TsaFindFirst ( TsaChannel )
Time = TsaGetTime ( TsaChannel, Position )
```

TsaInsert

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird ein neues Element an der vorgegebenen Position eingefügt.

Deklaration:

```
TsaInsert ( TsaChannel, Position, Data, Time )
```

Parameter:

TsaChannel	TsaChannel
Position	Position
Data	Data
Time	Time

Beschreibung:

Inhalt und Zeit des neuen Elementes werden vorgegeben. Der Inhalt wird als Datensatz vorgegeben. Der Datensatz wird Byte für Byte übernommen, egal in welchem Zahlenformat er vorliegt. Sinnvollerweise sollte der Datensatz das Zahlenformat Byte haben. Die maximale Größe eines Elementes (siehe Zeitstempel-Ascii-Doku) darf nicht überschritten werden. Die Zeit darf nicht kleiner als die des Vorgängers sein und nicht größer als die des Nachfolgers. Zeiten müssen monoton wachsen, nicht aber streng monoton. D.h. die Zeiten benachbarter Elemente dürfen gleich sein. Die Zeit wird ggf. gerundet. Denn die Zeit wird intern als x0+i*dx dargestellt, wobei i >= 0 eine ganze Zahl ist. Die Zeit wird als relative Zeit vorgegeben, also als Zeit ab dem Trigger, also ohne die absolute Triggerzeit, also ohne Datum und Uhrzeit des Starts der Messung. Der übergebene Kanal wird dabei verändert. Die Position muss gültig sein, z.B. mit TsaFindTime(), TsaFindFirst(), TsaFindNext() oder TsaFindValidPos() bestimmt. Aufgrund der Veränderung eines Elementes verändern sich i. Allg. die Positionen der folgenden Elemente. Nach erfolgreichem Ablauf der Funktion ist das Element, das vorher an der Position war, nun nach hinten hinter das neu eingefügte Element verschoben.

Beispiele:

Ganz vorn wird ein Element eingefügt, das 10 Werte von 48 bis 57 enthält.

```
Content = rampe ( 48, 1, 10 )
SetDatFormat ( Content, 5, 0, 255) ; Das Zahlenformat Byte erzwingen (0..255).
Time = 20 ;

Position = TsaFindFirst ( TsaChannel )
TsaInsert ( TsaChannel, Position, Content, Time )
```

TsaInsertText

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird ein neues Element an der vorgegebenen Position eingefügt.

Deklaration:

```
TsaInsertText ( TsaChannel, Position, Text, Time )
```

Parameter:

TsaChannel	TsaChannel
Position	Position
Text	Text
Time	Time

Beschreibung:

Inhalt und Zeit des neuen Elementes werden vorgegeben. Der Inhalt wird als Text vorgegeben. Die Zeit darf nicht kleiner als die des Vorgängers sein und nicht größer als die des Nachfolgers. Zeiten müssen monoton wachsen, nicht aber streng monoton. D.h. die Zeiten benachbarter Elemente dürfen gleich sein. Die Zeit wird ggf. gerundet. Denn die Zeit wird intern als x0+i*dx dargestellt, wobei i >= 0 eine ganze Zahl ist. Die Zeit wird als relative Zeit vorgegeben, also als Zeit ab dem Trigger, also ohne die absolute Triggerzeit, also ohne Datum und Uhrzeit des Starts der Messung. Der übergebene Kanal wird dabei verändert. Die Position muss gültig sein, z.B. mit TsaFindTime(), TsaFindFirst(), TsaFindNext() oder TsaFindValidPos() bestimmt. Aufgrund der Veränderung eines Elementes verändern sich i. Allg. die Positionen der folgenden Elemente. Nach erfolgreichem Ablauf der Funktion ist das Element, das vorher an der Position war, nun nach hinten hinter das neu eingefügte Element verschoben.

Beispiele:

Ganz vorn wird ein Element mit festem Text eingefügt.

```
Text = "hello"
Time = 20 ;

Position = TsaFindFirst ( TsaChannel )
TsaInsertText ( TsaChannel, Position, Text, Time )
```

TsaJoin

Anwendungsbereich: Zeitstempel-ASCII-Daten

An einen Zeitstempel-Ascii-Kanal wird ein zweiter Zeitstempel-Ascii-Kanal angehängt.

Deklaration:

```
TsaJoin ( TsaChannel, TsaChannelJoin, UseTriggerTime, DoMerge, AllowDuplicate )
```

Parameter:

TsaChannel	TsaChannel
TsaChannelJoin	Der anzuhängende Zeitstempel-Ascii-Kanal. Oder 0 im Sonderfall siehe unten.
UseTriggerTime	Triggerzeit beachten
	0 : Triggerzeit wird ignoriert. Nur die relativen Zeitstempel (Zeit ab Start bzw. Trigger) der Elemente werden beachtet.
	1 : Triggerzeit wird berücksichtigt. Die absoluten Zeitstempel (Dautm und Uhrzeit) der Elemente werden beachtet.
DoMerge	Sollen die Werte des anzuhängenden Kanals auch eingefügt werden?
	0 : Nur Anhängen. Nur die Elemente werden angehängt, deren Zeitstempel nicht kleiner ist als der letzte von TsaChannel. Alle anderen Elemente werden ignoriert.
	1 : Die Elemente des anzuhängenden Kanals werden zeitrichtig zwischen den Elementen von TsaChannel eingefügt bzw. auch angehängt.
AllowDuplicate	Wenn ein Element in TsaChannel und dem anzuhängenden Kanal identisch ist, kann es nur einmal oder auch doppelt eingefügt werden.
	0 : Keine doppelten erlauben
	1 : Doppelte zulassen

Beschreibung:

Die Funktion gestattet das zeit-/x-richtige Aneinanderhängen bzw. Verschmelzen von Datensätzen.

Für das Resultat wird die zeitliche Auflösung (dx) von TsaChannel benutzt. Falls der Wert beim angehängten Kanal abweicht, werden dessen Zeiten ggf. passend gerundet und damit verändert.

Falls TsaChannel nicht verändert werden soll, muss auf einer Kopie gearbeitet werden.

Beide aneinander zu bindenden Kanäle dürfen keine Events haben.

In einer Sonderanwendung kann die Funktion analog zu EventJoin() wirken: Dann ist TsaChannel ein Kanal mit Events und TsaChannelJoin wird auf den Wert 0 gesetzt. Alle Events werden aneinander gebunden. Danach hat TsaChannel keine Events mehr.

Beispiele:

2 aufeinander folgende Messungen werden zusammengefasst:

```
TsaJoin ( Tsa1, Tsa2, 1, 0, 0 )
```

Alle Events werden zusammengefasst:

```
TsaJoin ( TsaEvents, 0, 1, 0, 0 )
```

TsaSaveAscii

Anwendungsbereich: Zeitstempel-ASCII-Daten

Ein Zeitstempel-Ascii-Kanal wird in eine Datei im Ascii-Format gesichert.

Deklaration:

```
TsaSaveAscii ( TsaChannel, Filename, Separator, Header, AbsTime, DecimalKomma, Columns )
```

Parameter:

TsaChannel	TsaChannel
Filename	Filename ist der gültige komplette Dateiname der Datei, die neu angelegt wird.
Separator	Ist der Ascii-Code für den Separator zwischen den Spalten, z.B. 9 für einen Tabulator, 44 für ein Komma, 32 für ein Leerzeichen.
Header	Header
	0 : kein Header erwünscht
	1 : Kanalname, Kommentar, Triggerzeit und Einheit werden an den Anfang der Datei geschrieben.
AbsTime	AbsTime
	0 : Zeitangaben in der Zeitspalte in relativer Zeit
	1 : Zeitangaben in der Zeitspalte in absoluter Zeit mit Datum und Uhrzeit
DecimalKomma	DecimalKomma
	0 : Dezimalpunkt
	1 : Dezimalkomma
Columns	Columns
	0 : Texte ohne Zeitspalte
	1 : Texte mit Zeitspalte (default)
	3 : Hex-Darstellung und Zeitspalte
	5 : CAN-, LIN-Botschaft und Zeitspalte
	7 : Flexray-Botschaft und Zeitspalte

Beschreibung:

Der Kanal darf auch Events haben.

Beispiele:

Der Kanal TsaChannel wird in die angegebene Datei 1.txt mit Tabulator als Trennzeichen mit Dateivorspann geschrieben. Typischerweise wird mit Dezimalpunkt und in relativer Zeit geschrieben.

```
TsaSaveAscii ( TsaChannel, "c:\imc\dat\1.txt", 9, 1, 0, 0, 1 )
```

TsaSetData

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird zu einem Element an der vorgegebenen Position der Inhalt neu gesetzt.

Deklaration:

```
TsaSetData ( TsaChannel, Position, Data )
```

Parameter:

TsaChannel	TsaChannel
Position	Position
Data	Data

Beschreibung:

Der Inhalt wird als Datensatz vorgegeben. Der Datensatz wird Byte für Byte übernommen, egal in welchem Zahlenformat er vorliegt. Sinnvollerweise sollte der Datensatz das Zahlenformat Byte haben. Die maximale Größe eines Elementes (siehe Zeitstempel-Ascii-Doku) darf nicht überschritten werden. Der übergebene Kanal wird dabei verändert. Die Position muss gültig sein, z.B. mit TsaFindTime(), TsaFindFirst(), TsaFindNext() oder TsaFindValidPos() bestimmt. Aufgrund der Veränderung eines Elementes verändern sich i. Allg. die Positionen der folgenden Elemente.

Beispiele:

Der Inhalt des 1. Elementes wird auf eine Rampe gesetzt.

Es wird 100 Byte, aufsteigend von 1 bis 100 enthalten.

```
Content = rampe ( 1, 1, 100 )
SetDatFormat ( Content, 5, 0, 255) ; Das Zahlenformat Byte erzwingen (0..255).

Position = TsaFindFirst ( TsaChannel )
TsaSetData ( TsaChannel, Position, Content )
```

TsaSetText

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird zu einem Element an der vorgegebenen Position der Inhalt neu gesetzt. Der Inhalt wird als Text vorgegeben.

Deklaration:

```
TsaSetText ( TsaChannel, Position, Text )
```

Parameter:

TsaChannel	TsaChannel
Position	Position
Text	Text

Beschreibung:

Der übergebene Kanal wird dabei verändert. Die Position muss gültig sein, z.B. mit TsaFindTime(), TsaFindFirst(), TsaFindNext() oder TsaFindValidPos() bestimmt. Aufgrund der Veränderung eines Elementes verändern sich i. Allg. die Positionen der folgenden Elemente.

Beispiele:

Der Inhalt des 1. Elementes wird auf einen festen Text gesetzt.

```
Text = "hello"
Position = TsaFindFirst ( TsaChannel )
TsaSetText ( TsaChannel, Position, Text )
```

TsaSetTime

Anwendungsbereich: Zeitstempel-ASCII-Daten

Bei einem Zeitstempel-Ascii-Kanal wird zu einem Element an der vorgegebenen Position die Zeit neu gesetzt.

Deklaration:

```
TsaSetTime ( TsaChannel, Position, Time )
```

Parameter:

TsaChannel	TsaChannel
Position	Position
Time	Time

Beschreibung:

Die Zeit darf nicht kleiner als die des Vorgängers sein und nicht größer als die des Nachfolgers. Zeiten müssen monoton wachsen, nicht aber streng monoton. D.h. die Zeiten benachbarter Elemente dürfen gleich sein. Die Zeit wird ggf. gerundet. Denn die Zeit wird intern als x0+i*dx dargestellt, wobei i >= 0 eine ganze Zahl ist. Der übergebene Kanal wird dabei verändert. Die Position muss gültig sein, z.B. mit TsaFindTime(), TsaFindFirst(), TsaFindNext() oder TsaFindValidPos() bestimmt.

Beispiele:

Die Zeit des 1. Elementes wird auf 3s gesetzt.

```
Position = TsaFindFirst ( TsaChannel )
Time = 3.0
TsaSetTime ( TsaChannel, Position, Time )
```

TsaTextToData

Anwendungsbereich: Zeitstempel-ASCII-Daten

Ein Text wird in einen Datensatz konvertiert. Der Datensatz erhält das Zahlenformat Byte.

Deklaration:

```
TsaTextToData ( Text ) -> Data
```

Parameter:

Text	Text
Data	Data

Beschreibung:

Beispiele:

```
Data = TsaTextToData ( Text )
```

TtoSv

Aus einem Text wird eine Zahl in verschiedenen Formaten gelesen.
Alternativer Name: **TzuEW**

Deklaration:

```
TtoSv ( TxText, TxFormat ) -> EwZahl
```

Parameter:

TxText	Text, aus dem die Zahl herausgelesen werden soll.
TxFormat	Formatangabe
	"e" : Eine reelle oder ganze Dezimalzahl wird gelesen. Als Dezimaltrenner wird ein Punkt erwartet. Beispiele: "375" oder "-1.3e-4" oder "3.124".
	"f" : Identisch zu "e".
	"a" : Eine Dezimalzahl wird gelesen. Dezimalpunkt oder -komma entsprechend der globalen Voreinstellung für das bevorzugte Dezimaltrennzeichen für reelle Zahlen ('Extra'/'Optionen'/'Anzeige' bzw. SetOption("Display.DecimalSeparator",...)
	"b" : Eine Zahl in binärer Darstellung wird gelesen, z.B. "100100".
	"x" : Eine Zahl in hexadezimaler Darstellung wird gelesen, z.B. "9B" oder "a21E".
EwZahl	Rückgabewert ist die gelesene Zahl.

Beschreibung:

Wenn der Text keine zulässige Zahlendarstellung enthält und somit keine Zahl bestimmt werden kann, wird eine Fehlermeldung erzeugt.

Im Formattext ist die Groß-/Kleinschreibung egal.

Beispiele:

```
x = TtoSv("22", "x")
b = TtoSv("001001", "b")
```

x erhält den Wert 34 (= 22Hex). b erhält den Wert 9.

Siehe auch:

TForm, TPart, TConv, SvToChar

TxArrayClean

In einem Textfeld werden alle Elemente entfernt, die einer gegebenen Bedingung entsprechen.

Deklaration:

```
TxArrayClean ( TxaTextfeld, EwBedingung [, EwGroßKlein] [, TxMuster] ) -> EwAnzahlGelöscht
```

Parameter:

TxaTextfeld	Textfeld
EwBedingung	Gibt an, welche Elemente gelöscht werden sollen.
	0 : Nach Ausführung enthält das Textfeld keine Duplikate mehr.
	1 : Es werden alle leeren Texte gelöscht (Länge 0).
	2 : Es werden alle leeren Texte sowie alle Texte, die nur Leerstellen enthalten, gelöscht. Als Leerstellen gelten das Leerzeichen (0x20) sowie Wagenrücklaufzeichen, Zeilenvorschubzeichen, vertikales Tabulatorzeichen, Seitenvorschub (0x09 bis 0x0D).
	3 : Es werden alle Texte gelöscht, die dem im 4. Parameter angegebenen Muster entsprechen.
	4 : Es werden alle Texte gelöscht, die nicht dem im 4. Parameter angegebenen Muster entsprechen.
	5 : Es werden alle Texte gelöscht, die dem im 4. Parameter angegebenen regulären Ausdruck entsprechen.
	6 : Es werden alle Texte gelöscht, die nicht dem im 4. Parameter angegebenen regulären Ausdruck entsprechen.
EwGroßKlein	Legt fest, ob beim Test auf Duplikate oder beim Vergleich eines Elements mit dem Vergleichsmuster zwischen Groß- und Kleinschreibung unterschieden wird. Bei [EwBedingung] = 1 oder 2 nicht verwendet. (optional , Standardwert: 0)
	0 : Groß- und Kleinschreibung eines Buchstabens wird als identisch betrachtet.
	1 : Groß- und Kleinschreibung eines Buchstabens wird als verschieden betrachtet.
TxMuster	Für [EwBedingung] = 3 oder 4 ein einfaches Vergleichsmuster (Jokerzeichen "*","?"). Für [EwBedingung] = 5 oder 6 ein regulärer Ausdruck. Sonst nicht verwendet. (optional)
EwAnzahlGelöscht	Anzahl der gelöschten Elemente (optional).

Beschreibung:

Im einfachen Vergleichsmuster ([EwBedingung] = 3 oder 4) können die Jokerzeichen "*" und "?" ihrer üblichen Interpretation verwendet werden. "?" steht dabei für genau ein beliebiges Zeichen, "*" für beliebig viele beliebige Zeichen.

Für ([EwBedingung] = 5 oder 6) wird für das Vergleichsmuster ein "regulären Ausdruck" verwendet. Damit können komplexere Vergleiche formuliert werden. Eine Zusammenfassung der Syntax regulärer Ausdrücke finden Sie in der Hilfe der Funktion TxRegexMatch().

Beispiele:

Duplikate löschen

```
txa = [ "RPM", "Current", "rpm", "current", "voltage", "rpm"]
; entferne alle Duplikate, Groß/Kleinschreibung egal
count = TxArrayClean(txa, 0)
; txa enthält [ "RPM", "Current", "voltage"], count hat den Wert 3

txa = [ "RPM", "Current", "rpm", "current", "voltage", "rpm"]
; entferne alle Duplikate, Groß/Kleinschreibung beachten
TxArrayClean(txa, 0, 1)
; txa enthält [ "RPM", "Current", "rpm", "current", "voltage"]

txa = [ "RPM", "rpm"]
; entferne alle Duplikate, Groß/Kleinschreibung beachten
count = TxArrayClean(txa, 0, 1)
; txa enthält [ "RPM", "rpm"], count hat den Wert 0
```

Leere Texte löschen

```
txa = [ "", "   ", "rpm", "current", "", "rpm"]
; entferne alle Texte mit Länge 0
TxArrayClean(txa, 1)
; txa enthält [ "   ", "rpm", "current", "rpm"]

txa = [ "", "   ", "rpm", "current", "", "rpm"]
; entferne alle Texte mit Länge 0 oder ausschließlich Leerstellen
TxArrayClean(txa, 2)
; txa enthält [ "rpm", "current", "rpm"]
```

Einfacher Mustervergleich

```
txa = [ "current1", "RPM2", "rpm1", "Current2", "voltage"]
; entferne alle Texte, die auf "1" enden
TxArrayClean(txa, 3, 0, "*1")
; txa enthält [ "RPM2", "Current2", "voltage"]

txa = [ "current1", "RPM2", "rpm1", "Current2", "voltage"]
; entferne alle Texte, die nicht aus 3 beliebigen Zeichen, gefolgt von einer 1, bestehen
TxArrayClean(txa, 4, 0, "???1")
; txa enthält [ "rpm1"]

txa = [ "current1", "RPM2", "rpm1", "Current2", "voltage"]
; entferne alle Texte, die mit einem großen oder kleinen 'C' beginnen
TxArrayClean(txa, 3, 0, "C*")
; txa enthält [ "RPM2", "rpm1", "voltage"]

txa = [ "current1", "RPM2", "rpm1", "Current2", "voltage"]
; entferne alle Texte, die mit einem großen 'C' beginnen
TxArrayClean(txa, 3, 1, "C*")
; txa enthält [ "current1", "RPM2", "rpm1", "voltage"]
```

Reguläre Ausdrücke

```
txa = ["asin3.dat", "sin.dat", "SIN11.DAT", "SIN2.raw", "sin1.txt"]
; alle Dateinamen entfernen, deren Name aus "sin" + Nummer und der Erweiterung "dat" oder "raw" besteht
TxArrayClean(txa, 5, 0, "sin\d+\.(dat|raw)")
; txa enthält  ["asin3.dat", sin.dat", "sin1.txt"]

txa = ["1", "s", "1e2", "+2e-3", "2,3", "2a", "-1.235"]
; behalte nur Texte, die eine Zahl repräsentieren
TxArrayClean(txa, 6, 0, "[-+]?([0-9]+\.?[0-9]*|\.[0-9]+)([eE][-+]?[0-9]+)?")
; txa enthält ["1", "1e2", "+2e-3", "-1.235"]
```

Siehe auch:

TxArrayDelete, TLike, TComp, TxRegexMatch

TxArrayCombine

Zwei Textfelder werden nach einem gegebenen Kriterium kombiniert.

Deklaration:

```
TxArrayCombine ( Txa1, Txa2, EwBedingung [, EwGroßKlein] ) -> TxaErgebnis
```

Parameter:

Txa1	Erstes Textfeld
Txa2	Zweites Textfeld
EwBedingung	Gibt an, nach welcher Bedingung die Textfelder kombiniert werden.
	0 : ODER (OR): Das Ergebnis enthält alle Elemente, die in [Txa1] oder [Txa2] enthalten sind. Das Ergebnis enthält zuerst die Elemente aus [Txa1] und dann die Elemente aus [Txa2], jeweils in ihrer originalen Reihenfolge.
	1 : UND (AND): Das Ergebnis enthält alle Elemente, die sowohl in [Txa1] als auch in [Txa2] enthalten sind. Die Reihenfolge im Ergebnis entspricht der Reihenfolge in [Txa1].
	2 : Exklusiv-ODER (XOR): Das Ergebnis enthält alle Elemente, die entweder in [Txa1] oder in [Txa2] enthalten sind. Das Ergebnis enthält zuerst die Elemente aus [Txa1] und dann die Elemente aus [Txa2], jeweils in ihrer originalen Reihenfolge.
EwGroßKlein	Groß-/Kleinschreibung beachten (optional , Standardwert: 0)
	0 : Groß- und Kleinschreibung eines Buchstabens wird als identisch betrachtet, z.B. 'A' = 'a'.
	1 : Groß- und Kleinschreibung eines Buchstabens wird als verschieden betrachtet. z.B. 'A' <> 'a'.
TxaErgebnis	Ergebnis der Kombination der beiden Textfelder.

Beschreibung:

Das Ergebnis-Textfeld enthält keine Duplikate.

Beispiele:

```
Txa1 = [ "current", "Voltage", "rpm" ]
txa2 = [ "RPM", "velocity", "current"]

; alle Einträge, die in Txa1 ODER txa2 enthalten sind:
; ----------------------------------------------------
txaResult = TxArrayCombine(Txa1, txa2, 0, 0)
; txaResult enthält ["current", "Voltage","rpm", "velocity" ]

;... Groß-/Kleinschreibung berücksichtigen:
txaResult = TxArrayCombine(Txa1, txa2, 0, 1)
; txaResult enthält ["current", "Voltage","rpm", "RPM", "velocity"]

; alle Einträge, die in Txa1 UND txa2 enthalten sind:
; ---------------------------------------------------
txaResult = TxArrayCombine(Txa1, txa2, 1, 0)
; txaResult enthält ["current", "rpm"]

;... Groß-/Kleinschreibung berücksichtigen:
txaResult = TxArrayCombine(Txa1, txa2, 1, 1)
; txaResult enthält ["current"]

; alle Einträge, die ENTWEDER in Txa1 ODER in txa2 enthalten sind:
; ----------------------------------------------------------------
txaResult = TxArrayCombine(Txa1, txa2, 2, 0)
; txaResult enthält ["Voltage", "velocity"]

;... Groß-/Kleinschreibung berücksichtigen:
txaResult = TxArrayCombine(Txa1, txa2, 2, 1)
; txaResult enthält ["Voltage", "rpm", "RPM", "velocity"]
```

Siehe auch:

TxArrayInsert, TxArrayClean, TxArraySort, TComp, AND

TxArrayCreate

Die Funktion erzeugt ein Textfeld mit der angebenen initialen Anzahl von Elementen.

Deklaration:

```
TxArrayCreate ( EwDimension ) -> TxaNeuesTextFeld
```

Parameter:

EwDimension	Anzahl der enthaltenen Text-Elemente.
TxaNeuesTextFeld	Textfeld mit der angegebenen Anzahl von Elementen.

Beschreibung:

Die Funktion erzeugt eine Textfeld-Variable. Das Textfeld wird mit der angegebenen Anzahl von Elementen angelegt, die zunächst alle leer sind.

Eine Textfeld-Variable beschreibt eine Liste von Texten, die unter einem Namen zusammengefasst und durch ihren Index adressiert werden können.

Zum Setzen und Abfragen der einzelnen Texte verwenden Sie die Indexschreibweise mit eckigen Klammern, also z.B.

```
FourTexts = TxArrayCreate(4)
FourTexts[2] = "Just another text"
SecondText = FourTexts[2]
```

Sie können die Anzahl der Elemente in einem Textfeld nachträglich mit der Funktion TxArraySetSize() verändern.

Wenn bei der Zuweisung eines Textes der angegebene Index um 1 größer als die momentane Größe ist, wird intern die Größe automatisch angepasst. Dies ermöglicht eine bequeme Verwendung, wenn z.B. das Textfeld in einer Schleife schrittweise aufgefüllt werden soll und die endgültige Größe noch nicht bekannt ist.

```
TxArray = TxArrayCreate(0) ; Initial size = 0
Index = 1
WHILE ...
   ...
   TxArray[Index] = AnyText  ; TxArray grows automatically
   Index = Index + 1
END
```

Sobald die finale Größe bekannt ist, ist es allerdings effizienter, diese bei TxArrayCreate() oder mittels TxArraySetSize() explizit vorzugeben.

Zum Aufzählen der Elemente eines Textfeldes ist die FOREACH ELEMENT - Schleife gut geeignet.

Alternativ können (kurze) Textfelder auch bequem mit einer Initialisierungsliste erzeugt werden:

```
TxArrayAllUsers = ["Joe", "John", "Jack"]
```

Beispiele:

In einem Textfeld werden verschiedene Informationen zu einer Spektralanalyse abgelegt und zusammen mit den eigentlichen Ergebnisdaten in einer Datei gespeichert.

```
Result1 = AmpSpectrumRMS(Channel1, 1024, 1, 50, 1, 0)
Result2 = AmpSpectrumRMS(Channel2, 1024, 1, 50, 1, 0)
CalculationParameters = TxArrayCreate(3)
CalculationParameters[1] = "Window length: 1024"
CalculationParameters[2] = "Overlapping: 50%"
CalculationParameters[3] = "Window function: Hamming"
FileSave("c:\tmp\result", "", 0, Result1, Result2, CalculationParameters)
```

In einem gegebenen Verzeichnis werden alle Dateien mit der Extension .dat aufgelistet. Der Pfadname aller Dateien, deren Dateizeit neuer als der 1.1.2012 ist, werden in einem Textfeld gespeichert.

```
FileListID = FsFileListNew("c:\copy", "*.dat", 0, 0, 0)
n = FsFileListGetCount(FileListID)
time_fence = TimeJoin(1, 1, 2012, 0, 0, 0)
FileNames = TxArrayCreate(n)
j = 1
FOR i = 1 TO n
   IF FsFileListGetTime(FileListID, i) >= time_fence
      FileNames[j] = FsFileListGetName(FileListID, i)
      j = j + 1
   END
TxArraySetSize(FileNames, j-1) ; reduce size to actual element count
END
```

Der Inhalt einer Multikanal-Datei wird komplett gelesen. Die Namen der geladenen Kanäle werden in einem Textfeld gemerkt.

```
fh = FileOpenDSF("Samplefile.dat", 0)
IF fh > 0
   count = FileObjNum?(fh)
   LoadedVarNames = TxArrayCreate(count)
   FOR i = 1 TO count
      name = FileObjName?(fh, i)
      <name> = FileObjRead(fh, i)
      LoadedVarNames[i] = name
   END
   FileClose(fh)
END
```

Siehe auch:

TxArraySetSize, TxArrayGetSize

TxArrayDelete

Element aus einem Textfeld löschen.

Deklaration:

```
TxArrayDelete ( TxArray, Index ) -> TxArrayResult
```

Parameter:

TxArray	Textfeld
Index	Index des Elements, das gelöscht werden soll
TxArrayResult	Ergebnistextfeld

Beschreibung:

Mit dieser Funktion können Elemente aus einem Textfeld gelöscht werden.

- Der Index darf im Bereich von 1 bis zur Größe des Textfeldes liegen.
- Liegt der Index außerhalb der Bereichsgrenzen des Textfeldes, so kommt es zum Abbruch der Sequenz.

Beispiele:

Das 3. Element aus dem Textfeld wird gelöscht

```
txArray=TxArrayCreate(5)
txArray[1]="One"
txArray[2]="Two"
txArray[3]="Three"
txArray[4]="Four"
txArray[5]="Five"
txArrayResult = TxArrayDelete(txArray,3)
```

Siehe auch:

TxArrayInsert

TxArrayGetSize

Die Funktion gibt die aktuelle Größe eines Textfeldes (also die Anzahl der enthaltenen Text-Elemente) zurück

Deklaration:

```
TxArrayGetSize ( TxaTextfeld ) -> EwDimension
```

Parameter:

TxaTextfeld	Abzufragende Variable vom Typ Textfeld.
EwDimension	Größe, Anzahl der Elemente.

Beschreibung:

Die Anzahl der Elemente eines Textfeldes wird beim Anlegen mit der Funktion TxArrayCreate() vorgegeben und kann nachträglich mit der Funktion TxArraySetSize() verändert werden.

Beispiele:

Ein Textfeld mit dem Namen [Log] wird aus einer Datei geladen und in eine Liste im aktuell geöffneten Panel eingetragen.

```
FileLoad("c:\logfiles\Log.dat", "", 0)
n = TxArrayGetSize(Log)
FOR i = 1 TO n
   PnInsertItem("list", 0, Log[i], 0)
END
```

Anmerkung: Für eine solche Aufzählung kann auch eine FOREACH ELEMENT-Schleife verwendet werden.

Ein Textfeld mit dem Namen [Log] wird aus einer Datei geladen und um die Einträge eines existierenden Textfeldes [CurrentLog] erweitert. Anschließend wird das aktualisierte Textfeld wieder gespeichert.

```
FileLoad("c:\logfiles\Log.dat", "", 0)
size = TxArrayGetSize(Log)
current_size = TxArrayGetSize(CurrentLog)
TxArraySetSize(log, size + current_size)
FOR i = 1 TO current_size
   Log[size+i] = CurrentLog[i]
END
FileSave("c:\logfiles\Log.dat", "", 0, log)
```

TxArrayInsert

Text oder Textfeld in ein Textfeld einfügen.

Deklaration:

```
TxArrayInsert ( TxArray1, TextOrTxArray2, Index ) -> TxArrayResult
```

Parameter:

TxArray1	In dieses Textfeld wird der Text oder das Textfeld2 eingefügt
TextOrTxArray2	Text oder Textfeld, der in Textfeld 1 eingefügt wird.
Index	Index von Textfeld1. Vor diesem Index wird der Text oder das Textfeld 2 eingefügt
TxArrayResult	Ergebnistextfeld

Beschreibung:

Mit dieser Funktion können Textfelder zusammengefügt werden.

- Der 2. Parameter darf ein Text oder ein Textfeld sein.
- Der Text oder das Textfeld 2 werden vor dem angegebenen Index eingefügt.
- Ist der Index -1, so wird der Text oder das Textfeld 2 an das Textfeld1 angefügt.
- Der Index darf im Bereich von 1 bis zur Größe des Textfeldes 1 liegen.
- Liegt der Index außerhalb der Bereichsgrenzen des Textfeldes 1, so kommt es zum Abbruch der Sequenz.

Beispiele:

TextArray2 wird an TextArray 1 angefügt.

```
txArrayResult=TxArrayInsert(txArray1,txArray2,-1)
```

TextArray2 wird vor Index=1 in TextArray 1 angefügt.

```
txArrayResult=TxArrayInsert(txArray1,txArray2,1)
```

Ein Text wird vor Index=3 in TextArray 1 angefügt.

```
txArrayResult=TxArrayInsert(txArray1,"Hello",3)
```

Siehe auch:

TxArrayDelete

TxArrayPart

Ein Abschnitt eines Textfeldes wird in ein neues Textfeld kopiert.

Deklaration:

```
TxArrayPart ( TxaQuelle, EwIndex, EwAnzahl ) -> TxaTeilKopie
```

Parameter:

TxaQuelle	Textfeld, dessen Elemente kopiert werden sollen.
EwIndex	Index des ersten zu kopierenden Elements. Das erste Element hat den Index 1.
EwAnzahl	Anzahl der zu kopierenden Elemente. -1, wenn bis zum letzten Element kopiert werden soll.
TxaTeilKopie	Neu erzeugtes Textfeld mit Kopien der angegebenen Elemente.

Beschreibung:

Die Funktion erzeugt ein neues Textfeld, das eine Kopie eines Abschnitts des Parameter-Textfeldes ist.

Wenn für [EwIndex] eine -1 angegeben wird oder ([EwIndex]+[EwAnzahl]) größer ist als die Anzahl der Elemente im Quell-Textfeld, wird bis zum Ende kopiert.

Beispiele:

Die letzten 3 Elemente eines Textfeldes werden in ein neues Textfeld kopiert.

```
n = TxArrayGetSize(txaSource)
txaSourcePart = TxArrayPart(txaSource, n-2, 3)
```

Ein Textfeld enthält eine größere Anzahl von Dateinamen. Die entsprechenden Dateien sollen geladen und ausgewertet werden.

```
allFiles = FsGetFileNames("c:\data", "*.raw", 0, 1, 0)
FOREACH ELEMENT file in allFiles
   !WorkWithFile(file) ; lädt eine Datei und führt die Auswertung durch
END
```

Dieser Ablauf soll durch Parallelisierung der Ausführung beschleunigt werden. Das Textfeld wird dazu in 4 Teile aufgespalten, die jeweils parallel verarbeitet werden.

```
allFiles = FsGetFileNames("c:\data", "*.raw", 0, 1, 0)
count = TxArrayGetSize(allFiles)
chunksize = floor(count/4)
BEGIN_PARALLEL
   !WorkWithFiles(TxArrayPart(allFiles, 1,              chunksize))
   !WorkWithFiles(TxArrayPart(allFiles, 1+ chunksize,   chunksize))
   !WorkWithFiles(TxArrayPart(allFiles, 1+ chunksize*2, chunksize))
   !WorkWithFiles(TxArrayPart(allFiles, 1+ chunksize*3, count-3*chunksize))
END_PARALLEL
```

Die Sequenzfunktion !WorkWithFiles ist dabei wie folgt definiert:

```
; Deklaration: !WorkWithFiles(Par1 [Datentyp: Textfeld])
FOREACH ELEMENT file in Par1
   !WorkWithFile(file)
END
```

Siehe auch:

TxArrayInsert, TxArrayGetCount

Unterstützt ab:

Version 2022

TxArraySetSize

Die Funktion ändert die Größe eines Textfeldes (also die Anzahl der enthaltenen Text-Elemente).

Deklaration:

```
TxArraySetSize ( TxaTextfeld, EwDimension )
```

Parameter:

TxaTextfeld	Zu ändernde Variable vom Typ Textfeld.
EwDimension	Neue Größe, Anzahl der Elemente.

Beschreibung:

Die Anzahl der Elemente eines Textfeldes wird beim Anlegen mit der Funktion TxArrayCreate() vorgegeben und kann nachträglich mit der Funktion TxArraySetSize() verändert werden.

Beim Verkleinern der Größe wird der Inhalt der verbleibenden Texte nicht verändert. Beim Vergrößern werden die neuen Elemente mit einem leeren Text initialisiert.

Wenn bei der Zuweisung eines Textes der angegebene Index um 1 größer als die momentane Größe ist, wird intern die Größe automatisch angepasst. Dies ermöglicht eine bequeme Verwendung, wenn z.B. das Textfeld in einer Schleife schrittweise aufgefüllt werden soll und die endgültige Größe noch nicht bekannt ist. Sobald die finale Größe bekannt ist, ist es allerdings effizienter, diese bei TxArrayCreate() oder mittels TxArraySetSize() explizit vorzugeben.

Beispiele:

Ein Textfeld mit dem Namen [Log] wird aus einer Datei geladen und um die Einträge eines existierenden Textfeldes [CurrentLog] erweitert. Anschließend wird das aktualisierte Textfeld wieder gespeichert.

```
FileLoad("c:\logfiles\Log.dat", "", 0)
size = TxArrayGetSize(Log)
current_size = TxArrayGetSize(CurrentLog)
TxArraySetSize(Log, size + current_size)
FOR i = 1 TO current_size
   Log[size+i] = CurrentLog[i]
END
FileSave("c:\logfiles\Log.dat", "", 0, log)
```

Siehe auch:

TxArrayCreate, TxArrayGetSize

TxArraySort

Die Elemente in einem Textfeld werden sortiert.

Deklaration:

```
TxArraySort ( TxaTextfeld, EwModus, EwReihenfolge )
```

Parameter:

TxaTextfeld	Textfeld
EwModus	Modus, nach dem die Elemente sortiert werden sollen.
	0 : Alphabetische Sortierung. Groß- und Kleinschreibung eines Buchstabens wird als identisch betrachtet, z.B. 'A' = 'a'.
	1 : Alphabetische Sortierung. Groß- und Kleinschreibung eines Buchstabens wird als verschieden betrachtet, z.B. 'A' < 'a'.
	2 : Natürliche Sortierung in aufsteigender Reihenfolge. Bei der natürlichen Sortierreihenfolge wird prinzipiell auch alphabetisch sortiert, allerdings werden Ziffern im Texten über ihren Wert sortiert. Dadurch wird z.B. 'Channel_2' vor 'Channel_10' angeordnet. Groß- und Kleinschreibung eines Buchstabens wird als identisch betrachtet.
EwReihenfolge	Reihenfolge, in der die Elemente sortiert werden sollen.
	0 : Aufsteigende Reihenfolge, das "kleinste" Element zuerst.
	1 : Absteigende Reihenfolge, das "größte" Element zuerst.

Beispiele:

```
txa = [ "channel_10", "Channel_2", "channel_1"]

; alpabetisch, ignoriere Groß-/Kleinschreibung
TxArraySort(txa , 0, 0)
; txa = [ "channel_1", "channel_10", "Channel_2"]

; alpabetisch, beachte Groß-/Kleinschreibung
TxArraySort(txa , 1, 0)
; txa = [ "Channel_2", "channel_1", "channel_10"]

; natürlich
TxArraySort(txa , 2, 0)
; txa = [ "channel_1", "Channel_2", "channel_10"]
```

Siehe auch:

TxArrayClean, TComp, Sort

TxArrayToChannel

Die Elemente eines Textfeldes werden in Zahlen gewandelt und daraus ein numerischer Datensatz gebildet.

Deklaration:

```
TxArrayToChannel ( TxaTextfeld, EwFormat [, EwErsatzWert] ) -> Daten
```

Parameter:

TxaTextfeld	Textfeld, aus dessen Elementen die Zahlen herausgelesen werden sollen.
EwFormat	Formatangabe
	0 : Die Zahlen werden als ganze oder reelle Dezimalzahl erwartet. Als Dezimaltrenner sind Punkt oder Komma erlaubt. Beispiele: "375" oder "-1.3e-4" oder "3,124" oder "2E11".
	1 : Die Zahlen werden in binärer Darstellung erwartet, z.B. "100100".
	2 : Die Zahlen werden in hexadezimaler Darstellung erwartet, z.B. "9B" oder "a21E". Der Vorsatz '0x' oder '0X' ist erlaubt, aber nicht notwendig.
EwErsatzWert	Optional, zu verwendender Ersatzwert, wenn ein Element nicht konvertiert werden kann. (optional)
Daten	Datensatz mit den gelesenen Werten.

Beschreibung:

Das Verhalten der Funktion, wenn ein Text-Element keine zulässige Zahlendarstellung enthält und somit keine Zahl bestimmt werden kann, ist von der Angabe des optionalen Parameters [EwErsatzWert] abhängig.

- Wenn angegeben, wird der hier spezifizierte Ersatzwert in das Ergebnis übernommen. Das Ergebnis hat immer genau so viele Werte, wie das Textfeld Elemente hat.
- Wenn weggelassen, wird die Konvertierung abgebrochen und die bisher erfolgreich gelesenen Werte zurückgeliefert. Die Länge des Ergebnisses entspricht dem Feld-Index des letzten erfolgreich konvertierten Textes.

Bei Konvertierung aus dem Dezimalformat ist das Datenformat des Ergebnisses '8 Byte reell' (double). Als Dezimaltrennzeichen kann ein Punkt oder Komma verwendet werden, ein Tausender-Trennzeichen ist nicht erlaubt.

Bei Konvertierung aus dem Binär- oder Hexadezimalformat ist das Datenformat des Ergebnisses '4 Byte ganzzahlig ohne Vorzeichen', es können also Zahlen im Bereich 0..4294967295 konvertiert werden.

Führende und schließende Leerzeichen in den Texten sind erlaubt.

Um einen Text, der eine Reihe von Zahlen enthält, in einen Datensatz zu wandeln, können Sie den Text zunächst mit TxSplit() in ein Textfeld zerlegen und dann die Funktion TxArrrayToChannel() anwenden.

Beispiele:

```
txa = ["-1", " 1.2e3", "1,23"]
data = TxArrayToChannel(txa, 0)
; data hat die Werte [-1, 1200, 1.23]
```

```
tx = "-1,  1.2e3, 1.23"
data = TxArrayToChannel(TxSplit(tx, ","), 0)
; data hat die Werte [-1, 1200, 1.23]
```

```
txa = ["11.2", "", "---", "1,23"]
data = TxArrayToChannel(txa, 0)
; data hat den Wert [11.2]
data = TxArrayToChannel(txa, 0, -1)
; data hat die Werte [11.2, -1, -1, 1.23]
```

```
txa = [" 1", "a  ", "FFFFFFFF", "10"]
data = TxArrayToChannel(txa, 2)
; data hat die Werte [1, 10, 4294967295, 16]
```

```
txa = [" 01", "-a", "", "FFFFFFFFF", "10"]
data = TxArrayToChannel(txa, 2)
; data hat den Wert [1]
data = TxArrayToChannel(txa, 2, 0)
; data hat die Werte [1, 0, 0, 0, 16]
```

```
txa = [" 1", "010", "101"]
data = TxArrayToChannel(txa, 1)
; data hat die Werte [1, 2, 5]
```

```
txa = ["-1", "12"]
data = TxArrayToChannel(txa, 1)
; data ist leer
data = TxArrayToChannel(txa, 1, 0)
; data hat die Werte [0,0]
```

Aus einer mehrspaltigen Textdatei mit Zahlenwerten (Spalten separiert durch Komma) soll die erste Spalte ausgelesen werden:

```
idFile = FileOpenASCII("z:\000.txt", 0)
TxaLines = TxArrayCreate(0)
ret = FileLineRead(idFile, TxaLines, 0) ; read all text lines from file
IF ret = 0
   TxaLines = TxRegexMatch(TxaLines, "^[^,]+", " ", 0, 0) ; split 1st column (comma separated) from line
   ; To get the 3rd column, you could use the following line (note the '{2}' for the the zero based column index):
   ; TxaLines = TxRegexMatch(TxaLines, "^(?:[^,]+,){2}([^,]+)", " ", 0, 1)
   channel = TxArrayToChannel(TxaLines, 0);
END
FileClose(idFile) ; close file processing
```

Anmerkung: Solche Import-Aufgaben können oft bequemer mit dem ASCII-Import-Assistenten erledigt werden.

Siehe auch:

TxFind

In dem Textfeld oder Text wird nach einem Text gesucht.

Deklaration:

```
TxFind ( TextOrTxArray, TxFindText, Occurrence, Options ) -> Result
```

Parameter:

TextOrTxArray	Text oder Textfeld, in dem der Text gesucht wird
TxFindText	Zu suchender Text
Occurrence	Das n-te Vorkommen des zu suchenden Textes
	-1 : Das letzte Vorkommen des Suchtextes wird ermittelt.
	1... : Das n-te Vorkommen des Suchtextes wird ermittelt.
	0 : Alle Vorkommen des Suchtextes werden ermittelt (nicht für Textfelder).
Options	Optionen für das Suchkriterium
	0 : Gross- und Kleinschreibung wird beachtet
	1 : Gross- und Kleinschreibung wird ignoriert
Result	Datensatz mit dem Ergebnis

Beschreibung:

In einem Textfeld oder Text wird nach dem Vorkommen von einem Text gesucht.

Das Ergebnis der Funktion ist ein normaler Datensatz.

Die Länge des Datensatzes entspricht der Dimension des Textfeldes, d.h. pro Element im Textfeld gibt es einen Wert im Datensatz.

Dieser Wert kennzeichnet die erste Position des n-ten Auftretens des Suchtextes im Element des Textfeldes.

Ist ein Wert = 0, so wurde der Suchtext in diesem Element des Textfeldes nicht gefunden.

Der Parameter Occurrence legt fest, beim wievielten Auftreten des Suchtextes, die Position bestimmt werden soll.

Ist der erste Parameter ein einfacher Text (kein Array), dann ist auch Occurence = 0 erlaubt. Es werden dann alle Auftreten des Suchtextes bestimmt und in einem Datensatz zurückgegeben. Dieser hat die Länge 0, wenn der Suchtext nicht vorhanden ist.

- Ungültige Optionen führen zum Abbruch der Sequenz.
- Ein leerer Suchtext führt zum Abbruch der Sequenz.

Beispiele:

In einem Textfeld wird nach dem Begriff "Form" gesucht.

```
txArray=TxArrayCreate(3)
txArray[1]="Dezimalformat und Binärformat"
txArray[2]="Konvertierung"
txArray[3]="Das Formular ist gültig"
result=TxFind( txArray,"Form",1,1)
```

Der Datensatz "result" enthält die Werte 8,0,5

```
result=TxFind( txArray,"Form",2,1)
```

Der Datensatz "result" enthält die Werte 24,0,0

Siehe auch:

TxReplace, TxRegexMatch, TxRegexReplace

TxFormatEx

Funktion zum Formatieren von Texten

Deklaration:

```
TxFormatEx ( TxFormat, Parameter1 [, Parameter2] [, Parameter3] [, Parameter4] [, Parameter5] [, Parameter6] [, Parameter7] [, Parameter8] ) -> TxResult
```

Parameter:

TxFormat	Formatierungszeichenfolge
Parameter1	1. Parameter für die Formatierung
Parameter2	2. Parameter für die Formatierung (optional)
Parameter3	3. Parameter für die Formatierung (optional)
Parameter4	4. Parameter für die Formatierung (optional)
Parameter5	5. Parameter für die Formatierung (optional)
Parameter6	6. Parameter für die Formatierung (optional)
Parameter7	7. Parameter für die Formatierung (optional)
Parameter8	8. Parameter für die Formatierung (optional)
TxResult	Ergebnistext

Beschreibung:

Die Funktion nimmt die Formatierungszeichenkette mit Text und Formatierungselementen sowie Parametern entgegen. Der Text wird zurückgegeben, und dabei werden die Parameter in der entsprechenden Formatierung eingefügt.

Die Parameter dürfen Texte oder Einzelwerte sein.

Die Formatierungszeichenkette besteht aus Text, Formatelementen und optional einem Kulturelement. Die Formatelemente werden durch die Parameter ersetzt. Ein Formatelement legt fest, welcher Parameter verwendet wird und wie er formatiert wird.

Ein Formatelement weist die folgende Form auf und besteht aus folgenden Komponenten:**{[T]Index[Ausrichtung][:FormatString]}**

Die übereinstimmenden geschweiften Klammern ("{" und "}") sind erforderlich, sie werden als Beginn und Ende eines Formatelements interpretiert. Um eine öffnende bzw. schließende geschweifte Klammer als Literal aus zugegeben, muss sie doppelt angegeben werden.

Die optionale Angabe **T** am Anfang eines Formatelements kennzeichnet den zugehörigen Parameter als Zeit. Mit dieser Angabe wird der betroffene Parameter als Zeitangabe im imc-Zeitformat (Sekunden ab 1.1.1980) interpretiert. Als Format-String muss eine Formatzeichenfolge für Datum und Uhrzeit angegeben werden. Ein Parameter kann nicht gleichzeitig als Zeitobjekt und als numerischer Wert formatiert werden.

Der obligatorische **Index** kennzeichnet den zu verwendenden Parameter. Er beginnt bei 1 und endet bei 8 (da nur maximal 8 Parameter übergeben werden können). Mehrere Formatelemente können auf den gleichen Parameter zurückgreifen, indem in den Formatelementen der gleiche Index verwendet wird. Für die Indizes in den Formatelementen müssen entsprechende Parameter übergeben werden. Ist in einem Formatelement ein Index angegeben, für den kein Parameter übergeben wurde, kommt es zum Abbruch der Sequenz.

Die optionale Angabe zur **Ausrichtung** ist eine ganze Zahl mit Vorzeichen. Sie gibt die gewünschte Feldbreite an. Bei einer positiven Zahl erfolgt eine rechtsbündige, im anderen Falle eine linksbündige Ausrichtung. Als Füllzeichen werden Leerzeichen verwendet. Wenn die Ausrichtung kleiner als die Länge der formatierten Zeichen ist, so wird die Ausrichtung ignoriert. Das Komma ist erforderlich, wenn eine Ausrichtung angegeben wird.

Der optionale **Format-String** steuert die Formatierung. Er muss zu dem Parameter passen. Ist der Parameter ein numerischer Wert, so muss eine numerische Formatzeichenfolge angewendet werden. Ist der Parameter ein Zeitwert (Datum/ Uhrzeit), so muss eine Formatzeichenfolge für Datum und Uhrzeit angegeben werden. Der Doppelpunkt ist erforderlich, wenn ein Format-String angegeben wird.

Beim Fehlen des Format-Strings wird das Formatzeichen "G" verwendet. Die Ausgabe wählt dann kompakteste Darstellung mit 7 (Eingangsdaten 4 Byte reell, 1 oder 2 Byte Integer) oder 15 signifikanten Stellen (8 Byte reell, 4 oder 8 Byte Integer).

Im folgenden sind die gültigen Formatzeichen für numerische und Datum-/ Zeitwerte zusammengestellt. Die Tabellen erheben keinen Anspruch auf Vollständigkeit. Die Beispiele setzen alle die Kultur en-US voraus.

Standardmäßige Format-Strings für numerische Werte:

Formatzeichen	Beschreibung
E oder e	Ausgabe im Exponentialformat. Durch das Anhängen einer Zahl kann die Anzahl der Nachkommastellen festgelegt werden. Beispiel: 1052.0329112756 ("E") -> 1.052033E+003
F oder f	Ausgabe einer Zahl im Dezimalformat. Durch das Anhängen einer Zahl kann die Anzahl der Nachkommastellen festgelegt werden. Der Standard sind zwei Stellen. Beispiel: -1234.56 ("F4") -> -1234.5600

G oder g	Die kompakteste Festkomma- oder wissenschaftliche Notation. Durch das Anhängen einer Zahl kann die Anzahl der signifikanten Stellen definiert werden. Beispiel: 123.4546 ("G4") -> 123.5
N oder n	Ganze Zahlen oder Dezimalzahlen mit Gruppentrennzeichen, einem Dezimaltrennzeichen und mit optionalem Minuszeichen. Durch das Anhängen einer Zahl kann die Anzahl der Nachkommastellen festgelegt werden. Die Ausgabe erfolgt im Format ddd.ddd.ddd,dd. Beispiel: -1234.56 ("N3") -> -1,234.560
P oder p	Ausgabe als Prozentzahl. Die Zahl wird mit 100 multipliziert und mit einem Prozentzeichen versehen. Durch das Anhängen einer Zahl kann die Anzahl der Nachkommastellen festgelegt werden. Beispiel: 1 ("P") -> 100.00 %
C oder c	Ausgabe im Währungsformat (einschließlich des Währungssymbols der aktuellen Ländereinstellung). Durch das Anhängen einer Zahl kann die Anzahl der Nachkommastellen festgelegt werden. Beispiel: 123.456 ("C") -> $123.46

Benutzerdefinierte Format-Strings für numerische Werte

Formatzeichen	Beschreibung
0	Die Zahl 0 dient als Platzhalter für eine Zahl. Nichtsignifikante Nullen werden durch die Zahl 0 dargestellt. Beispiel: 1234.5678 ("00000") -> 01235
#	Ersetzt das "#"-Symbol ggf. durch eine entsprechende vorhandene Ziffer; andernfalls wird keine Ziffer im Ergebnis angezeigt. Beispiel: 0.45678 ("#.##") -> .46
.	Das erste .-Zeichen im Format-String bestimmt die Position des Dezimaltrennzeichens im formatierten Wert. Beispiel: 0.45678 ("0.00") -> 0.46
,	Dient als Tausendertrennung. Beispiel: 2147483647 ("##,#") -> 2,147,483,647
%	Das Zeichen bewirkt die Multiplikation mit 100. Das Prozentzeichen wird mit ausgegeben. Beispiel: 0.3697 ("%#0.00") -> %36.97
E0, E+0, E-0, e0, e+0, e-0	Die Formatzeichen bewirken die Exponentialdarstellung einer Zahl. Mit E+0 und e+0 wird das positive Vorzeichen immer angezeigt, mit allen anderen immer nur das negative. Die Anzahl der Nullen bestimmt die Mindestanzahl von Ziffern des Exponenten. Beispiel: 987654 ("#0.0e0") -> 98.8e4
\	Das Zeichen, das auf das Escape- Zeichen folgt, wird als Literal und nicht als benutzerdefiniertes Formatzeichen interpretiert. Beispiel: 987654 ("\###00\#") -> #987654#

Standardmäßige Format-Strings für Datum- / Zeitwerte

Formatzeichen	Beschreibung
d	Kurzes Datum Beispiel: 2016-06-02T14:50:09 ("d")-> 6/2/2016
D	Langes Datum Beispiel: 2016-06-02T14:50:09 ("D")-> Thursday, June 02, 2016
f	Langes Datum mit kurzer Zeitangabe Beispiel: 2016-06-02T14:50:09 ("f")-> Thursday, June 02, 2016 2:50 PM
F	Langes Datum mit langer Zeitangabe Beispiel: 2016-06-02T14:50:09 ("F") -> Thursday, June 02, 2016 2:50:09 PM
g	Kurzes Datum mit kurzer Zeitangabe Beispiel: 2016-06-02T14:50:09 ("g") -> 6/2/2016 2:50 PM
G	Kurzes Datum mit langer Zeitangabe Beispiel: 2016-06-02T14:50:09 ("G") -> 6/2/2016 2:50:09 PM
M oder m	Tag und Monat Beispiel: 2016-06-02T14:50:09 ("M") -> June 02
R oder r	Datum nach dem Muster des RFC 1123 Beispiel: 2016-06-02T14:50:09 ("R")-> Thu, 02 Jun 2016 14:50:09 GMT
t	Kurze Zeitangabe Beispiel: 2016-06-02T14:50:09 ("t") -> 2:50 PM
T	Lange Zeitangabe Beispiel: 2016-06-02T14:50:09 ("T") -> 2:50:09 PM
Y oder y	Monat und Jahr Beispiel: 2016-06-02T14:50:09 ("Y") -> June, 2016

Benutzerdefinierte Format-Strings für Datum- / Zeitwerte

Formatzeichen	Beschreibung
dd / d	Monatstag mit/ohne führender Null Beispiel: 2016-06-02T14:50:09 ("dd") -> 02
ddd	Abkürzung des Wochentags Beispiel: 2016-06-02T14:50:09 ("ddd") -> Thu
dddd	Vollständiger Name des Wochentags Beispiel: 2016-06-02T14:50:09 ("dddd") -> Thursday
f / ff / .. / fffffff	Angabe einer Zehntelsekunde, Hundertstelsekunde ... Zehnmillionstelsekunde Beispiel: 2016-06-02T14:50:09.123456 ("fff ") -> 123
HH / H	Stundenangabe in 24-Stunden-Schreibweise mit/ohne führender Null Beispiel: 2016-06-02T14:50:09 ("HH") -> 14
hh / h	Stundenangabe in 12-Stunden-Schreibweise mit/ohne führender Null Beispiel: 2016-06-02T14:50:09 ("hh") -> 02
MM / M	Monat mit/ohne führender Null Beispiel: 2016-06-02T14:50:09 ("MM") -> 06
MMM	Abkürzung des Monatsnamens Beispiel: 2016-06-02T14:50:09 ("MMM") -> Jun
MMMM	Vollständiger Monatsname Beispiel: 2016-06-02T14:50:09 ("MMMM") -> June
mm / m	Minutenangabe mit/ohne führender Null Beispiel: 2016-06-02T14:50:09 ("mm") -> 50
ss / s	Sekundenangabe mit/ohne führender Null Beispiel: 2016-06-02T14:50:09 ("ss") -> 09
tt	Der AM/PM-Kennzeichner. Beispiel: 2016-06-02T14:50:09 ("tt") -> PM
yy / y	Zweistellige Jahreszahl mit/ohne führender Null Beispiel: 2016-06-02T14:50:09 ("yy") -> 16
yyyy	Vollständige Jahreszahl Beispiel: 2016-06-02T14:50:09 ("yyyy") -> 2016
zz / z	Zeitzonenangabe (+ oder -, gefolgt von der Stundenangabe; mit/ohne führender Null) Beispiel: 2016-06-02T14:50:09 ("zz") -> +02
zzz	Vierstellige Zeitzonenangabe Beispiel: 2016-06-02T14:50:09 ("zz") -> +02:00
:	Ausgabe des Zeittrennzeichens. Beispiel: 2016-06-02T14:50:09 (": ") -> :
/	Ausgabe des Datumstrennzeichens Beispiel: 2016-06-02T14:50:09 ("/ ") -> /

Am Anfang der Formatierungszeichenkette kann ein **Kulturelement** stehen.

Ist kein Kulturelement angegeben, so wird die aktuell im System eingestellte Kultur in der Formatierung angewendet. Dabei gelten die imc FAMOS- Einstellungen (siehe "Einstellungen für Anzeige / Kurvenfenster").

Wurde dort als bevorzugtes Dezimaltrennzeichen der Dezimalpunkt ausgewählt, so werden reelle Zahlen, Währungen oder Prozentausgaben immer mit dem Dezimalpunkt als Dezimaltrennzeichen ausgegeben.

Wurde dort als bevorzugtes Dezimaltrennzeichen das Dezimalkomma ausgewählt, so werden reelle Zahlen, Währungen oder Prozentausgaben mit dem Dezimalkomma als Dezimaltrennzeichen ausgegeben. Das Gruppentrennzeichen (Tausendertrennung) wird in beiden Fällen unterdrückt.

Sollen die Formatierungen einer bestimmten Kultur folgen, so kann ein Kulturelement angegeben werden. Das Kulturelement hat folgenden Aufbau: **{xx-yy}**. xx-yy steht für den Kulturnamen. Er besteht aus 2 Zeichen für den Language- Code nach ISO 639-1 und 2 Zeichen für Country/ Region- Code nach ISO 3166

Beispiele:

Kulturelemente	Bedeutung
{}	Invariante Kultur - Es wird keine Kultur berücksichtigt.
{en}	Neutrale Kultur englisch. Die Kultur ist mit der Sprache englisch verknüpft, nicht aber mit einem Land.
{en-US}	Kultur Englisch und Land USA
{en-GB}	Englisch - Grossbritanien
{de-DE}	Deutsch - Deutschland

{ja-JP}	Japanisch - Japan
{zh-CN}	Chinesische - Volksrepublik China
{zh-TW}	Chinesische - Taiwan

Wenn die Ressourcen für eine bestimmte Kultur nicht im Betriebssystem verfügbar sind, werden die Ressourcen für die zugehörige neutrale Kultur verwendet.

Beispiele:

Ausgabe des gleichen Parameters in Exponential- und Dezimaldarstellung:

```
para2=2341052.0329112756
result=TxformatEx("Wert exponential= {1:e6}, Wert dezimal={1:F6}",para2)
```

result="Wert exponential= 2.341052e+006, Wert dezimal=2341052.032911"

Bildung eines Kanalnamens aus festem Vorspann und angehänger Zahl:

```
index = 12
ChannelName=TxformatEx("Chan{1:#}", index)
```

ChannelName hat den Inhalt "Chan12".

Bildung eines Kanalnamens aus festem Vorspann und angehänger Zahl (4 Stellen, Nullen aufgefüllt):

```
index = 12
ChannelName=TxformatEx("Chan{1:0000}", index)
```

ChannelName hat den Inhalt "Chan0012".

Eine Fehlermeldung wird zusammengebaut:

```
TxChanname = "u1"
TxMsg = TxFormatEx( "Die Frequenz von {1} ist zu klein, {1}_ergebnis ist ungenau", TxChanname)
```

TxMsg hat den Inhalt "Die Frequenz von u1 ist zu klein, u1_ergebnis ist ungenau".

Ausgabe einer Zeitinformation mit langen Datum und Zeit im US-Format

```
time=TimeJoin(02,06,2016,14,50,09)
result=TxFormatEx("{en-US}{T1:F}",time)
```

result="Thursday, June 02, 2016 2:50:09 PM"

Ausgabe einer Zeitinformation mit benutzerdefinierter Formatierung im deutschen Format

```
time=TimeJoin(02,06,2016,14,50,09)
time=time+0.123456
result=TxFormatEx("{de}{T1:dd.MM.yyyy HH:mm:ss ffffff }",time)
```

result="02.06.2016 14:50:09 123456"

Siehe auch:

TxGetValidVarName

Funktion bildet aus einem Text einen gültigen Variablennamen

Deklaration:

```
TxGetValidVarName ( TxText ) -> TxVarName
```

Parameter:

TxText	Zeichenfolge
TxVarName	Variablenname

Beschreibung:

Ein beliebiger Text wird in einen gültigen FAMOS-Variablennamen umgewandelt.

- Dazu werden alle ungültigen Zeichen durch einen Unterstrich "_" ersetzt.
- Zu den ungültigen Zeichen zählen:
  - Alle Zeichen mit einem ASCII-Code < 32
  - < > + - ( ) * / ^ = { } [ ] | . @ ' : , Komma, Semikolon, Leerzeichen und Anführungszeichen
- Ist das 1. Zeichen eine Ziffer, so wird ein Unterstrich vorangestellt.
- Entspricht der Text einer FAMOS-Konstante, so wird ein Unterstrich vorangestellt.
- Hat der Text mehr als 255 Zeichen, so wird der Text nach 255 Zeichen abgeschnitten.
- Wird ein leerer Text übergeben, so kommt es zum Abbruch der Sequenz.

Beispiele:

```
TxGetValidVarName("ValidName") => "ValidName"
tab="~009"
TxGetValidVarName("Valid"+tab+"Name") => "Valid_Name"
TxGetValidVarName("1"+tab+"ValidName") => "_1_ValidName"
TxGetValidVarName("<Valid>.Name") => "_Valid__Name"
TxGetValidVarName("pi") => "_pi"
```

Siehe auch:

TForm

TxRegexMatch

Texte in einem Textfeld oder in einem Text mit Hilfe eines regulären Ausdrucks suchen.

Deklaration:

```
TxRegexMatch ( TextOrTxArray, TxPattern, TxResultSeparator, Options [, GroupIndex] ) -> TextOrTxArrayResult
```

Parameter:

TextOrTxArray	Text oder Textfeld, in dem das Muster gesucht wird
TxPattern	Suchmuster
TxResultSeparator	Zeichenfolge zum Trennen der Ergebnisse
Options	Optionen für die Suche.Die einzelnen Werte dürfen addiert werden.
	0 : Keine Option
	1 : IgnoreCase
	2 : Multiline
	4 : ExplicitCapture
	16 : Singleline
	32 : IgnorePatternWhitespace
GroupIndex	Gruppenindex, Parameter ist optional, Standardwert=0 (optional)
TextOrTxArrayResult	Enthält die gefundenen Texte

Beschreibung:

Ist der erste Parameter der Funktion ein Text, so ist der Rückgabewert auch ein Text.

Ist der erste Parameter der Funktion ein Textfeld, so ist der Rückgabewert auch ein Textfeld der selben Dimension.

Die Bedeutung der Optionen:

- IgnoreCase: Groß- und Kleinschreibung wird nicht berücksichtigt.
- Multiline: Ändert die Bedeutung von ^ und $, sodass sie jeweils dem Anfang und Ende einer beliebigen Zeile und nicht nur dem Anfang und Ende der gesamten Zeichenfolge entsprechen. Zirkumflex und Dollar passen zu Zeilenumbruch
- ExplicitCapture: Die Option sorgt dafür, dass alle Gruppen, mit Ausnahme der benannten Gruppen, nicht eingefangen sind. Mit dieser Option ist (Gruppe) das Gleiche wie (?Gruppe). Anstatt ExplicitCapture zu verwenden, können Sie diese Option auch aktivieren, in dem Sie (?n) an den Anfang des regulären Ausdrucks stellen
- Singleline: Ändert die Bedeutung des Punkts (.), sodass dieser jedem Zeichen entspricht (und nicht jedem Zeichen mit Ausnahme von Newline). Der Punkt passt zum Zeilenumbruch.
- IgnorePatternWhitespace:Ist die Option gesetzt, werden im Muster alle Leerzeichen, die nicht maskiert sind oder sich innerhalb von Zeichenklassen befinden, ignoriert. Damit lässt sich ein Ausdruck übersichtlicher gestalten. Außerdem werden alle Zeichen, die sich außerhalb einer Zeichenklasse zwischen zwei unmaskierten # befinden, ignoriert (einschließlich dem nächsten Zeilenumbruch). Damit lassen sich Kommentare in den regulären Ausdruck einfügen.
- Ungültige Optionen führen zum Abbruch der Sequenz.

Findet der reguläre Ausdruck im Eingangstext mehrere Texte, so werden alle im Ergebnistext, getrennt durch den TxResultSeparator, zurückgegeben.

Ein leeres Suchmuster führt zum Abbruch der Sequenz.

Das Muster eines regulären Ausdrucks kann Teilausdrücke einschließen, die definiert werden, indem ein Teil des Musters in Klammern () eingeschlossen wird. Dieser Teilausdruck bildet eine Gruppe. Die Gruppe mit dem Index =0 enthält die Zeichenfolge, die dem gesamten regulären Ausdruck entspricht (Standardeinstellung).

Soll als Ergebnis die Zeichenfolge, die einem Teilausdrucks entspricht, zurückgegeben werden, so ist der gewünschte Gruppenindex anzugeben.

Beispiel: Das folgende Muster wird verwendet, um ein Datum im amerikanischen Format zu erkennen.

```
pattern="\b(\d{1,2})/(\d{1,2})/(\d{2,4})\b"
```

Das Muster enthält 3 Teilausdrücke. Mit dem ersten Teilausdruck wird der Monat erkannt. Möchte man nur diese Information erhalten, so ist als Gruppenindex die 1 anzugeben.

```
str=TxFormatEx("{en-US}The date in American format is {T1:d}",currentTime)
; -> The date in American format is 10/11/2016
result= TxRegexMatch(str,pattern,";",0); -> 10/11/2016
result= TxRegexMatch(str,pattern,";",1); -> 10 Ergebnis des 1. Teilausdruck
```

Kurzübersicht Reguläre Ausdrücke

Reguläre Ausdrücke ("Regular Expressions") sind Muster mit einer standardisierten Syntax zur Beschreibung des Aufbaus einer Zeichenkette. Die Funktion verwendet das im Microsoft .NET Framework implementierte Paket für reguläre Ausdrücke. Im Folgenden werden Zusammensetzung und die wichtigsten Bestandteile der Muster aufgelistet. Eine vollständigere Auflistung finden Sie in der Microsoft .NET- Framework-Dokumentation.

Bei der Suche nach dem Muster wird der String Zeichen für Zeichen von links nach rechts gescannt. Jede neue Position wird mit dem Muster verglichen. Entspricht der Teilstring ab dieser Position dem Muster, so wird das als Erfolg gewertet. Die Suche geht dann aber erst ab dem Zeichen weiter, das der Fundstelle folgt.

Die Muster können aus normalen Zeichen, Escape- Sequenzen, Zeichenklassen, Quantifizierern, Gruppierungen und weiteren speziellen Zeichen, wie Alternativen, bestehen.

Die folgenden Zeichen werden als **Metazeichen** bezeichnet. Sie erfüllen spezielle Funktionen im Suchmuster. Soll ein Metazeichen als normales Zeichen und nicht als Metazeichen behandelt werden, muss dem Zeichen ein Backslash vorangestellt werden.

Metazeichen	
.	Steht für ein beliebiges Zeichen außer dem Newline
^	Suchmuster muss am Anfang des zu durchsuchenden Textes stehen bzw. negiert am Anfang einer Zeichenklasse das Muster
$	Suchmuster muss am Ende des zu durchsuchenden Textes stehen
|	Kennzeichnet eine Alternative im Suchmuster
?	Kennzeichnet eine Anzahl von Wiederholungen oder Leitet eine Modifizierer ein oder Kennzeichnet eine Lookahead- Assertion
+	Kennzeichnet eine Anzahl von Wiederholungen
*	Kennzeichnet eine Anzahl von Wiederholungen
()	Dient der Gruppierung
[]	Kennzeichnet eine Zeichenklasse
{}	Kennzeichnet einen Quantifizierer
-	Zeichenbereichen in Zeichenklassen
\	Hebt die besondere Bedeutung der Metazeichen auf, um sie als Zeichen zu suchen

Eine **Zeichenklasse** repräsentiert ein oder mehrere Zeichen. Sie wird durch die Metazeichen **[ ]** eingeschlossen. Innerhalb einer Zeichenklasse gilt:

\	zum Maskieren
^	zum Negieren der Zeichenklasse, wenn es das erste Zeichen ist.
-	zum Kennzeichnen eines Bereichs

Zeichenklassen	
.	Entspricht jedem Zeichen außer Newline. Über die Option "Singleline" ist einstellbar, ob auch Newlines enthalten sind.
\d	Beliebige Dezimalziffer; identisch mit [0-9]
\D	Alle Zeichen außer Ziffern; identisch mit [^0-9]
\s	Whitespace- Zeichen (Leerzeichen , Tabulator, Carriage Return, Line Feed
\S	steht für ein Zeichen, das kein Whitespace- Zeichen ist; identisch mit [^\s]
\w	Jedes alphanummerisches Zeichen und der Unterstrich; identisch mit [a-zA-Z_0-9]
\W	Alle Zeichen außer alphanummerische Zeichen ; identisch mit [^\w]
[abc]	a,b oder c
[a-z]	a bis z
[a-z&&[^bc]]	a bis z , ohne b und c

Die **Quantifizierer** stehen hinter einfachen Zeichen oder Zeichenklassen und bestimmen die Anzahl der Zeichen. Quantifizierer bestimmen, wie oft sich etwas wiederholen darf, was vor ihm steht.

Quantifizierer	
{n}	gibt an, dass das auf der linken Seite spezifizierte Zeichen genau n mal hintereinander vorkommen muss.
{n,}	spezifiziert, dass das Zeichen mindestens n mal vorkommen muss
{n,m}	spezifiziert, dass das Zeichen n bis m mal vorkommen muss
*	gibt an, dass das auf der linken Seite spezifizierte Zeichen keinmal, einmal oder beliebig oft in Folge vorkommen kann.

*?	gibt an, dass das auf der linken Seite spezifizierte Zeichen keinmal, einmal oder beliebig oft in Folge vorkommen kann.
+	gibt an, dass das auf der linken Seite spezifizierte Zeichen einmal vorkommen muss und beliebig oft in Folge vorkommen kann. Ist identisch mit {1,}. Der reguläre Ausdruck \w+ erkennt ein Wort.
?	gibt an, dass das auf der linken Seite spezifizierte Zeichen keinmal oder einmal vorkommen darf. Ist identisch mit {0,1}.

Normalerweise sind Quantifizierer "greedy". Sie bewirken, dass die Engine für reguläre Ausdrücke so viele Vorkommen bestimmter Muster wie möglich abgleicht. Das Anhängen des ?- Zeichens zu einem Quantifizierer (z.B. *?) macht ihn "lazy". Es bewirkt, dass so wenige Vorkommen wie möglich abgeglichen werden.

Reguläre Ausdrücke können neben den normalen Zeichen auch **Escape-Sequenzen** beinhalten. Escape-Sequenzen beginnen mit einem Backslash. Danach folgt ein Zeichen, das eine besondere Bedeutung besitzt und nicht als normales Zeichen ausgewertet werden soll.

Escape-Sequenzen	
\\	Backslash
\b	Backspace
\e	Escape
\t	HorizontalerTabulator
\r	Wagenrücklauf (Carriage Return)
\v	Vertikaler Tabulator
\f	Formularvorschub (Form Feed)
\ n	Zeilenvorschub (New Line)
\a	Alarm (Bell)
\xhh	spezifiziert ein Zeichen über einen zweistelligen hexadezimalen ASCII-Zeichencode
\uhhhh	spezifiziert ein Zeichen über einen vierstelligen hexadezimalen Unicode-Zeichencode
\cz	ASCII- Steuerzeichen , korrespondierend zu z

Mit **Ankern** können die möglichen Treffer für einen regulären Ausdruck eingeschränkt werden. Ein Anker legt fest, an welcher Stelle bzw. an welchen Stellen im Suchtext ein Treffer vorkommen muss, wobei insbesondere Anfang und Ende des Suchtextes von Bedeutung sind.

Anker	
^	Zeilenanfang. Der ^-Anker gibt an, dass das folgende Muster an der ersten Zeichenposition der Zeichenfolge beginnen muss. Kann durch den Modifizierer /m beeinflusst werden.
$	Zeilenende. Der $-Anker gibt an, dass das vorangehende Muster am Ende der Eingabezeichenfolge oder vor am Ende der Eingabezeichenfolge vorliegen muss.Kann durch den Modifizierer /m-beeinflusst werden.
\b	Wortgrenze
\B	Nicht an einer Wortgrenze
\A	Anfang einer Zeichenkette. Der \A-Anker gibt an, dass eine Übereinstimmung am Anfang der Eingabezeichenfolge vorliegen muss. Dies ist mit dem ^-Anker identisch, jedoch wird die Option Multiline von \A ignoriert. Daher kann hiermit in einer mehrzeiligen Eingabezeichenfolge nur eine Übereinstimmung nur am Anfang der ersten Zeile gesucht werden. Kann nicht durch einen Modifizierer beeinflusst
\Z	Ende einer Zeichenkette. Der \Z-Anker gibt an, dass eine Übereinstimmung am Ende der Eingabezeichenfolge oder vor \ n am Ende der Eingabezeichenfolge vorliegen muss. Dies ist mit dem $-Anker identisch, jedoch wird die Option Multiline von \Z ignoriert. Daher kann hiermit in einer mehrzeiligen Zeichenfolge nur nach einer Übereinstimmung mit dem Ende der letzten Zeile oder der letzten Zeile vor \ n gesucht werden. Kann durch einen Modifizierer nicht beeinflusst werden.
\z	Der \z-Anker gibt an, dass eine Übereinstimmung am Ende der Eingabezeichenfolge vorliegen muss. Im Gegensatz zum $-Sprachelement ignoriert \z die Option Multiline. Im Gegensatz zum \Z-Sprachelement stimmt \z nicht mit einem \ n-Zeichen am Ende einer Zeichenfolge überein. Daher kann hiermit nur eine Übereinstimmung mit der letzten Zeile der Eingabezeichenfolge gesucht werden. Kann durch einen Modifizierer nicht beeinflusst werden.
\G	Der \G-Anker gibt an, dass eine Übereinstimmung an dem Punkt vorliegen muss, an dem die vorherige Übereinstimmung endet.

Die Anker **^** und **\A** einerseits bzw. **$** und **\Z** andererseits unterscheiden sich nur dann, wenn im Mehrzeilen-Modus gearbeitet wird oder der Suchtext mehr als eine Zeile umfasst.

Um nach **alternativen** Zeichenketten zu suchen, wird das Zeichen "|" verwendet. Wird der Teilausdruck vor dem | nicht gefunden, so wird nach dem Teilausdruck hinter dem | gesucht.

Alternativen	
x|y	Alternative. x|y steht für x oder y. Die Alternative bezieht sich normalerweise auf den kompletten linken bzw. rechten Teilausdruck. So bedeutet abc|xyz -> abc oder xyz. Durch die Verwendung von Klammern können Alternativen eingeschränkt werden. So bedeutet ab(x|y)cd -> ab, gefolgt von x oder y , gefolgt von cd

Reguläre Ausdrücke erlauben das **Gruppieren** von Teilausdrücken. Sie grenzen die Teilausdrücke ab und zeichnen die Teilzeichenfolgen einer Eingabezeichenfolge auf. Eine Gruppe wird durch die Metazeichen ( ) eingeschlossen.

Gruppierung	
(x)	Übereinstimmender Teilausdruck x. Diese Gruppierung besitzt zwei Bedeutungen: Zum einen wird das Teilmuster gruppiert, so dass Sie z. B. Quantifizierer auf das Teilmuster anwenden oder durch die Alternative | verfeinern können. (\w\d){1,} steht beispielsweise für eine Kombination aus einer Dezimalziffer und einem Wortzeichen, die einmal oder beliebig oft vorkommen darf. x(\d{3}|###) steht für "x gefolgt von einer dreistelligen Ziffer oder gefolgt von ###". Zum anderen wird die Fundstelle in ein internes Array geschrieben, sodass Sie sich in Rückwärtsreferenzen darauf beziehen können.
(?:x)	Nicht erfassende Gruppe, gruppiert ein Teilmuster. Die Fundstelle des Teilmusters x wird aber nicht in das interne Array eingetragen.Das Teilmuster kann nicht Rückwärtsreferenzen verwendet werden.
(?<name>x)	Erfasst ein übereinstimmendes Teilmuster x und ermöglicht den Zugriff über einen Namen oder eine Zahl. Dabei ist name ein Gruppenname und Teilmuster x ein beliebiges Muster eines regulären Ausdrucks.Mit \k<name> kann auf den erfassten Teilausdruck zugegriffen werden. Alternativ kann mit _nummer_, wobei _nummer_ die Ordinalzahl des erfassten Teilausdrucks x ist, zugegriffen werden.
x (?=y)	Positive Lookahead- Assertion. Dieses Muster bewirkt, dass das Teilmuster x nur dann zum Erfolg führt, wenn das Teilmuster y direkt folgt. So können Sie Teilstrings finden, denen ein anderer String direkt angehängt ist. Das Muster in der Klammer ist im Suchergebnis nicht enthalten. Das Muster \d{3}(?=x) findet z. B. alle dreistelligen Zahlen, denen ein x folgt. Im String "123a 456x 789x" würden beispielsweise die Teilstrings "456" und "789" gefunden werden, nicht aber "123".
x(?!y)	Negative Lookahead- Assertion. Das Muster bewirkt, dass nur die Fundstellen in das Ergebnis übernommen werden, die dem Muster x entsprechen und denen nicht eine Zeichenkette folgt, der dem Muster y entspricht. Das Muster in der Klammer ist im Suchergebnis nicht enthalten.
(?<=x) y	Positive Lookbefore- Assertion. Das Muster y nach der Klammer wird nur erkannt, wenn das Muster x in der Klammer davor gefunden wird. Das Muster in der Klammer ist im Suchergebnis nicht enthalten. Mit dem Muster (?<=\b20)\d{2}\b können die letzten beiden Ziffern des Jahres ermittelt werden, deren Jahreszahl im 21. Jahrhundert ist. Bei einer Eingabezeichenkette "2010 1999 1861 2140 2009" wäre das Ergebnis 10 und 09.
(?<!x) y	Negative Lookbefore- Assertion. Das Muster y nach der Klammer wird nur erkannt, wenn das Muster x in der Klammer davor nicht gefunden wurde. Das Muster in der Klammer ist im Suchergebnis nicht enthalten. Das Muster (?<!halt)bar findet das bar nur wenn kein halt davor steht.
(?>x)	Das Sprachkonstrukt (?>x) deaktiviert die Rückverfolgung. Das Modul für reguläre Ausdrücke stimmt mit so vielen Zeichen in der Eingabezeichenfolge überein wie möglich. Wenn keine weitere Übereinstimmung möglich ist, findet keine Rückverfolgung statt, um alternative Musterübereinstimmungen zu versuchen. (D. h., der Teilausdruck sucht nur Entsprechungen für Zeichenfolgen, zu denen der Teilausdruck allein passen würde. Er versucht nicht, eine Entsprechung für eine Zeichenfolge auf Grundlage des Teilausdrucks und beliebige folgende Teilausdrücke zu finden.)
(?(x)y|z)	Bedingter Ausdruck Wenn der Ausdruck x gefunden wird, kommt der Ausdruck y zur Anwendung. Wenn der Ausdruck x nicht gefunden wurde, kommt der Ausdruck z zur Anwendung.

Mit () gruppierte gefundene Muster können in des Suche selbst oder später beim Ersetzen als **Rückwärtsreferenz** wieder verwendet werden. Es können maximal 9 Rückwärtsreferenzen verwendet werden.

Rückwärtsreferenz	
nummer	Mit _nummer_ kann auf den erfassten Teilausdrucks zugegriffen werden, wobei _nummer_ die Ordinalzahl des Teilausdrucks ist. (\w)(\w) Es wird nach 2 alphanumerischen Zeichen gesucht. Mit \2 kann beispielsweise der zweite Teilausdruck wieder verwendet werden.
\k<_name_>	Hat der Teilausdruck einen Namen (?<_name_>Teilmuster), so kann mit \k<_name_> auf den erfassten Teilausdruck zugegriffen werden.
	Mit (?:x) kann die Erzeugung einer Rückwärtsreferenz verhindert werden, so können Rückwärtsreferenzen eingespart, Geschwindigkeit verbessert und Speicherplatz gespart werden.

Das Verhalten eines regulären Ausdrucks kann durch **Modifizierer** gesteuert werden. Einige Modifizierer können auch durch Optionen ersetzt werden (z. B entspricht die Option IgnoreCase dem Modifizierer i, mit dem Unterschied das die Option auf den ganzen Ausdruck wirkt, der Modifizierer aber auch auf Teilausdrücke angewendet werden kann. Ein Modifizierer wird in der Form (?i) angegeben. Anstelle des Modifizierers i sind auch die Modifizierer m, s, und x erlaubt.

Modifizierer	
(?i)	Modifizierer i wird eingeschaltet, was bedeutet, dass Groß- und Kleinschreibung nicht berücksichtigt werden.
(?-i)	Modifizierer i wird ausgeschaltet
(?i-msx)	Modifizierer i wird eingeschaltet und die Modifizierer m s und x werden ausgeschaltet

(?m)	Wenn m aktiv, wird die Zeichenkette als mehrzeilig angesehen. "^" und "$" finden den Anfang und das Ende jeder internen Zeile. Wenn m deaktiviert ist, finden "^" und "$" den Anfang und das Ende der gesamten Zeichenkette
(?n)	Wenn n aktiv ist, werden die normalen Klammern (...), die normalerweise Text einfangen, zu nicht-einfangenden Klammern. Sie verhalten sich also wie (?:...) und gruppieren nur.
(?s)	Wenn s aktiv ist, wird die Zeichenkette als einzeilig angesehen. Der Punkt (.) findet alle Zeichen, einschließlich Zeilenumbrüche.
?(x)	Wenn x aktiv ist, werden Kommentare und Whitespaces (z.B. Leerzeichen) innerhalb des Musters ignoriert.

Beispiele:

Aus einem Textfeld, das Kanalnamen enthält, sollen alle Kanalnamen gesucht werden, die mit "sintest" beginnen, dann eine beliebige Zahl haben und mit der Erweiterung "dat" oder "raw" enden. Groß-/Kleinschreibung wird nicht unterschieden.

Das Suchmuster (?i)^sintest\d*\.(dat|raw)$ bedeutet:

(?i)	Modifizierer: Groß- und Kleinschreibung wird nicht berücksichtigt. Anstelle des Modifizierers kann auch mit der Option IgnoreCase gearbeitet werden.
^sintest	Der Eingangstext muss mit sintest beginnen.
\d*	Danach können keine, eine oder beliebig viele Ziffern folgen.
\.	Der Punkt muss gefunden werden.
(dat|raw)$	Die Erweiterung darf auf dat oder raw enden.

```
txArray= TxArrayCreate(5)
txArray[1]="sin3.dat"
txArray[2]="SINTEST1.DAT"
txArray[3]="SINTEST2.DAT"
txArray[4]="SINTEST3.RAW"
txArray[5]="SPANNUNG.DAT"
separator=";"
pattern = "(?i)^sintest\d*\.(dat|raw)$"
erg= TxRegexMatch(txArray,pattern,separator,0)
```

Das Textfeld "erg" enthält folgende Zeichenketten:

```
[1]
[2] SINTEST1.DAT
[3] SINTEST2.DAT
[4] SINTEST3.RAW
[5]
```

Aus einer Textfeld mit beliebigen Texten, sollen aus jedem Element die Gleitkommazahlen extrahiert werden.

Das Suchmuster [-+]?([0-9]+\.?[0-9]*|\.[0-9]+)([eE][-+]?[0-9]+)? bedeutet:

[+-]?	Ein Plus oder Minuszeichen kann einmal oder keinmal vorkommen
(	Beginn der Gruppe 1 Beginn des 1. Teils der Alternative in Gruppe 1
[0-9]+	Eine Dezimalziffer muss mindestens einmal vorkommen, und dann beliebig oft in Folge
\.?	Der Dezimalpunkt kann einmal oder keinmal vorkommen
[0-9]*	Eine Dezimalziffer kann keinmal, einmal oder beliebig oft vorkommen.
	Ende des 1. Teils der Alternative
|	Oder
	Beginn des 2. Teils der Alternative
\.	Der Dezimalpunkt muss vorkommen
[0-9]+	Eine Dezimalziffer muss mindestens einmal vorkommen, und dann beliebig oft in Folge
	Ende des 2. Teils der Alternative
)	Ende der Gruppe 1
(	Beginn der Gruppe 2
[eE]	Das Exponentialzeichen muss vorkommen
[-+]?	Plus oder Minuszeichen können einmal oder keinmal vorkommen
[0-9]+	Eine Dezimalziffer muss mindestens einmal vorkommen, und dann beliebig oft in Folge

)	Ende der Gruppe2
?	Die Gruppe 2 kann einmal oder keinmal vorkommen

```
txArray= TxArrayCreate(3)
txArray[1]="Das Minimum liegt bei -123.8 und das Maximum +1234."
txArray[2]="Bei 1.9e3°C hat die Temperatur den Grenzwert überschritten."
txArray[3]="Diese Zeile enthält keine Zahl."
pattern ="[-+]?([0-9]+\.?[0-9]*|\.[0-9]+)([eE][-+]?[0-9]+)?"
erg= TxRegexMatch(txArray,pattern,";",0)
```

Das Textfeld "erg" enthält folgende Zeichenketten:

```
[1] -123.8;+1234.
[2] 1.9e3
[3]
```

Mit dem nachstehenden Aufruf wird aus der Angabe von Längen- und Breitengrad die zugehörige Adresse geliefert (reverse Geocoding).

Dazu wird die Suchmaschine von OpenStreetMap verwendet. Der Parameter 'format' spezifiziert das gewünschte Format der Antwort (hier XML, eine Alternative wäre JSON), die Parameter 'lat' (latitude) und 'lon' (longitude) geben Längen- und Breitengrad an.

```
answer = HttpGetText( "http://nominatim.openstreetmap.org/reverse?format=xml&lat=52.541861&lon=13.3869693")
```

Der gelieferte Text enthält u.a. den folgenden Abschnitt:

```
<road>Voltastraße</road>
<suburb>Gesundbrunnen</suburb>
<city_district>Mitte</city_district>
<state>Berlin</state>
<postcode>13355</postcode>
```

Um aus solchen strukturierten Anworten einzelne Felder zu extrahieren, empfiehlt sich die Funktion TxRegexMatch():

```
NameOfTheRoad = TxRegexMatch( answer, "<road>(.*?)</road>", ",", 0, 1)
; NameOfTheRoad hat jetzt den Inhalt 'Voltastraße'.
```

Wenn statt dessen die Anwort im JSON-Format geliefert wird:

```
answer = HttpGetText( "http://nominatim.openstreetmap.org/reverse?format=json&lat=52.541861&lon=13.3869693")
;answer enthält u.a.: ...{... "road":"Voltastraße","suburb":"Gesundbrunnen", ...}
NameOfTheRoad = TxRegexMatch( answer, "~034road~034:~034(.*?)~034", ",", 0, 1)
```

Siehe auch:

TxFind, TxReplace, TxRegexReplace

TxRegexReplace

Texte in einem Textfeld oder in einem Text mit Hilfe eines regulären Ausdrucks ersetzen.

Deklaration:

```
TxRegexReplace ( TextOrTxArray, TxPattern, TxReplace, Options ) -> TextOrTxArrayResult
```

Parameter:

TextOrTxArray	Text oder Textfeld, in dem das Muster ersetzt wird
TxPattern	Muster
TxReplace	Zu ersetzender Text
Options	Optionen für das Ersetzen. Die einzelnen Werte dürfen addiert werden.
	0 : Keine Option
	1 : IgnoreCase
	2 : Multiline
	4 : ExplicitCapture
	16 : Singleline
	32 : IgnorePatternWhitespace
TextOrTxArrayResult	Enthält die ersetzten Texte

Beschreibung:

Die Bedeutung der Optionen:

- IgnoreCase: Groß- und Kleinschreibung wird nicht berücksichtigt.
- Multiline: Ändert die Bedeutung von ^ und $, sodass sie jeweils dem Anfang und Ende einer beliebigen Zeile und nicht nur dem Anfang und Ende der gesamten Zeichenfolge entsprechen. Zirkumflex und Dollar passen zu Zeilenumbruch
- ExplicitCapture: Die Option sorgt dafür, dass alle Gruppen, mit Ausnahme der benannten Gruppen, nicht eingefangen sind. Mit dieser Option ist (Gruppe) das Gleiche wie (?Gruppe). Anstatt ExplicitCapture zu verwenden, können Sie diese Option auch aktivieren, in dem Sie (?n) an den Anfang des regulären Ausdrucks stellen
- Singleline: Ändert die Bedeutung des Punkts (.), sodass dieser jedem Zeichen entspricht (und nicht jedem Zeichen mit Ausnahme von Newline). Der Punkt passt zum Zeilenumbruch.
- IgnorePatternWhitespace:Ist die Option gesetzt, werden im Muster alle Leerzeichen, die nicht maskiert sind oder sich innerhalb von Zeichenklassen befinden, ignoriert. Damit lässt sich ein Ausdruck übersichtlicher gestalten. Außerdem werden alle Zeichen, die sich außerhalb einer Zeichenklasse zwischen zwei unmaskierten # befinden, ignoriert (einschließlich dem nächsten Zeilenumbruch). Damit lassen sich Kommentare in den regulären Ausdruck einfügen.
- Ungültige Optionen führen zum Abbruch der Sequenz.
- Ein leeres Suchmuster führt zum Abbruch der Sequenz.

Eine **Kurzübersicht über die Syntax Regulärer Ausdrücke** finden Sie in der Hilfe zur Funktion TxRegexMatch()

Der Parameter **TxReplace** kennzeichnet den ersetzenden Text für jeden Fund im Eingangstext. Er kann aus einer beliebigen Kombination aus normalem Text und **Substitutionen** bestehen.

Substitution	Beschreibung
$number	Fügt den letzten gefundenen Teiltext ein, der von der Erfassungsgruppe mit der angegebenen Nummer gefunden wurde.
${name}	Fügt den letzten gefundenen Teiltext ein, der von der Erfassungsgruppe mit den angegeben Namen - benannt mittels (?<name>) - gefunden wurde.
$$	Fügt ein einzelnes '$'-Zeichen ein.
$&	Fügt eine Kopie der gesamten Übereinstimmung in die Ersetzungszeichenfolge ein.
$`	Fügt den gesamten Text der Eingabezeichenfolge vor der Übereinstimmung ein.
$'	Fügt den gesamten Text der Eingabezeichenfolge nach der Übereinstimmung ein.
$+	Fügt die letzte erfasste Gruppe in die Ersetzungszeichenfolge ein.
$_	Fügt die gesamte Eingabezeichenfolge in die Ersetzungszeichenfolge ein.

Beispiele:

Ersetze alle Leerzeichen in einem Text durch einen Unterstrich:

```
TxPatched = TxRegexReplace( "Text with spaces", "\s", "_", 0)
```

TxPatched hat den Inhalt "Text_with_spaces"

Entferne alle runden Klammern und Leerzeichen aus einem Text:

```
TxPatched = TxRegexReplace( "(0190) 1234567", "[\s()]", "", 0)
```

TxPatched hat den Inhalt "01901234567"

Entferne alle führenden Nullen aus einem Text:

```
TxPatched = TxRegexReplace( "0001023", "^0*", "", 0)
```

TxPatched hat den Inhalt "1023"

Das aktuelle Datum wird vom amerikanischen Format in das deutsche Format umgewandelt. Dazu wird in der ersten Zeile die aktuelle Systemzeit ermittelt. In der zweiten Zeile wird die Zeit in ein kurzes Datum im amerikanischen Format formatiert.

Das Suchmuster \b(?<month>\d{1,2})/(?<day>\d{1,2})/(?<year>\d{2,4})\b bedeutet:

\b	Vergleich beginnt an einer Wortgrenze
(?<month>\d{1,2})	Ein oder zwei Dezimalzahlen werden in der benannten Gruppe month zwischengespeichert.
/	Es muss ein Schrägstrich gefunden werden.
(?<day>\d{1,2})	Ein oder zwei Dezimalzahlen werden in der benannten Gruppe day zwischengespeichert.
/	Es muss ein Schrägstrich gefunden werden.
(?<year>\d{2,4})	Ein oder zwei Dezimalzahlen werden in der benannten Gruppe year zwischengespeichert.
\b	Der Vergleich endet an einer Wortgrenze

Im zu ersetzenden Text wird mittels $ auf die benannten Gruppen zugegriffen. Zwischen den Gruppen wird ein Punkt eingefügt.

```
currentTime=TimeSystem?()
strDate=TxFormatEx("{en-US}{T1:d}",currentTime); -> 6/30/2016
pattern="\b(?<month>\d{1,2})/(?<day>\d{1,2})/(?<year>\d{2,4})\b"
replace="${day}.${month}.${year}"
germanDate=TxRegexReplace( strDate,pattern,replace,0); -> 30.06.2016
```

Mit dem folgenden Ausdruck soll ein Text in einen gültigen imc FAMOS- Variablennamen um-gewandelt werden. Alle ungültigen Zeichen werden durch einen Unterstrich "_" ersetzt. Zu den ungültigen Zeichen zählen alle Zeichen mit einem ASCII-Code <32. und < > >+ - ( ) * / ^ = { } [ ] | . @amp; , ; " ' sowie Leerzeichen. Ist das 1. Zeichen eine Ziffer, so wird ein Unterstrich vorangestellt.

Das Suchmuster (^\d)|[^\w!$%&?~#] bedeutet:

(^\d)	ist das erste Zeichen eine Ziffer, so wird es in einer Gruppe zwischengespeichert
[^w!$%&?~#]	Der Inhalt der Zeichenklasse findet alle Zeichen, die nicht 0.9, a-z,_, A-Z und nicht zu den Zeichen !$% &?~# passen.

Im Ersatztext wird _$1 angegeben, d.h. jedes Zeichen das durch das Suchmuster gefunden wurde, wird durch einen Unterstrich und dem Inhalt der ersten Gruppe $1 ersetzt. Die erste Gruppe ist nur dann gesetzt, wenn das erste Zeichen eine Ziffer ist.

```
pattern ="(^\d)|[^\w!$%&?~#]"
varname=TxRegexReplace( "Valid Name?",pattern,"_$1",0);-> Valid_Name?
varname=TxRegexReplace( "1Valid Name+",pattern,"_$1",0);-> _1Valid_Name_
```

Anmerkung: Die Funktion TxGetValidVarName() ist für vorstehende Aufgabe besser geeignet.

Siehe auch:

TxFind, TxReplace, TxRegexMatch

TxReplace

In dem Textfeld oder Text wird nach einem Text gesucht und dieser durch einen anderen ersetzt.

Deklaration:

```
TxReplace ( TextOrTxArray, TxFindText, TxReplaceText, Occurrence, Options ) -> TextOrTxArrayResult
```

Parameter:

TextOrTxArray	Text oder Textfeld, in dem der Text gesucht und ersetzt wird
TxFindText	Zu suchender Text
TxReplaceText	Zu ersetzender Text
Occurrence	Das n-te Vorkommen des zu suchenden Textes
	-1 : Das letzte Vorkommen des Suchtextes wird ersetzt.
	0 : Alle Vorkommen des Suchtextes werden ersetzt.
	1... : Das n-te Vorkommen des Suchtextes wird ersetzt.
Options	Optionen für das Suchen und Ersetzen
	0 : Gross- und Kleinschreibung wird beachtet.
	1 : Gross- und Kleinschreibung wird ignoriert.
TextOrTxArrayResult	Enthält die ersetzten Texte

Beschreibung:

In einem Textfeld oder Text wird nach einem Text gesucht und dieser durch einen anderen ersetzt.

Ist der erste Parameter ein Textfeld, so ist das Ergebnis der Funktion ein Textfeld. Es hat die gleiche Dimension wie das übergebene.

Ist der erste Parameter ein Text, so ist das Ergebnis der Funktion ein Text.

Im resultierenden Text oder Textfeld sind die gefundenen Texte durch die Ersatztexte ersetzt.

Der Parameter Occurrence legt fest, beim wievielten Vorkommen des Suchtextes, der Text ersetzt wird.

- Ungültige Optionen führen zum Abbruch der Sequenz.
- Ein leerer Suchtext führt zum Abbruch der Sequenz.

Beispiele:

Ersetze alle Leerzeichen in einem Text durch einen Unterstrich:

```
TxPatched = TxReplace( "Text with spaces", " ", "_", 0, 0)
```

TxPatched hat den Inhalt "Text_with_spaces"

Entferne alle Bindestriche aus einem Text:

```
TxPatched = TxReplace( "FF-1A-13-2E", "-", "", 0, 0)
```

TxPatched hat den Inhalt "FF1A132E""

Aus einem Dateinamen wird ein neuer Dateiname für das berechnete Ergebnis abgeleitet:

```
TxInputFileName = "c:\Measurements.raw\Test1.raw"
TxOutputFileName = TxReplace(TxInputFileName, ".raw", "_processed.dat", -1, 1)
```

TxOutputFileName hat den Inhalt 'c:\Measurements.raw\Test1_processed.dat'

In einem Textfeld mit vorbereiteten Fehlermeldungen wird der Platzhalter "%1" durch einen Variablennamen ersetzt.

```
txArray=TxArrayCreate(4)
txArray[1]="%1 hat den falschen Datentyp"
txArray[2]="Nicht genügend Speicher"
txArray[3]="Zulässiger Wertebereich von %1 ist überschritten"
txArray[4]="Die Abtastzeit von %1 ist zu klein, %1_ergebnis ist ungenau."
txArrayResult=TxReplace( txArray,"%1","Channel1",0,0)
```

Das Textfeld "txArrayResult" enthält folgende Zeichenketten:

```
txArray[1]="Channel1 hat den falschen Datentyp"
txArray[2]="Nicht genügend Speicher"
```

```
txArray[3]="Zulässiger Wertebereich von Channel1 ist überschritten"
txArray[4]="Die Abtastzeit von Channel1 ist zu klein, Channel1_ergebnis ist ungenau."
```

Siehe auch:

TxFind, TxRegexMatch, TxRegexReplace

TxSplit

Funktion zum Splitten eines Textes

Deklaration:

```
TxSplit ( TxText, TxSeparator ) -> TxArrayResult
```

Parameter:

TxText	Zeichenfolge, die gesplittet werden soll
TxSeparator	Zeichenfolge mit den Trennzeichenfolgen
TxArrayResult	Ergebnistextfeld

Beschreibung:

Der Text TxText wird anhand der Trennzeichenfolgen in TxSeparator aufgetrennt.

Die gesplitteten Zeichenfolgen werden in einem TextFeld zurückgegeben.

- Der Text TxSeparator kann mehrere Trennzeichenfolgen enthalten.
- Die einzelnen Trennzeichenfolgen sind durch ein Semikolon ; zu trennen.
- Soll das Semikolon als Trennzeichen verwendet werden, so ist \; anzugeben.
- Trennzeichenfolgen sind in den Elementen des zurückgegebenen Textfeldes nicht enthalten.
- Ist der TxSeparator leer, so wird der Text an den so genannten Whitespaces (Leerzeichen,Tabulator,Zeilenende) aufgetrennt.
- Die Funktion berücksichtigt die Groß-/Kleinschreibung.

Beispiele:

Der Text wird am Semikolon, Komma oder Punkt aufgetrennt.

```
txArray1=TxSplit("Das ist ein Text. Dieser soll zerlegt werden; gesplittet werden." ,"\;;,;.;")
```

Ergebnis in txArray1:

```
[1] Das ist ein Text
[2] Dieser soll zerlegt werden
[3] gesplittet werden
```

.

Der Text wird an den Whitespaces aufgetrennt.

```
tab="~009"
cr="~010"
txArray2=TxSplit("Das"+tab+"ist der Text"+cr+";Der soll gesplittet werden." ,"")
```

Ergebnis in txArray2:

```
[1] Das
[2] ist
[3] der
[4] Text
[5] ;Der
[6] soll
[7] gesplittet
[8] werden.
```

Siehe auch:

TxFind, TxRegexMatch

TxToClipboard

Die Funktion kopiert einen Text bzw. ein Textfeld in die Windows-Zwischenablage.

Deklaration:

```
TxToClipboard ( TxOrTxArray ) -> EwErfolg
```

Parameter:

TxOrTxArray	Zu kopierender Text/Textfeld
EwErfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Der Inhalt des Parameters wird in die Windows-Zwischenablage kopiert (ANSI-Text-Format).

Bei Textfeldern bildet jedes Element eine neue Zeile.

Beispiele:

Der komplette Inhalt einer Tabelle in einem Panel wird in die Windows-Zwischenablage kopiert.

Die Spalten werden durch ein Tabulatorzeichen getrennt.

```
cols = PnTableColumns?("Table1")
rows = PnTableRows?("Table1")
tx = ""
FOR r = 1 TO rows
   FOR c = 1 TO cols
      tx = tx + PnTableGetCellText("Table1", c, r)
      IF c < cols
         tx = tx + "~009"  ; Tabulator
      END
   END
   tx= tx + "~013~010"   ; Carriage return/Line feed
END
TxToClipboard(tx)
```

Siehe auch:

TxWhere

Die Position einer Zeichenfolge in einem Text wird ermittelt.
Alternativer Name: **TxWo**

Deklaration:

```
TxWhere ( TxText, TxSuche ) -> EwPosition
```

Parameter:

TxText	Zu untersuchender Text
TxSuche	Der Text, nach dem gesucht werden soll. Groß- und Kleinschreibung wird nicht unterschieden.
EwPosition	Die Position der gesuchten Zeichenfolge im Text. Die erste mögliche Position ist 1. Wird die Zeichenfolge nicht gefunden, wird eine 0 zurückgegeben

Beschreibung:

Ein Text wird nach einer Zeichenfolge durchsucht. Wird die Zeichenfolge gefunden, wird die gefundene Position zurückgegeben.

- Der Text wird von vorn nach hinten nach der Zeichenfolge durchsucht.
- Taucht die Zeichenfolge im Text mehrmals auf, wird nur das erste Auftreten berücksichtigt.
- Der Rückgabewert sollte stets auf 0 überprüft werden, um festzustellen, ob die gesuchte Zeichenfolge auch wirklich gefunden wurde.

Beispiele:

```
Position = TxWhere("Das ist ein Text", "ist")
```

Nach dieser Zeile enthält die Variable [Position] den Wert 5.

Siehe auch:

TLike, TReplace, TComp, TPart

UNCERTAINTY_LOOP

Verfügbar ab: Professional Edition

Schleife zur Bestimmung der Messunsicherheit der Ergebnisse eines Algorithmus mittels Monte-Carlo-Methode (MCM).

Deklaration:

```
UNCERTAINTY_LOOP EwVersuchsanzahl EwInit
```

Parameter:

EwVersuchsanzahl	Anzahl der durchzuführenden Monte-Carlo-Versuche. Der Parameter liegt häufig im Bereich 100..10000.
EwInit	Der Zufallszahlengenerator für die Funktion UncertaintyModify() wird mit diesem Wert (neu) initialisiert. Wenn der Parameter weggelassen wird, wird ein fester interner Wert verwendet.
	0 : Wird gewählt, wenn bei einer erneuten Ausführung der gesamten Schleife mit anderen Zufallswerten gearbeitet werden soll.
	>0 : Ganzzahliger Wert für die Neuinitialisierung des Zufallszahlen-Generators. Sorgt für ein reproduzierbares Verhalten der gesamten Schleife.

Beschreibung:

Die UNCERTAINTY_LOOP ist eine Schleife und dient der Bestimmung der Messunsicherheit der Ergebnisse eines Algorithmus mittels Monte-Carlo Methode. Sie bildet den Rahmen für die Funktionsaufrufe UncertaintyModify() und UncertaintyCalc(). Nach Ablauf der Schleife ist die Messunsicherheit als anwenderdefinierte Eigenschaft am Ergebnis gesetzt.

Die Schleife wird (M+2) mal ausgeführt.

Im ersten Durchlauf wird der Algorithmus mit unbeeinflussten Eingangsdaten durchlaufen. Anschließend wird der Algorithmus M-mal in M Monte-Carlo Versuchen mit verrauschten Eingangsdaten (typischerweise erzeugt mit der Funktion UncertaintyModify) durchlaufen. Dabei wird ein verrauschtes Ergebnis berechnet. Aus diesem Ergebnis wird die Messunsicherheit mit der Funktion UncertaintyCalc ermittelt. Zum Schluss erfolgt ein Durchlauf mit den unbeeinflussten Eingangsdaten zur Wiederherstellung.

Die mit dem Befehl UNCERTAINTY_MODIFY definierte Schleife wird mit dem Befehl END abgeschlossen.

Innerhalb einer UNCERTAINTY_LOOP kann keine weitere UNCERTAINTY_LOOP gestartet werden.

Weitergehende Informationen zur Bestimmung der Messunsicherheit in FAMOS finden Sie im Bedienerhandbuch/Online-Hilfe, Abschnitt "Bestimmung der Messunsicherheit". Hier werden auch die zugrunde liegenden Normen und Verfahren (GUM, GUM, Monte-Carlo Verfahren) näher erläutert.

Der Befehl ist in der FAMOS Standard-Edition nicht verfügbar.

Beispiele:

Aus einem Spannungskanal Input1 mit Messbereich 10V und Messunsicherheit 0.1V soll über einen Algorithmus (Tiefpassfilter, Maximalwert) ein Ergebnis mit Namen Result bestimmt werden. Der Algorithmus ohne Bestimmung der Messunsicherheit:

```
_In1 = Input1
Result = Max(FiltLp(_In1, 0, 0, 1, 0.02))
```

Um die Messunsicherheit für das Ergebnis zu bestimmen, muss der Algorithmus wie folgt erweitert werden:

```
; Die Messunsicherheit der Eingangsdaten bekannt machen:
UncertaintySet(Input1, "Uncertainty", 0.1)
; alle Monte-Carlo Versuche durchführen, hier Anzahl M = 1000
UNCERTAINTY_LOOP 1000
   ; Die Eingangsdaten verrauschen
   _In1 = UncertaintyModify(Input1)
   ; den eigentlichen Algorithmus ausführen und das Ergebnis berechnen
   Result = Max(FiltLp(_In1, 0, 0, 1, 0.02))
   ; die Messunsicherheit des Ergebnisses berechnen
   UncertaintyCalc(Result)
END
; Die Messunsicherheit des Resultats abfragen
uc = UncertaintyGet(Result, "Uncertainty")
```

Siehe auch:

UncertaintyModify, UncertaintyCalc, UncertaintySet, UncertaintySnapshot

UncertaintyCalc

Verfügbar ab: Professional Edition

Ermitteln der Messunsicherheit mittels Monte-Carlo Methode.

Deklaration:

```
UncertaintyCalc ( Variable [, EwÜberdeckungswahrscheinlichkeit] [, Null] [, TxErweitert1] [, TxErweitert2] [, TxErweitert3] [, TxErweitert4] )
```

Parameter:

Variable	Die Variable mit dem Ergebnis eines Algorithmus. Zu dieser Variable soll die Messunsicherheit ermittelt werden. Die Variable muss direkt angegeben werden, temporäre oder indizierte Daten sind nicht zulässig.
EwÜberdeckungswahrscheinlichkeit	Überdeckungswahrscheinlichkeit (Confidency Level) in Prozent. Wenn 0 oder nicht angegeben, wird die erweiterte Messunsicherheit nicht bestimmt. Wenn ungleich 0, wird aus diesem Wert das Überdeckungsintervall bestimmt und daraus die erweiterte Messunsicherheit. Typische Werte sind 95 oder 99. Der wahre Wert liegt um das unbeeinflusste Ergebnis herum mit der angegebenen Wahrscheinlichkeit in dem Bereich, der durch die erweiterte Messunsicherheit angegeben wird. (optional)
Null	Reserviert, auf 0 zu setzen. (optional)
TxErweitert1	1. erweitertes Analyseergebnis (optional)
	"uc" : Für jeden Messpunkt des Ergebnisses wird die Messunsicherheit bestimmt.
	"mean" : Für jeden Messpunkt des Ergebnisses wird der Mittelwert aus allen Monte-Carlo-Versuchen bestimmt.
	"min/max" : Für jeden Messpunkt des Ergebnisses wird Minimum und Maximum aus allen Monte-Carlo-Versuchen bestimmt.
	"pdf0" : Verteilungsdichte der Abweichungen vom unbeeinflussten Ergebnis, um ein scheinbares Ergebnis von 0.0 herum zentriert.
	"pdf" : Verteilungsdichte der Abweichungen vom unbeeinflussten Ergebnis.
TxErweitert2	2. erweitertes Analyseergebnis (optional)
	"uc" : Für jeden Messpunkt des Ergebnisses wird die Messunsicherheit bestimmt.
	"mean" : Für jeden Messpunkt des Ergebnisses wird der Mittelwert aus allen Monte-Carlo-Versuchen bestimmt.
	"min/max" : Einhüllende, für jeden Messpunkt des Ergebnisses wird Minimum und Maximum aus allen Monte-Carlo-Versuchen bestimmt.
	"pdf0" : Verteilungsdichte der Abweichungen vom unbeeinflussten Ergebnis, um ein scheinbares Ergebnis von 0.0 herum zentriert.
	"pdf" : Verteilungsdichte (nur für Einzelwerte) der Abweichungen vom unbeeinflussten Ergebnis.
TxErweitert3	3. erweitertes Analyseergebnis (optional)
	"uc" : Für jeden Messpunkt des Ergebnisses wird die Messunsicherheit bestimmt.
	"mean" : Für jeden Messpunkt des Ergebnisses wird der Mittelwert aus allen Monte-Carlo-Versuchen bestimmt.
	"min/max" : Für jeden Messpunkt des Ergebnisses wird Minimum und Maximum aus allen Monte-Carlo-Versuchen bestimmt.
	"pdf0" : Verteilungsdichte der Abweichungen vom unbeeinflussten Ergebnis, um ein scheinbares Ergebnis von 0.0 herum zentriert.
	"pdf" : Verteilungsdichte der Abweichungen vom unbeeinflussten Ergebnis.
TxErweitert4	4. erweitertes Analyseergebnis (optional)
	"uc" : Für jeden Messpunkt des Ergebnisses wird die Messunsicherheit bestimmt.
	"mean" : Für jeden Messpunkt des Ergebnisses wird der Mittelwert aus allen Monte-Carlo-Versuchen bestimmt.
	"min/max" : Für jeden Messpunkt des Ergebnisses wird Minimum und Maximum aus allen Monte-Carlo-Versuchen bestimmt.

	"pdf0" : Verteilungsdichte der Abweichungen vom unbeeinflussten Ergebnis, um ein scheinbares Ergebnis von 0.0 herum zentriert.
	"pdf" : Verteilungsdichte der Abweichungen vom unbeeinflussten Ergebnis.

Beschreibung:

Die Prozedur wird üblicherweise für die Ergebnisvariable eines Algorithmus gegen Ende einer UNCERTAINTY_LOOP aufgerufen. Das Ergebnis eines Monte-Carlo Versuchs wird übergeben, um damit die Schätzung der Messunsicherheit zu verbessern.

Messunsicherheit und erweiterte Messunsicherheit (sofern bestimmt) werden als anwenderdefinierte Eigenschaften an die übergebene Ergebnisvariable angehängt. Sie können mit der Funktion UncertaintyGet() abgefragt werden.

Mit den optionalen Parametern 4 bis 7 können zusätzlich erweiterte Analyseergebnisse angefordert werden. Bei der erweiterten Analyse werden alle bislang durchgeführten Monte-Carlo Versuche berücksichtigt. Die Reihenfolge der zusätzlich spezifizierten Analysen ist egal, "pdf" und "pdf0" können nicht gleichzeitig verwendet werden. Die Ergebnisse der erweiterten Analyse werden in eine Gruppe geschrieben, diese wird nur angelegt, wenn mindestens eine erweiterte Analyse gewünscht ist. Die Gruppen-Variable wird als Seiteneffekt dieser Prozedur erzeugt, initialisiert und aktualisiert.

Der Name der erzeugten Gruppenvariablen wird aus dem Namen der Ergebnisvariable durch Anängen von "_uc_result" gebildet und enthält, sofern die entsprechende Option angegeben ist, folgende Kanäle:

original:	Das unbeinflusste Ergebnis aus dem ersten Durchlauf der UNCERTAINTY_LOOP-Schleife (immer enthalten)
uc:	Messunsicherheit entsprechend Option "uc"
min:	Untere Einhüllende entsprechend Option "min/max"
max:	Obere Einhüllende entsprechend Option "min/max"
mean:	Mittelwert entsprechend Option "mean"
pdf0:	Null-basierte Verteilungsdichte der Abweichungen entsprechend Option "pdf0"
pdf:	Verteilungsdichte für Einzelwerte entsprechend Option "pdf"

Beispiel: Der Aufruf

```
UncertaintyCalc(x, 99, 0, "uc", "pdf0", "min/max")
```

erzeugt die Gruppe "x_uc_result" mit der folgenden Struktur:

```
x_uc_result
   |_ original
   |_ uc
   |_ min
   |_ mac
   |_ pdf0
```

Wenn die Ergebnisvariable selbst ein Kanal einer Gruppe ist, erfolgt die Namensbildung für die Gruppe mit den erweiterten Ergebnissen nach dem Muster "[GruppenName]_[KanalName]_uc_result". Eine eventuelle Messungszugehörigkeit wird übernommen.

Die Prozedur kann nur innerhalb der UNCERTAINTY_LOOP benutzt werden. Beim ersten Durchlauf initialisiert sie die Gruppe und anwenderdefinierten Eigenschaften. Im letzten Durchlauf werden die Ergebnisse ggf. noch finalisiert.

Die Prozedur darf für jedes Ergebnis genau einmal pro Schleifendurchlauf aufgerufen werden. Ausnahme ist der allererste Durchlauf, bei dem der Prozeduraufruf samt Parameter in Betrieb genommen wird.

Die Berechnung der Messunsicherheit wird für folgende Datentypen unterstützt:

- Äquidistante Daten
- y-Komponente von xy-Daten
- Betrag bei komplexen Daten

Liegen andere Datentypen vor, muss zunächst eine Zuweisung an eine namesgebende Hilfsvariable eingefügt werden, etwa A_P = A.P und danach UncertaintyCalc(A_P).

Nach Ablauf der UNCERTAINTY_LOOP löscht der Anwender die Gruppe der erweiterten Analyse bei Bedarf.

Weitergehende Informationen zur Bestimmung der Messunsicherheit in FAMOS finden Sie im Bedienerhandbuch/Online-Hilfe, Abschnitt "Bestimmung der Messunsicherheit". Hier werden auch die zugrunde liegenden Normen und Verfahren (GUM, Monte-Carlo Verfahren) näher erläutert.

Die Funktion ist in der FAMOS Standard-Edition nicht verfügbar.

Beispiele:

Aus einem Spannungskanal Input1 mit Messbereich 10V und Messunsicherheit 0.1V soll über einen Algorithmus (Tiefpassfilter, Maximalwert) ein Ergebnis mit Namen Result bestimmt werden. Außerdem soll die Verteilungsdichte bestimmt werden.

```
; Die Messunsicherheit der Eingangsdaten bekannt machen:
UncertaintySet(Input1, "Uncertainty", 0.1)
; alle Monte-Carlo Versuche durchführen, hier Anzahl M = 1000
UNCERTAINTY_LOOP 1000
   ; Die Eingangsdaten verrauschen
```

```
   _In1 = UncertaintyModify(Input1)
   ; den eigentlichen Algorithmus ausführen und das Ergebnis berechnen
   Result = Max(FiltLp(_In1, 0, 0, 1, 0.02))
   ; die Messunsicherheit und Erweiterte Messunsicherheit des Ergebnisses berechnen
   UncertaintyCalc(Result, 95 , 0 , "pdf")
   ; erzeugt auch die Gruppe "Result_uc_result" mit den Kanälen "original" und "pdf"!
END
; Die Messunsicherheiten des Resultats abfragen
uc = UncertaintyGet(Result, "Uncertainty")
U = UncertaintyGet(Result, "Expanded uncertainty")
```

Siehe auch:

UNCERTAINTY_LOOP, UncertaintyModify, UncertaintySet, UncertaintyGet, UncertaintySnapshot

UncertaintyGet

Verfügbar ab: Professional Edition

Abfragen einer anwenderdefinierten Eigenschaft zum Thema Messunsicherheit.

Deklaration:

```
UncertaintyGet ( Variable, TxPropName ) -> EwWert
```

Parameter:

Variable	Abzufragende Variable
TxPropName	Name der Eigenschaft
	"Uncertainty" : Standardmessunsicherheit. Angegeben in der y-Einheit des Datensatzes. Wird von der Funktion UncertaintyCalc() berechnet. Die Funktion UncertaintyModify() addiert eine normalverteile Einflussgröße.
	"Expanded uncertainty" : Erweiterte Messunsicherheit. Angegeben in der y-Einheit des Datensatzes. Wird von der Funktion UncertaintyCalc() berechnet. Die Funktion UncertaintyModify() addiert eine normalverteile Einflussgröße, wenn außerdem Überdeckungswahrscheinlichkeit oder Erweiterungsfaktor gegeben sind.
	"Coverage probability" : Die Überdeckungswahrscheinlichkeit, angegeben in Prozent. Z.B. 95 oder 99.7. Mit dieser Wahrscheinlichkeit liegt der wahre Wert im Überdeckungsintervall (Vertrauensbereich). Wird von UncertaintyCalc() gesetzt. Die Funktion UncertaintyModify() beachtet den Wert in Zusammenhang mit der erweiterten Messunsicherheit.
	"Coverage factor" : Verhältnis von erweiterter Messunsicherheit zu Standardmessunsicherheit, typisch im Bereich 2 bis 3. Die Funktion UncertaintyModify() beachtet den Wert in Zusammenhang mit der erweiterten Messunsicherheit.
	"Uncertainty Source.Rectangular" : UncertaintyModify() addiert eine symmetrische Gleichverteilung. Die halbe Breite der Verteilung ist angegeben.
	"Uncertainty Source.LSBs" : Anzahl von LSBs, um die das Signal klappert. Nur für ganzzahlige Daten. UncertaintyModify() addiert ein Bitklappern der in LSBs angegebenen Standardabweichung.
	"Uncertainty Source.Amplitude" : Verstärkungsabweichung um den Wert 1.0 herum. UncertaintyModify() addiert eine zufällige, aber für die Messung feste Verstärkungsabweichung. Z.B. 0.1, falls die Verstärkung um 10% zu variieren ist, zu deuten als Standardabweichung.
	"Uncertainty Source.Offset" : Gleichbleibende Offsetabweichung in physikalischen Einheiten des Kanals. UncertaintyModify() addiert eine zufällige, aber für die Messung feste Offsetabweichung. Die Standardabweichung ist angegeben.
	"Uncertainty Source.Drift Offset" : Zeitlich schwankende Drift. Höhe der Drift-Offsetfehler angegeben als Standardabweichung (in physikalischen Einheiten des Kanals). UncertaintyModify() addiert eine Offsetdrift.
	"Uncertainty Source.Drift Time" : Zeitlich schwankende Drift. Wenn UncertaintyModify() eine Offsetdrift addiert, wird hiermit ein zeitliches Maß für die Änderung festgelegt.
	"Uncertainty Source.Gain Drift" : Zeitlich driftende Verstärkungsabweichung angegeben als Faktor um den Wert 1.0 herum. Z.B. 0.01, falls die Verstärkung um 1% zu variieren ist, zu deuten als Standardabweichung. UncertaintyModify() addiert diese Drift.
	"Uncertainty Source.Gain Drift Time" : Wenn UncertaintyModify() eine zeitlich driftende Verstärkungsabweichung addiert, wird hiermit ein zeitliches Maß für die Änderung festgelegt
	"Uncertainty Source.Hum Amplitude" : UncertaintyModify() addiert ein Netzbrummen. Die Amplitude ist zufällig gewählt und für eine Messung fest. Der angegebene Wert ist die Standardabweichung der gewählten Amplituden.
	"Uncertainty Source.Hum Frequency" : Wenn UncertaintyModify() ein Netzbrummen addiert, wird hiermit die feste Frequenz angegeben.
	"Uncertainty Source.Hum Harmonics" : Wenn UncertaintyModify() ein Netzbrummen addiert, wird hiermit das Verhältnis der Leistung der Oberschwingungen zur Grundschwingung angegeben, z.B. 0.01, falls das Verhältnis 1% beträgt.
	"Uncertainty Source.Spikes Max" : Spikes, UncertaintyModify() addiert Störpulse mit dem angegebenen Maximalwert (in y-Einheiten des Datensatzes).
	"Uncertainty Source.Spikes Min" : Spikes, UncertaintyModify() addiert Störpulse mit dem angegebenen Minimalwert.
	"Uncertainty Source.Spikes Width" : Wenn UncertaintyModify() Störpulse addiert, dann wird hiermit die maximale Breite der Störpulse festgelegt.
	"Uncertainty Source.Spikes Time" : Wenn UncertaintyModify() Störpulse addiert, dann wird hiermit festgelegt, nach welcher Zeit spätestens ein neuer Störpuls auftritt.
	"Uncertainty Source.Noise RMS" : UncertaintyModify() addiert Rauschen, das vor allen höhere Frequenzanteile enthält. Die Standardabweichung ist gegeben.
	"Uncertainty Source.Noise Frequency" : Wenn UncertaintyModify() ein Rauschen addiert, wird hiermit die untere Grenzfrequenz angegeben.

	"Uncertainty Source.Temp Off" : Proportional zur Temperatur steigender Offset angegeben als Standardabweichung pro Kelvin (Einheit des beeinflussenden Temperaturkanals). UncertaintyModify() addiert eine solche Offsetabweichung mit einem zufälligen aber für die Messung festen Proportionalitätsfaktor, falls ein Temperaturkanal angegeben ist. Z.B. 0.1, falls die Abweichung 0.1 Einheiten pro Kelvin beträgt. Die addierte Abweichung ist proportional zur Abweichung des Temperaturkanals von der Referenztemperatur. Die Referenztemperatur muss spezifiziert werden.
	"Uncertainty Source.Temp Gain" : Proportional zur Temperatur steigende Verstärkungsabweichung pro Kelvin (Einheit des beeinflussenden Temperaturkanals) um den Wert 1.0 herum. UncertaintyModify() addiert eine solche Verstärkungsabweichung mit einem zufälligen, aber für die Messung festen Proportionalitätsfaktor. Z.B. 0.01, falls die Verstärkungsabweichung 1% pro Kelvin beträgt, zu deuten als Standardabweichung. Die addierte Abweichung ist proportional zum Messwert und zur Abweichung des Temperaturkanals von der Referenztemperatur. Die Referenztemperatur muss spezifiziert werden.
	"Uncertainty Source.Temp Ref" : Für die temperaturabhängigen Beeinflussungen die in UncertaintyModify() benutzte Referenztemperatur in der Einheit des beeinflussenden Temperaturkanals. Falls nicht angegeben, wird 20 angenommen. Falls der Temperaturkanal in °C gegeben ist und 23 als Wert angegeben wird, so wird das als 23°C gedeutet.
EwWert	Wert der Eigenschaft (Einzelwert)

Beschreibung:

Die hier definierten Eigenschaften bilden direkte Kenngrößen zum Thema Messunsicherheit ab, die für Eingangsgrößen durch den Anwender oder das Messsystem definiert werden und für Ausgangsgrößen durch die Funktion UncertaintyCalc berechnet werden - oder beschreiben Einflussgrößen, die Messunsicherheiten verursachen ("Uncertainty Source.*") und von der Funktion UncertaintyModify() berücksichtigt werden.

Es gibt eine weitere, hier nicht aufgeführte, Kenngröße zum Thema Messunsicherheit: "Uncertainty evaluation". Diese enthält die Methode, nach der die Messunsicherheit ermittelt wird, wird von der Funktion UncertaintyCalc() gesetzt und ist vom Datentyp [Text], z.B. "MCM, M=100". Zum Abfragen dieser Eigenschaft verwenden Sie die Funktion UserPropText?(.., "Uncertainty evaluation").

Weitergehende Informationen zur Bestimmung der Messunsicherheit in FAMOS finden Sie im Bedienerhandbuch/Online-Hilfe, Abschnitt "Bestimmung der Messunsicherheit". Dort werden auch die hier definierten Kenngrößen und die zugrunde liegenden Normen und Verfahren (GUM, Monte-Carlo Verfahren) näher erläutert.

Die Funktion ist in der FAMOS Standard-Edition nicht verfügbar.

Beispiele:

Aus einem Spannungskanal Input1 mit Messbereich 10V und Messunsicherheit 0.1V soll über einen Algorithmus (Tiefpassfilter, Maximalwert) ein Ergebnis mit Namen Result bestimmt werden.

```
; Die Messunsicherheit der Eingangsdaten bekannt machen:
UncertaintySet(Input1, "Uncertainty", 0.1)
; alle Monte-Carlo Versuche durchführen, hier Anzahl M = 1000
UNCERTAINTY_LOOP 1000
   ; Die Eingangsdaten verrauschen
   _In1 = UncertaintyModify(Input1)
   ; den eigentlichen Algorithmus ausführen und das Ergebnis berechnen
   Result = Max(FiltLp(_In1, 0, 0, 1, 0.02))
   ; die Messunsicherheit und Erweiterte Messunsicherheit des Ergebnisses berechnen
   UncertaintyCalc(Result, 95)
END
; Die Messunsicherheiten des Resultats abfragen
uc = UncertaintyGet(Result, "Uncertainty")
U = UncertaintyGet(Result, "Expanded uncertainty")
```

Definition einer Offsetdrift:

```
UncertaintySet(Input1, "Uncertainty Source.Drift Offset", 0.01) ; [mV]
UncertaintySet(Input1, "Uncertainty Source.Drift Time", 1000) ; [s]
UNCERTAINTY_LOOP 1000
...
```

Siehe auch:

UNCERTAINTY_LOOP, UncertaintySet, UncertaintyModify, UncertaintyCalc, UncertaintySnapshot

UncertaintyModify

Verfügbar ab: Professional Edition

Verrauschen eines Signals im Rahmen der Bestimmung der Messunsicherheit nach Monte-Carlo Verfahren.

Deklaration:

```
UncertaintyModify ( Variable [, Temperatur] ) -> Ergebnis
```

Parameter:

Variable	Zu verrauschendes Signal. Die Variable muss direkt angegeben werden, temporäre oder indizierte Daten sind nicht zulässig.
Temperatur	Optionaler Temperaturkanal, falls temperaturabhängige Abweichungen zu berücksichtigen sind. (optional)
Ergebnis	Modifiziertes Signal

Beschreibung:

Ein Signal wird verrauscht. Wenn der Eingangskanal anwenderdefinierte Eigenschaften aufweist, aus denen sich eine Störung berechnen lässt, wird diese Berechnung durchgeführt. Solche Eigenschaften sind beispielsweise "Uncertainty" oder "Uncertainty Source.Offset" und werden mit der Funktion UncertaintySet() definiert.

Weist das Eingangssignal z.B. die Eigenschaft "Uncertainty" auf, so wird eine normalverteilte Beeinflussung addiert.

Zurückgegeben wird das Signal, das mit entsprechenden Störungen beaufschlagt ist. Die Funktion addiert Rauschen und andere Störungen.

Die Funktion kann nur innerhalb der UNCERTAINTY_LOOP benutzt werden. Beim ersten und letzten Durchlauf liefert die Funktion ohne Prüfung das unveränderte Eingangssignal zurück. In den dazwischen liegenden Schleifendurchläufen wird dem Monte-Carlo Verfahren folgend das Eingangssignal mit einer zufälligen Störung beaufschlagt.

Der benutzte Zufallszahlengenerator wird vom Schleifenrahmen UNCERTAINTY_LOOP initialisiert und läuft unabhängig vom Generator, den die random() Funktion benutzt.

Die Funktion darf auch aufgerufen werden für Signale, die (aktuell) keine passenden anwenderdefinierten Eigenschaften aufweisen. Die Funktion liefert dann einfach nur eine Kopie.

Die Funktion beeinflusst nur die y-Koordinate von äquidistanten und XY-Daten, außerdem den Betrag von komplexen Daten. Liegen andere Datentypen vor, so ist ein Aufbrechen in einzelne Teilvariablen nötig. Soll z.B. die Phase eines komplexen Datensatzes verrauscht werden, muss eine separate Variable mit der Phase vor Beginn der UNCERTAINTY_LOOP eingeführt werden, z.B. A_P = A.P. Mit UncertaintySet() erhält sie passende Vorgaben.

Der optionale Temperaturkanal muss dieselbe oder eine um ein ganzes Vielfaches größere Abtastzeit haben wie das zu verrauschende Signal. Ist die zeitliche Ausdehnung des Temperaturkanals größer, wird dieser entsprechend eingekürzt. Ist die zeitliche Ausdehnung des Temperaturkanals kleiner, wird dieser mit seinem letzten Wert entsprechend verlängert.

Weitergehende Informationen zur Bestimmung der Messunsicherheit in FAMOS finden Sie im Bedienerhandbuch/Online-Hilfe, Abschnitt "Bestimmung der Messunsicherheit". Hier werden auch die zugrunde liegenden Normen und Verfahren (GUM, Monte-Carlo Verfahren) näher erläutert.

Die Funktion ist in der FAMOS Standard-Edition nicht verfügbar.

Beispiele:

Aus einem Spannungskanal Input1 mit Messbereich 10V und Messunsicherheit 0.1V soll über einen Algorithmus (Tiefpassfilter, Maximalwert) ein Ergebnis mit Namen Result bestimmt werden. Außerdem soll die Verteilungsdichte bestimmt werden.

```
; Die Messunsicherheit der Eingangsdaten bekannt machen:
UncertaintySet(Input1, "Uncertainty", 0.1)
; alle Monte-Carlo Versuche durchführen, hier Anzahl M = 1000
UNCERTAINTY_LOOP 1000
   ; Die Eingangsdaten verrauschen
   _In1 = UncertaintyModify(Input1)
   ; den eigentlichen Algorithmus ausführen und das Ergebnis berechnen
   Result = Max(FiltLp(_In1, 0, 0, 1, 0.02))
   ; die Messunsicherheit und Erweiterte Messunsicherheit des Ergebnisses berechnen
   UncertaintyCalc(Result, 95 , 0 , "pdf")
   ; erzeugt auch die Gruppe "Result_uc_result" mit den Kanälen "original" und "pdf"!
END
; Die Messunsicherheiten des Resultats abfragen
uc = UncertaintyGet(Result, "Uncertainty")
U = UncertaintyGet(Result, "Expanded uncertainty")
```

Aus einem Spannungskanal U1 mit Messbereich 10V und Messunsicherheit 0.1V und einem Stromkanal I1 (2A, 0.01A) soll die Leistung bestimmt werden. Der Spannungskanal ist zusätzlich von Netzbrummen betroffen.

```
UncertaintySet(U1, "Uncertainty", 0.1)
UncertaintySet(U1, "Uncertainty Source.Hum Amplitude", 0.01)
UncertaintySet(I1, "Uncertainty", 0.01)
UNCERTAINTY_LOOP 1000
```

```
   U1_mod = UncertaintyModify(U1)
   I1_mod = UncertaintyModify(I1)
   P = Mean(U1_mod * I1_mod)
   UncertaintyCalc(P)
END
uc = UncertaintyGet(W, "Uncertainty")
```

Gegeben ist ein Spannungskanal 'Channel' in V und ein Temperaturkanal 'Temperature' in °C skaliert. Im Datenblatt des Sensors steht eine temperaturabhängige Nullpunktabweichung von 0.001V/K und eine temperaturabhängige Verstärkungsabweichung von 50ppm/K, ausgehend vom Referenzwert 23°C.

```
UncertaintySet(Channel, "Uncertainty Source.Temp Off", 0.001)
UncertaintySet(Channel, "Uncertainty Source.Temp Gain", 0.000050)
UncertaintySet(Channel, "Uncertainty Source.Temp Ref", 23)
UNCERTAINTY_LOOP 100
   _Channel = UncertaintyModify(Channel, Temperature)
   ...
```

Siehe auch:

UNCERTAINTY_LOOP, UncertaintyCalc, UncertaintySet, UncertaintyGet, UncertaintySnapshot

UncertaintySet

Verfügbar ab: Professional Edition

Setzen einer anwenderdefinierten Eigenschaft zum Thema Messunsicherheit.

Deklaration:

```
UncertaintySet ( Variable, TxPropName, EwWert )
```

Parameter:

Variable	Zu ändernde Variable
TxPropName	Name der zu ändernden Eigenschaft
	"Uncertainty" : Standardmessunsicherheit. Angegeben in der y-Einheit des Datensatzes. Wird von der Funktion UncertaintyCalc() berechnet. Die Funktion UncertaintyModify() addiert eine normalverteile Einflussgröße, also ein weißes Rauschen mit Standardabweichung bzw. Effektivwert gleich Messunsicherheit.
	"Expanded uncertainty" : Erweiterte Messunsicherheit. Angegeben in der y-Einheit des Datensatzes. Wird von der Funktion UncertaintyCalc() berechnet. Die Funktion UncertaintyModify() addiert eine normalverteile Einflussgröße (Rauschen), wenn außerdem Überdeckungswahrscheinlichkeit oder Erweiterungsfaktor gegeben sind.
	"Uncertainty evaluation" : Die Methode, nach der die Messunsicherheit bzw. auch die erweiterte Messunsicherheit ermittelt werden. Z.B. "MCM, M=100". Wird von UncertaintyCalc() gesetzt. (Datentyp: Text)
	"Coverage probability" : Die Überdeckungswahrscheinlichkeit, angegeben in Prozent. Z.B. 95 oder 99.7. Mit dieser Wahrscheinlichkeit liegt der wahre Wert im Überdeckungsintervall (Vertrauensbereich). Wird von UncertaintyCalc() gesetzt. Die Funktion UncertaintyModify() beachtet den Wert in Zusammenhang mit der erweiterten Messunsicherheit.
	"Coverage factor" : Verhältnis von erweiterter Messunsicherheit zu Standardmessunsicherheit, typisch im Bereich 2 bis 3. Die Funktion UncertaintyModify() beachtet den Wert in Zusammenhang mit der erweiterten Messunsicherheit.
	"Uncertainty Source.Rectangular" : UncertaintyModify() addiert eine symmetrische Gleichverteilung. Die halbe Breite der Verteilung ist angegeben.
	"Uncertainty Source.LSBs" : Anzahl von LSBs, um die das Signal klappert. Nur für ganzzahlige Daten. UncertaintyModify() addiert ein Bitklappern der in LSBs angegebenen Standardabweichung.
	"Uncertainty Source.Amplitude" : Verstärkungsabweichung um den Wert 1.0 herum. UncertaintyModify() addiert eine zufällige, aber für die Messung feste Verstärkungsabweichung. Z.B. 0.1, falls die Verstärkung um 10% zu variieren ist, zu deuten als Standardabweichung.
	"Uncertainty Source.Offset" : Gleichbleibende Offsetabweichung in physikalischen Einheiten des Kanals. UncertaintyModify() addiert eine zufällige, aber für die Messung feste Offsetabweichung. Die Standardabweichung ist angegeben.
	"Uncertainty Source.Drift Offset" : Zeitlich schwankende Drift. Höhe der Drift-Offsetabweichung angegeben als Standardabweichung (in physikalischen Einheiten des Kanals). UncertaintyModify() addiert eine Offsetdrift.
	"Uncertainty Source.Drift Time" : Zeitlich schwankende Drift. Wenn UncertaintyModify() eine Offsetdrift addiert, wird hiermit ein zeitliches Maß für die Änderung festgelegt.
	"Uncertainty Source.Gain Drift" : Zeitlich driftende Verstärkungsabweichung angegeben als Faktor um den Wert 1.0 herum. Z.B. 0.01, falls die Verstärkung um 1% zu variieren ist, zu deuten als Standardabweichung. UncertaintyModify() addiert diese Drift.
	"Uncertainty Source.Gain Drift Time" : Wenn UncertaintyModify() eine zeitlich driftende Verstärkungsabweichung addiert, wird hiermit ein zeitliches Maß für die Änderung festgelegt
	"Uncertainty Source.Hum Amplitude" : UncertaintyModify() addiert ein Netzbrummen. Die Amplitude ist zufällig gewählt und für eine Messung fest. Der angegebene Wert ist die Standardabweichung der gewählten Amplituden.
	"Uncertainty Source.Hum Frequency" : Wenn UncertaintyModify() ein Netzbrummen addiert, wird hiermit die feste Frequenz angegeben.
	"Uncertainty Source.Hum Harmonics" : Wenn UncertaintyModify() ein Netzbrummen addiert, wird hiermit das Verhältnis der Leistung der Oberschwingungen zur Grundschwingung angegeben, z.B. 0.01, falls das Verhältnis 1% beträgt.
	"Uncertainty Source.Spikes Max" : Spikes, UncertaintyModify() addiert Störpulse mit dem angegebenen Maximalwert (in y-Einheiten des Datensatzes).
	"Uncertainty Source.Spikes Min" : Spikes, UncertaintyModify() addiert Störpulse mit dem angegebenen Minimalwert.
	"Uncertainty Source.Spikes Width" : Wenn UncertaintyModify() Störpulse addiert, dann wird hiermit die maximale Breite der Störpulse festgelegt.
	"Uncertainty Source.Spikes Time" : Wenn UncertaintyModify() Störpulse addiert, dann wird hiermit festgelegt, nach welcher Zeit spätestens ein neuer Störpuls auftritt.

	"Uncertainty Source.Noise RMS" : UncertaintyModify() addiert Rauschen, das vor allem höhere Frequenzanteile enthält. Die Standardabweichung ist gegeben. Ist gewöhnliches (weißes) Rauschen gewünscht, muss die Eigenschaft "Uncertainty" benutzt werden.
	"Uncertainty Source.Noise Frequency" : Wenn UncertaintyModify() ein Rauschen addiert, wird hiermit die untere Grenzfrequenz (größer 0)angegeben. Ist diese Eigenschaft nicht angegeben, wird ein Wert von 20% der Abtastfrequenz angenommen.
	"Uncertainty Source.Temp Off" : Proportional zur Temperatur steigender Offset angegeben als Standardabweichung pro Kelvin (Einheit des beeinflussenden Temperaturkanals). UncertaintyModify() addiert eine solche Offsetabweichung mit einem zufälligen aber für die Messung festen Proportionalitätsfaktor, falls ein Temperaturkanal angegeben ist. Z.B. 0.1, falls die Abweichung 0.1 Einheiten pro Kelvin beträgt. Die addierte Abweichung ist proportional zur Abweichung des Temperaturkanals von der Referenztemperatur. Die Referenztemperatur muss spezifiziert werden.
	"Uncertainty Source.Temp Gain" : Proportional zur Temperatur steigende Verstärkungsabweichung pro Kelvin (Einheit des beeinflussenden Temperaturkanals) um den Wert 1.0 herum. UncertaintyModify() addiert eine solche Verstärkungsabweichung mit einem zufälligen, aber für die Messung festen Proportionalitätsfaktor. Z.B. 0.01, falls die Verstärkungsabweichung 1% pro Kelvin beträgt, zu deuten als Standardabweichung. Die addierte Abweichung ist proportional zum Messwert und zur Abweichung des Temperaturkanals von der Referenztemperatur. Die Referenztemperatur muss spezifiziert werden.
	"Uncertainty Source.Temp Ref" : Für die temperaturabhängigen Beeinflussungen die in UncertaintyModify() benutzte Referenztemperatur in der Einheit des beeinflussenden Temperaturkanals. Falls nicht angegeben, wird 20 angenommen. Falls der Temperaturkanal in °C gegeben ist und 23 als Wert angegeben wird, so wird das als 23°C gedeutet.
	"*" : Löschen aller Eigenschaften zum Thema Messunsicherheit. Für den 2. Parameter muss ein leerer Text angegeben werden.
EwWert	Neuer Wert, auf den die Eigenschaft gesetzt werden soll. Für "Uncertainty evaluation" und "*" Datentyp Text, sonst eine Zahl. Wenn ein leerer Text angegeben wird, wird die Eigenschaft gelöscht.

Beschreibung:

Die hier definierten Eigenschaften bilden direkte Kenngrößen zum Thema Messunsicherheit ab, die für Eingangsgrößen durch den Anwender oder das Messsystem definiert werden und für Ausgangsgrößen durch die Funktion UncertaintyCalc berechnet werden - oder beschreiben Einflussgrößen, die Messunsicherheiten verursachen ("Uncertainty Source.*") und von der Funktion UncertaintyModify() berücksichtigt werden.

Weitergehende Informationen zur Bestimmung der Messunsicherheit in FAMOS finden Sie im Bedienerhandbuch/Online-Hilfe, Abschnitt "Bestimmung der Messunsicherheit". Dort werden auch die hier definierten Kenngrößen und die zugrunde liegenden Normen und Verfahren (GUM, Monte-Carlo Verfahren) näher erläutert.

Der Aufruf UncertaintySet("*", "") löscht alle Eigenschaften zum Thema Messunsicherheit.

Die Funktion ist in der FAMOS Standard-Edition nicht verfügbar.

Beispiele:

Aus einem Spannungskanal Input1 mit Messbereich 10V und Messunsicherheit 0.1V soll über einen Algorithmus (Tiefpassfilter, Maximalwert) ein Ergebnis mit Namen Result bestimmt werden.

```
; Die Messunsicherheit der Eingangsdaten bekannt machen:
UncertaintySet(Input1, "Uncertainty", 0.1)
; alle Monte-Carlo Versuche durchführen, hier Anzahl M = 1000
UNCERTAINTY_LOOP 1000
   ; Die Eingangsdaten verrauschen
   _In1 = UncertaintyModify(Input1)
   ; den eigentlichen Algorithmus ausführen und das Ergebnis berechnen
   Result = Max(FiltLp(_In1, 0, 0, 1, 0.02))
   ; die Messunsicherheit des Ergebnisses berechnen
   UncertaintyCalc(Result)
END
; Die Messunsicherheit des Resultats abfragen
uc = UncertaintyGet(Result, "Uncertainty")
```

Definition einer Offsetdrift:

```
UncertaintySet(Input1, "Uncertainty Source.Drift Offset", 0.01) ; [mV]
UncertaintySet(Input1, "Uncertainty Source.Drift Time", 1000) ; [s]
UNCERTAINTY_LOOP 1000
...
```

Siehe auch:

UNCERTAINTY_LOOP, UncertaintyGet, UncertaintyModify, UncertaintyCalc, UncertaintySnapshot

UncertaintySnapshot

Verfügbar ab: Professional Edition

Sammlung der innerhalb einer UNCERTAINTY_LOOP durch die Monte-Carlo Methode erzeugten Versuchs-Variationen einer Variablen.

Deklaration:

```
UncertaintySnapshot ( Variable [, Option] )
```

Parameter:

Variable	Variable, dessen Variationen eingesammelt werden sollen. Beispielsweise ein verrauschter Eingangskanal, der von UncertaintyModify() zurückgegeben wird, ein Zwischenergebnis im Algorithmus oder auch ein Endergebnis, welches schließlich an UncertaintyCalc() übergeben wird. Die Variable muss direkt angegeben werden, temporäre oder indizierte Daten sind nicht zulässig.
Option	Art des Ergebnisses (optional)
	0 : Für jeden Monte-Carlo Versuch wird eine Kopie des erzeugten Datensatzes an eine Gruppe angehängt. Der Gruppenname wird automatisch vergeben, die Datensätze werden durchnummeriert. Oft verwendet, wenn ein anschließender Vergleich aller Variationen über alle Versuche gewünscht ist.
	1 : Jeder Monte-Carlo Versuch wird als eine Messung gedeutet. Entsprechend dem Messungskonzept von FAMOS werden Kopien der übergebenen Datensätze einem automatisch durchnummerierten Messungsnamen zugeordnet. Oft verwendet, wenn mehrere UncertaintySnapshot()-Aufrufe (z.B. für Eingangsparameter, Zwischenergebnisse und Endergebnis) vorhanden sind und pro Monte-Carlo Versuch verglichen werden sollen (z.B. um die Fortpflanzung des Eingangsfehlers zu begutachten). Diese Methode erlaubt dann die bequeme Visualisierung und den Vergleich über die Messungsansicht der Variablenliste.

Beschreibung:

Die Prozedur kann nur innerhalb einer UNCERTAINTY_LOOP aufgerufen werden und dient zur erweiterten Beurteilung der erfolgten Messunsicherheitsbestimmung. Beispielsweise können alle Ergebnisse der Funktion UncertaintyModify() gesammelt werden und so der Einfluss der Messunsicherheits-Eigenschaften der Eingangsvariablen auf die Generierung des 'verrauschten Signals' kontrolliert werden.

Diese Prozedur erzeugt als Seiteneffekt Variablen, initialisiert, aktualisiert sie, löscht sie sogar. Eine "UNCERTAINTY_LOOP M" Schleife wird (M+2)mal durchlaufen - (Initialisierung, M Monte-Carlo Versuche, Finalisierung). In Initialisierung und Finalisierung tut die Funktion nichts, beim Aufruf innerhalb des ersten Monte-Carlo Versuchs werden alle existierenden Variablen, die auf die Spezifikation der Zielnamen passen, gelöscht und die erste Kopie (entsprechend der gewählten Art des Ergebnisses) angelegt.

Zu beachten ist, dass die Funktion i.Allg. eine sehr große Zahl von Datensätzen anlegt (entsprechend der bei UNCERTAINTY_LOOP angegebenen Versuchszahl) und somit einen großen Einfluss auf Ausführungszeit und Speicherplatzbedarf der Sequenz hat. Nach Erstellung und Inbetriebname einer Sequenz sollten nicht mehr benötigte Aufrufe daher vor dem produktiven Einsatz der Sequenz entfernt werden.

Weitergehende Informationen zur Bestimmung der Messunsicherheit in FAMOS finden Sie im Bedienerhandbuch/Online-Hilfe, Abschnitt "Bestimmung der Messunsicherheit". Hier werden auch die zugrunde liegenden Normen und Verfahren (GUM, Monte-Carlo Verfahren) näher erläutert.

Die Funktion ist in der FAMOS Standard-Edition nicht verfügbar.

Prinzip der Namensgestaltung bei Option = 0 (Gruppe erzeugen)

Der Name der erzeugten Gruppe wird aus dem Originalnamen durch Anhängen von "_uc_shots" gebildet. Die erzeugten Kanäle in der Gruppe erhalten den Namen "L[Laufende Nummer der Schleife]", wobei die laufende Schleifennummer entsprechend der Breite der letzten Schleifennummer mit führenden Nullen aufgefüllt wird.

Beispiel: Der folgende Code erzeugt eine Gruppe "x_uc_shots" mit den Kanälen "L001", "L002" ... "L100"

```
UNCERTAINTY_LOOP 100
   x = UncertaintyModify(input)
   UncertaintySnapshot(x, 0)
   ...
END
```

Falls der Parameter ein Kanal einer Gruppe ist, erfolgt die Namensbildung nach dem Muster "[Gruppenname]_[Kanalname]_uc_shots". Eine eventuelle Messungszugehörigkeit wird übernommen.

Prinzip der Namensgestaltung bei Option = 1 (Messungen erzeugen)

Der Variablenname wird beibehalten, die jeweilige Messung erhält den Namen "uc_loop[Laufende Nummer der Schleife]", wobei die laufende Schleifennummer entsprechend der Breite der letzten Schleifennummer mit führenden Nullen aufgefüllt wird.

Beispiel: Der folgende Code erzeugt die Variablen "x@uc_loop001", "x@uc_loop002" ... "x@uc_loop100"

```
UNCERTAINTY_LOOP 100
   x = UncertaintyModify(input)
   UncertaintySnapshot(x, 1)
   ...
END
```

Falls der Parameter ein Kanal einer Gruppe ist, erfolgt die Namensbildung nach dem Muster "[Gruppenname]_[Kanalname]@uc_loop###"

Falls der Parameter selbst bereits einer Messung zugewiesen ist, erfolgt die Namensbildung nach dem Muster "[Variablenname]@[Messungsname]_uc_loop###" bzw. "[Gruppenname]_[Kanalname]@[Messungsname]_uc_loop###"

Beispiele:

Aus einem Spannungskanal Input1 mit Messbereich 10V und Messunsicherheit 0.1V soll über einen Algorithmus (Tiefpassfilter, Maximalwert) ein Ergebnis mit Namen Result bestimmt werden. Der folgende Code speichert alle 1000 Variationen des Eingangskanals (die Ergebnisse der Funktion UncertaintyModify) in der Gruppe "x_uc_snapshots".

```
UncertaintySet(Input1, "Uncertainty", 0.1)
UNCERTAINTY_LOOP 1000
   x = UncertaintyModify(Input1)
   UncertaintySnapshot(x, 0)
   Result = Max(FiltLp(x, 0, 0, 1, 0.02))
   UncertaintyCalc(Result)
END
```

Wie zuvor, aber zusätzlich werden auch die 1000 Variationen des Ergebnisses des Algorithmus gesammelt. Es entstehen die Variablen "x@uc_loop0001", "Result@uc_loop0001" bis "x@uc_loop1000", "Result@uc_loop1000".

```
UncertaintySet(Input1, "Uncertainty", 0.1)
UNCERTAINTY_LOOP 1000
   x = UncertaintyModify(Input1)
   UncertaintySnapshot(x, 1)
   Result = Max(FiltLp(x, 0, 0, 1, 0.02))
   UncertaintySnapshot(Result, 1)
   UncertaintyCalc(Result)
END
```

Siehe auch:

UNCERTAINTY_LOOP, UncertaintyModify, UncertaintyCalc, UncertaintySet, UncertaintyGet

Unit?

Eine Einheit eines Datensatzes wird abgefragt.
Alternativer Name: **Einheit?**

Deklaration:

```
Unit? ( Daten, EwCode ) -> TxEinheit
```

Parameter:

Daten	Datensatz, von dem eine Einheit ermittelt werden soll.
EwCode	Angabe, welche Einheit abgefragt werden soll.
	0 : X-Einheit bei einkomponentigen Daten. Einheit der X-Komponente bei XY-Daten. Einheit der Phase bzw. des Imaginärteils bei komplexen Daten.
	1 : Y-Einheit bei einkomponentigen Daten. Einheit der Y-Komponente bei XY-Daten. Einheit des Betrages bzw. des Realteils bei komplexen Daten.
	2 : Z-Einheit
	3 : Einheit des Parameters bei 2-komponentigen Daten
TxEinheit	Die ermittelte Einheit.

Beschreibung:

Beispiele:

Die Y-Einheit eines Datensatzes wird abgefragt. Falls die Einheit "W" ist, wird sie auf "VA" umgesetzt:

```
unitY = Unit?(data, 1)
cmp = TComp(unitY, "W")
IF cmp = 0
   SetUnit(data, "VA", 1)
END
```

Siehe auch:

SetUnit, ConvertUnit, XUNIT, YUNIT

UpperValue

Liefert den jeweils größeren Wert der beiden Parameter.

Deklaration:

```
UpperValue ( Parameter1, Parameter2 ) -> Ergebnis
```

Parameter:

Parameter1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Parameter2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
Ergebnis	Der jeweils größere Wert der beiden Parameter.

Beschreibung:

Die Funktion hat zwei praktische Anwendungen. Bei der Parameterkombination Datensatz/Einzelwert findet eine untere Begrenzung des Datensatzes auf den Wert des 2. Parameters statt. Falls beide Parameter Datensätze sind, ist das Ergebnis die obere Hüllkurve.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder die exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Der Eingangskanal wird in Dezibel umgerechnet und auf eine untere Grenze von -100 dB begrenzt.

```
Channel_01_dB = UpperValue(db(Channel_01), -100)
```

Die untere und obere Einhüllende zweier Datensätze wird bestimmt.

```
Env_L = LowerValue(Channel_01, Channel_02)
Env_H = UpperValue(Channel_01, Channel_02)
```

Siehe auch:

LowerValue, >, RangeSet

UserPropCopy

Kopieren von anwenderdefinierten Eigenschaften.

Deklaration:

```
UserPropCopy ( Ziel, Quelle, TxPropName )
```

Parameter:

Ziel	Die Zielvariable, in die die Eigenschaften kopiert werden sollen.
Quelle	Variable mit den zu kopierenden Eigenschaften.
TxPropName	

Beschreibung:

Im Namen der Eigenschaft können auch die Jokerzeichen '*' (steht für beliebige Anzahl beliebiger Zeichen) und '?' (steht für genau ein beliebiges Zeichen) angegeben werden. Es werden dann alle Eigenschaften kopiert, deren Name dem gegebenen Muster entspricht.

Beim Namen von anwenderdefinierten Eigenschaften wird bezüglich Groß-/Kleinschreibung nicht unterschieden.

Bei den von imc bereits fest vordefinierten Eigenschaften kann statt dem internen Namen ("imc??") auch der Anzeigename angegeben werden (z.B. "imc33" oder "Uncertainty").

Beispiele:

Eine Variable wird Hochpass-gefiltert. In das Ergebnis werden außerdem alle anwenderdefinierten Eigenschaft der Quelle kopiert, deren Name mit 'Env.' beginnt.

```
Channel1_Filtered =  FiltHP(Channel1, 0, 0, 4, 100)
UserPropCopy(Channel1_Filtered, Channel1, "Env.*")
```

Siehe auch:

UserPropSet, UserPropText?, UserPropValue?

UserPropCount?

Anzahl der anwenderdefinierten Eigenschaften einer Variablen abfragen.

Deklaration:

```
UserPropCount? ( Variable ) -> EwAnzahl
```

Parameter:

Variable	Die abzufragende Variable
EwAnzahl	Anzahl der anwenderdefinierten Eigenschaften.

Beschreibung:

Die Funktion wird zusammen mit der Funktion UserPropName?() verwendet, um alle anwenderdefinierten Eigenschaften einer Variablen aufzuzählen.

Beispiele:

Zur Variablen 'Channel' werden alle anwenderdefinierten Eigenschaften aufgezählt und zusammen mit ihrem aktuellen Inhalt im Ausgabefenster angezeigt.

```
n = UserPropCount?(Channel)
FOR i = 1 TO n
   TxName = UserPropName?(Channel, i)
   TxValue = UserPropText?(Channel, TxName)
   BoxOutput(TxName + " : " + TxValue, EMPTY, "", 1)
END
```

Siehe auch:

UserPropName?

UserPropDel

Löschen von anwenderdefinierten Eigenschaften.

Deklaration:

```
UserPropDel ( Variable, TxPropName )
```

Parameter:

Variable	Variable, von der die Eigenschaft entfernt werden soll.
TxPropName	Gibt die zu löschende Eigenschaft(en) an. Leerer Text oder "*", um alle Eigenschaften zu löschen.

Beschreibung:

- Im Namen der Eigenschaft können auch die Jokerzeichen '*' (steht für beliebige Anzahl beliebiger Zeichen) und '?' (steht für genau ein beliebiges Zeichen) angegeben werden. Es werden dann alle Eigenschaften gelöscht, deren Name dem gegebenen Muster entspricht.
- Vordefinierte Eigenschaften mit dem Attribut 'Schreibgeschützt' können nicht gelöscht werden.
- Beim Namen von anwenderdefinierten Eigenschaften wird bezüglich Groß-/Kleinschreibung nicht unterschieden.
- Bei den von imc bereits fest vordefinierten Eigenschaften kann statt dem internen Namen ("imc??") auch der Anzeigename angegeben werden (z.B. "imc33" oder "Uncertainty").

Beispiele:

Von der Variablen 'Kanal1' werden alle Eigenschaften gelöscht, deren Name mit 'Env.' beginnt:

```
UserPropDel(Kanal1, "Env.*")
```

Siehe auch:

UserPropSet, UserPropText?

UserPropInfo?

Informationen über eine anwenderdefinierte Eigenschaft abfragen.

Deklaration:

```
UserPropInfo? ( Variable, TxPropName, TxAuswahl ) -> EwErgebnis
```

Parameter:

Variable	
TxPropName	Der Name der abzufragenden Eigenschaft
TxAuswahl	Auswahl der zu ermittelnden Information
	0 : Vorhanden? - Liefert 1, wenn die Eigenschaft für diese Variable definiert ist. 0 sonst.
	1 : Gültig? - Liefert 1, wenn die Eigenschaft für diese Variable definiert ist und bei numerischen Typen auch initialisiert ist (also nicht leer ist). 0 sonst.
	2 : Datentyp - Liefert den Datentyp der Eigenschaft (1: Text, 2: Ganzzahlig, 3: Reell, 4: Bool, 5: Aufzählungstyp, 6: Zeit) oder -1, wenn die Eigenschaft nicht definiert ist.
	3 : Temporär? - Liefert 1, wenn die Eigenschaft für diese Variable das Attribut 'temporär' besitzt, d.h. beim Speichern der Variablen ignoriert wird. 0 wenn das Attribut nicht gesetzt ist, -1 wenn die Eigenschaft nicht definiert ist.
	4 : Schreibgeschützt? - Liefert 1, wenn die Eigenschaft für diese Variable das Attribut 'schreibgeschützt' besitzt. 0 wenn das Attribut nicht gesetzt ist, -1 wenn die Eigenschaft nicht definiert ist.
EwErgebnis	Ergebnis entsprechend [Auswahl].

Beschreibung:

Beim Namen von anwenderdefinierten Eigenschaften wird bezüglich Groß-/Kleinschreibung nicht unterschieden.

Bei den von imc bereits fest vordefinierten Eigenschaften kann statt dem internen Namen ("imc??") auch der Anzeigename angegeben werden (z.B. "imc33" oder "Uncertainty").

Beispiele:

Alle aktuell in FAMOS vorhandenen Variablen, bei denen die Eigenschaft 'UserComment' definiert ist, werden im Ausgabefenster zusammen mit dem Inhalt dieser Eigenschaft angezeigt.

```
Count = VarGetInit2("*", 2)
n = 1
WHILE n <= Count
   TxVarName = VarGetName?(n)
   IF UserPropInfo?(<TxVarName>, "UserComment", 0)
      TxValue = UserPropText?(<TxVarName>, "UserComment")
      BoxOutput(TxVarName + " : " + TxValue, EMPTY, "", 1)
   END
   n = n+1
END
```

Siehe auch:

UserPropSet, UserPropText?

UserPropName?

Der Name einer anwenderdefinierten Eigenschaft zu einer Variablen wird abgefragt.

Deklaration:

```
UserPropName? ( Variable, EwIndex ) -> TxName
```

Parameter:

Variable	Die abzufragende Variable
EwIndex	Index (beginnend mit 1) der abzufragenden Eigenschaft.
TxName	Name der Eigenschaft.

Beschreibung:

Die Funktion wird zusammen mit der Funktion UserPropCount?() verwendet, um alle anwenderdefinierten Eigenschaften einer Variablen aufzuzählen.

Beispiele:

Zur Variablen 'Channel' werden alle anwenderdefinierten Eigenschaften aufgezählt und zusammen mit ihrem aktuellen Inhalt im Ausgabefenster angezeigt.

```
n = UserPropCount?(Channel)
i = 1
WHILE i <= n
   TxName = UserPropName?(Channel, i)
   TxValue = UserPropText?(Channel, TxName)
   BoxOutput(TxName + " : " + TxValue, EMPTY, "", 1)
   i = i+1
END
```

Siehe auch:

UserPropCount?

UserPropSet

Erstellen/Ändern einer anwenderdefinierten Eigenschaft.

Deklaration:

```
UserPropSet ( Variable, TxPropName, Inhalt, EwTyp, EwAttribute )
```

Parameter:

Variable	
TxPropName	Name der zu ändernden Eigenschaft
Inhalt	Neuer Inhalt bzw. Wert, auf den die Eigenschaft gesetzt werden soll.
EwTyp	Datentyp für die Eigenschaft.
	0 : Automatisch bzw. beibehalten: Beim neuen Anlegen einer Eigenschaft wird der Typ automatisch aus dem übergebenen Parameter bestimmt. Bei Text wird der Typ 'Text' gewählt, bei einer Zahl der Typ 'reell'. Beim Ändern einer existierenden Eigenschaft wird der bestehende Typ beibehalten.
	1 : Beliebiger Text.
	2 : Eine ganze Zahl.
	3 : Eine reelle Zahl.
	4 : Eigenschaft kann nur die Werte 0 oder 1 annehmen.
EwAttribute	Zusätzliche Attribute für die zu setzende Eigenschaft.
	0 : Automatisch bzw. beibehalten: Beim neuen Anlegen einer Eigenschaft wird 'permanent' angenommen, d.h. die Eigenschaft wird beim Speichern der Variablen mit in der Datei abgelegt.
	1 : Temporär: Die Eigenschaft wird als temporär betrachtet und beim Speichern der Variablen ignoriert.
	2 : Permanent: Die Eigenschaft wird beim Speichern der Variablen berücksichtigt (imc-Dateiformat).

Beschreibung:

Der Typ des als 3. Parameters übergebenen neuen Inhalts (Text/Zahl) für die Eigenschaft muss zum deren festgelegten Datentyp passen.

Eigenschaften, deren Name mit 'imc' beginnt, sind für interne Zwecke reserviert. Sie können keine neuen Eigenschaften mit solchen Namen erzeugen und beim Schreiben bestehender Eigenschaften darf nur der Inhalt, aber nicht Datentyp oder Attribute geändert werden. [EwAttribut] und [EwTyp] werden dann ignoriert. Interne Eigenschaften können darüber hinaus auch mit dem Attribut 'Schreibgeschützt' versehen sein. Bei den von imc bereits fest vordefinierten Eigenschaften kann statt dem internen Name auch der Anzeigenname angegeben werden (z.B. "imc33" oder "Uncertainty").

Enthält der Namen für eine Property einen Punkt, so wird der Teil vor dem Punkt als Kategorie-Bezeichnung betrachtet. Sie können so Eigenschaften mit verwandter Bedeutung in einer Kategorie zusammenfassen. Die Kategorie wird z.B. bei der Anzeige der Eigenschaften im Dialog 'Variable'/'Eigenschaften' für eine hierarchische Auflistung der Eigenschaften verwendet.

Für den Namen einer neuen Eigenschaft gelten folgende Einschränkungen:

- Das erste Zeichen darf kein Punkt sein
- Das erste und letzte Zeichen darf kein Leerzeichen sein
- Die folgenden Zeichen dürfen nicht enthalten sein: <>{}[]()&%$§|
- Sonderzeichen mit einem ASCII-Code < 32 (z.B. Tabulator) dürfen nicht enthalten sein

Beim Namen von anwenderdefinierten Eigenschaften wird bezüglich Groß-/Kleinschreibung nicht unterschieden.

Beispiele:

An einen Datensatz werden verschiedene Messungs-spezifische Informationen angetragen, die beim Speichern des Datensatzes im imc-Dateiformat mit in die Datei geschrieben werden:

```
UserPropSet(data, "TestNumber", 212, 2, 0)        ; ganzzahlig
UserPropSet(data, "Env.UserName", "Smith", 1, 0)  ; Text
UserPropSet(data, "Env.Temperature", 21.5 , 3, 0) ; reell
UserPropSet(data, "Env.CalibrationOK", 1, 4, 0)   ; bool
```

Siehe auch:

UserPropText?, UserPropValue?, UserPropCopy

UserPropText?

Abfragen einer anwenderdefinierten Eigenschaft

Deklaration:

```
UserPropText? ( Variable, TxPropName ) -> TxInhalt
```

Parameter:

Variable	Die abzufragende Variable.
TxPropName	Name der abzufragenden Eigenschaft.
TxInhalt	Inhalt der abgefragten Eigenschaft

Beschreibung:

Wenn die Eigenschaft nicht existiert, wird ein leerer Text zurückgegeben. Sie können die Funktion UserPropInfo?() mit der Option 0 verwenden, um vorher die Existenz der Eigenschaft zu prüfen.

Beim Namen von anwenderdefinierten Eigenschaften wird bezüglich Groß-/Kleinschreibung nicht unterschieden.

Bei den von imc bereits fest vordefinierten Eigenschaften kann statt dem internen Namen ("imc??") auch der Anzeigename angegeben werden (z.B. "imc33" oder "Uncertainty").

Beispiele:

Zur Variablen 'Channel' werden alle anwenderdefinierten Eigenschaften aufgezählt und zusammen mit ihrem aktuellen Inhalt im Ausgabefenster angezeigt.

```
n = UserPropCount?(Channel)
i = 1
WHILE i <= n
   TxName = UserPropName?(Channel, i)
   TxValue = UserPropText?(Channel, TxName)
   BoxOutput(TxName + " : " + TxValue, EMPTY, "", 1)
   i = i+1
END
```

Die zweite Spalte einer Tabelle im aktiven Panel wird mit einem Datensatz gefüllt. Die erste Zeile enthält den Namen, die 2. und 3. Zeile den Inhalt zweier anwenderdefinierter Eigenschaften. Ab der 4. Zeile beginnen die Zahlenwerte.

```
PnTableSetCell("Tab1", 2, 1, "channel1")
PnTableSetCell("Tab1", 2, 2, UserPropText?(Kanal1,  "Env.UserName"))
PnTableSetCell("Tab1", 2, 3, UserPropValue?(Kanal1, "Env.Temperature"))
PnTableSetColumn("Tab1", 2, 4, channel1)
```

Siehe auch:

UserPropSet, UserPropValue?

UserPropValue?

Abfragen einer numerischen anwenderdefinierten Eigenschaft.

Deklaration:

```
UserPropValue? ( Variable, TxPropName ) -> EwWert
```

Parameter:

Variable	
TxPropName	Name der abzufragenden Eigenschaft.
EwWert	Wert der abgefragten Eigenschaft.

Beschreibung:

Wenn die Eigenschaft nicht existiert oder nicht initialisiert ist, wird 0 zurückgegeben. Sie können die Funktion UserPropInfo?() mit der Option 1 verwenden, um vorher die Gültigkeit der Eigenschaft zu prüfen.

Die Funktion kann nur für Eigenschaften mit einem numerischen Typ (reell, ganzzahlig, bool) aufgerufen werden.

Beim Namen von anwenderdefinierten Eigenschaften wird bezüglich Groß-/Kleinschreibung nicht unterschieden.

Bei den von imc bereits fest vordefinierten Eigenschaften kann statt dem internen Namen ("imc??") auch der Anzeigename angegeben werden (z.B. "imc33" oder "Uncertainty").

Beispiele:

Eine mehr-kanalige Datei im imc-Format wird geladen. Alle Kanäle enthalten in der Eigenschaft 'Env.Temperature' die Umgebungstemperatur bei der Datenaufnahme. Es werden diejenigen Kanäle angezeigt, bei denen die Umgebungstemperatur größer als 30 Grad war.

```
idFile = FileOpenDSF("c:\imc\dat\xxx.dat", 0)
IF idFile >= 1
   count = FileObjNum?(idFile)
   index = 1
   WHILE index <= count
      TxName  = FileObjName?(idFile, index)
       <TxName>= FileObjRead(idFile, index)
      envTemp = UserPropValue?(<TxName>, "Env.Temperature")
      IF envTemp > 30
         SHOW <TxName>
      END
      index = index + 1
   END
   FileClose(idFile)
END
```

Siehe auch:

UserPropSet, UserPropText?

Value

Liefert zu vorgegebenen x-Positionen die entsprechenden Y-Werte eines Datensatzes.
Alternativer Name: **Wert**

Deklaration:

```
Value ( Daten, XPositionen ) -> Werte
```

Parameter:

Daten	Abzufragender Datensatz. Erlaubte Typen: [ND],[XY].
XPositionen	X-Koordinaten, zu denen die zugehörigen Y-Werte ermittelt werden sollen.
Werte	Ermittelte Y-Werte. [ND]

Beschreibung:

Diese Funktion liefert den Wert des übergebenen Datensatzes an einer vorgegebenen x-Koordinate, d.h. zu einer x-Koordinate wird die entsprechende y-Koordinate ermittelt.

Datentyp ND: Trifft die x-Koordinate nicht genau einen Abtastwert des Datensatzes, wird die x-Koordinate auf den vorhergehenden Abtastwert abgerundet.

Datentyp XY: Die Funktion ist nur auf Datensätze mit streng monotoner X-Spur anwendbar. Trifft die x-Koordinate nicht genau einen Abtastwert des Datensatzes, wird die y-Koordinate linear interpoliert.

Die Funktion ist veraltet, die leistungsfähigere Funktion Value2() ist im Allgemeinen vorzuziehen.

- Das Ergebnis hat dieselbe y-Einheit wie der Eingangs-Datensatz.
- Die x-Position des zu ermittelnden Wertes des Datensatzes ist als x-Koordinate anzugeben, nicht als Index eines Punktes im Datensatz.
- Liegt die geforderte x-Position außerhalb des Bereichs des Datensatzes, wird der Anfangs- oder Endwert des Datensatzes entsprechend geliefert.
- Das Abrunden bei normalen Datensätzen auf den vorhergehenden Abtastwert arbeitet mit einer leichten Unschärfe, ein angegebener x-Wert gilt auch noch als genauer Treffer eines Abtastwertes, wenn er weniger als 1/100 eines Abtastschrittes vor diesem Abtastwert liegt. Ist der gesuchte Wert also nur geringfügig kleiner als ein Abtastwert, wird trotzdem der Wert an dieser Position zurückgegeben. Dieses Verhalten dient dazu, um numerische Abweichungen bei zuvor berechneten x-Koordinaten auszugleichen.
- Der übergebene zweite Parameter sollte die Einheit der x-Koordinate des Datensatzes aufweisen.
- Wenn eine Reihe von x-Koordinaten angegeben wird, werden entsprechend viele y-Werte erzeugt und als Datensatz zurückgegeben.
- Wenn Sie die Zahlenwerte eines Datensatzes wissen möchten, ohne sie weiter verrechnen zu wollen, können Sie auch die Messcursoren im Messwertfenster zum entsprechenden Kurvenfenster benutzen. Beachten Sie, dass Sie zu diesem Zweck sinnvollerweise eine Treppenstufendarstellung wählen. Andererseits können Sie den Dateneditor benutzen, um die Werte in tabellarischer Form zu anzusehen.
- Sie können alternativ mit der Funktion ValueIndex() oder Indizierung mittels eckiger Klammern [...] arbeiten, wenn Sie die gewünschten Positionen über den Index der Punkte im Datensatz angeben wollen.

Beispiele:

Die y-Koordinate eines Datensatzes, der eine Ausdehnung in x-Richtung von 2s ... 10s hat, wird an der x- Koordinate 5s bestimmt:

```
yValue = Value(NDdata, 5 's')
```

Der letzte Wert des Datensatzes wird bestimmt. Die Ausdehnung des Datensatzes ist kleiner als 1E20:

```
yLast = Value(NDdata, 1E20)
```

Der Wert eines Datensatzes an der x-Koordinate 6 wird verdoppelt:

```
NDdata = Set(NDdata, 6, 2 * Value(NDdata, 6))
```

Siehe auch:

Value2, ValueIndex, PosiEx2, Set

Value2

Liefert zu vorgegebenen x-Positionen die entsprechenden Y-Werte eines Datensatzes.
Alternativer Name: **Wert2**

Deklaration:

```
Value2 ( Daten, XPositionen, EwInterpolation ) -> YWerte
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND],[XY]. Die Zeit- bzw. x-Achse muss monoton sein.
XPositionen	X-Koordinaten, zu denen die zugehörigen Y-Werte ermittelt werden sollen.
EwInterpolation	Trifft die x-Koordinate nicht genau einen Abtastwert des Datensatzes, wird das Ergebnis wie folgt interpoliert:
	0 : Linear. Der Eingangsdatensatz wird linear interpoliert.
	1 : Konstant, davorliegender Wert. Der Eingangsdatensatz wird konstant interpoliert, d.h. jeder Wert wird konstant gehalten, bis der nächste Wert gültig wird. Als Ergebniswert wird also derjenige Wert des Eingangsdatensatzes verwendet, dessen x-Koordinate unmittelbar VOR der gesuchten x-Koordinate liegt.
	2 : Konstant, nächstliegender Wert. Als Ergebniswert wird derjenige Wert des Eingangsdatensatzes verwendet, dessen x-Koordinate am NÄCHSTEN zur gesuchten x-Koordinate liegt.
YWerte	Die ermittelten Y-Werte.

Beschreibung:

Diese Funktion liefert die Werte des übergebenen Datensatzes an vorgegebenen x-Koordinaten, d.h. zu den x-Koordinaten werden die entsprechenden y-Koordinaten ermittelt.

- Liegt die geforderte x-Position außerhalb des Bereichs des Datensatzes, wird der Anfangs- bzw. Endwert des Datensatzes geliefert.
- Lineare Interpolation wird im Allgemeinen bei kontinuierlichen Eingangssignalen verwendet.
- Die konstanten Interpolationsarten werden dann sinnvoll verwendet, wenn das Eingangssignal aus vordefinierten diskreten Werten gebildet wird. Dies ist z.B. bei digitalen Daten der Fall oder auch bei Messdaten, die Ihrer Bedeutung nach nur ganzzahlig sein können (z.B. die aktuelle Getriebestufe bei Antrieben). Der Ergebnisdatensatz besteht nur aus Werten, die auch im Eingangsdatensatz vorhanden sind, und besitzt das selbe Datenformat.
- Lineare Interpolation bei digitalen Eingangsdaten ist nicht möglich, der Interpolationsparameter wird dann ggf. automatisch auf den Wert 1 (konstante Interpolation) korrigiert.
- Sie können alternativ mit der Funktion ValueIndex() arbeiten, wenn Sie die gewünschten Positionen über den Index der Punkte statt über die x-Koordinate im Datensatz angeben wollen.

Besonderheit für EwInterpolation = 1 (konstant, davorliegend):
In diesem Fall wird mit einer leichten Unschärfe gearbeitet, ein angegebener x-Wert gilt auch noch als genauer Treffer eines Abtastwertes, wenn er weniger als 1/1000 eines Abtastschrittes vor diesem Abtastwert liegt. Ist der gesuchte Wert also nur geringfügig kleiner als ein Abtastwert, wird trotzdem der Wert an dieser Position zurückgegeben. Dieses Verhalten dient dazu, um numerische Abweichungen bei zuvor berechneten X-Koordinaten auszugleichen.

Beispiele:

Der Y-Wert eines Datensatzes an der Stelle X=5 wird bestimmt. Es wird ggf. linear interpoliert.

```
SignalAt5 = Value2(Signal, 5 's', 0)
```

An einem Fahrzeug werden Geschwindigkeit <velocity> und Getriebestufe (<gear>, enthält nur die Werte 1 - 5) gemessen. Es werden alle Zeiten bestimmt, an denen das Fahrzeug über 50 km/h beschleunigt hat und der dabei jeweils aktive Gang ermittelt.

```
t = PosiEx2(velocity, 50, 1, 0)
GearAt50 = Value2(Gear, t, 1)
GearAt50XY = XYof(t, GearAt50)
```

Siehe auch:

ValueIndex, PosiEx2, MatrixGet, RSampEx

ValueIndex

Liefert zu vorgegebenen Indizes die entsprechenden y-Werte eines Datensatzes.
Alternativer Name: **WertIndex**

Deklaration:

```
ValueIndex ( Daten, Indizes ) -> Werte
```

Parameter:

Daten	Abzufragender Datensatz. Erlaubte Typen: [ND],[XY].
Indizes	Indizes der Punkte, zu denen die zugehörigen y-Werte ermittelt werden sollen.
Werte	Ermittelte Y-Werte. [NW]

Beschreibung:

Für den Datensatz [Daten] werden zu allen vorgegebenen Positionen (Indizes) die zugehörigen y-Werte ermittelt.

Die in [Indizes] angegebenen Positionen (Indizes) sollten zwischen 1 und der Datensatzlänge von [NyDaten] liegen.

Für Indizes < 1 wird der erste y-Wert, für Indizes > Datensatzlänge der letzte Wert des Datensatzes zurückgegeben.

Um in imc FAMOS einen einzelnen Wert über dessen Index im Datensatz abzufragen, können Sie den Datensatz in Formeln auch indizieren:

```
singleValue = Data[ Index ]
```

- Der erste Punkt in einem Datensatz hat den Index 1, der Index des letzten Punktes entspricht der Datensatzlänge.
- Alternativ kann die Funktion Value2() verwendet werden, bei der die Positionen als x-Koordinaten der Punkte definiert werden.
- Wenn eine Reihe von Indizes angegeben wird, werden entsprechend viele y-Werte erzeugt und als Datensatz zurückgegeben.
- Wenn Sie die Zahlenwerte eines Datensatzes wissen möchten, ohne sie weiter verrechnen zu wollen, können Sie auch die Messcursoren im Messwertfenster zum entsprechenden Kurvenfenster benutzen. Beachten Sie, dass Sie zu diesem Zweck sinnvollerweise eine Treppenstufendarstellung wählen. Andererseits können Sie den FAMOS-Dateneditor benutzen, um die Werte in tabellarischer Form anzusehen.

Beispiele:

Der letzte y-Wert des Datensatzes wird berechnet:

```
LastValue = ValueIndex(Data, Leng?(Data))
; auch:
LastValue = Data[Leng?(Data)]
```

Jeder zweite y-Wert des Datensatzes wird verdoppelt:

```
Indizes = Ramp(1, 2, Leng?(Data)/2) ;Indizes = 1,3,5..
NewY= ValueIndex(Data, Indizes) * 2
DataNew = SetIndex(Data, Indizes, NewY)
```

Siehe auch:

Value2, CutIndex, SamplesGate, ReplIndex, SetIndex, MatrixGet

VarExist?

Abfrage, ob eine Variable mit einem bestimmten Namen in imc FAMOS existiert.

Deklaration:

```
VarExist? ( TxVariablenName ) -> EwVorhanden
```

Parameter:

TxVariablenName	Name der gesuchten Variablen
EwVorhanden	0: Nicht vorhanden / 1: Vorhanden

Beschreibung:

Prüft, ob eine Variable mit dem angegebenen Namen zum Zeitpunkt des Aufrufes in imc FAMOS existiert.

Zu Gruppen gehörige Datensätze können in der üblichen Form 'GruppenName:KanalName' angegeben werden.

Lokale Variablen werden nicht gefunden.

Beispiele:

```
FileLoad("c:\imc\dat\multi.dat", "", 0) ; Multikanaldatei
IF VarExist?("channel012")
   SHOW channel012
END
```

Eine Multikanaldatei wird geladen. Es wird geprüft, ob in dieser ein Kanal mit einem bestimmten Namen vorhanden ist. Falls ja, wird er angezeigt.

Siehe auch:

VarGetInit2, VarGetInit, VarGetName?, Name?

VarGetInit

Abfrage der zum Start der Sequenzausführung vorhandenen FAMOS-Variablen. Initialisierung für nachfolgende VarGetName?()-Aufrufe.

Deklaration:

```
VarGetInit ( EwOption ) -> EwAnzahl
```

Parameter:

EwOption	Optionsparameter
	0 : Alle Einträge
	1 : Nur selektierte Einträge
	2 : Alle Einträge, erweiterte Syntax
	3 : Nur selektierte Einträge, erweiterte Syntax
	4 : Anzahl der übergebenen Sequenz-Parameter. Dieser Aufruf ist veraltet, neu erstellte Sequenzen sollten statt dessen die Funktion ParametersPassed?() (verfügbar seit Version 2024) verwenden.
EwAnzahl	Anzahl der gefundenen Variablen.

Beschreibung:

Diese Funktion VarGetInit() liefert den Inhalt der Standard-Variablenliste beim Start der (Sequenz-) Ausführung. Im Gegensatz dazu liefert die Funktion VarGetInit2() die zum Aufrufzeitpunkt aktuell vorhandenen Variablen.

Während eine Sequenz abgearbeitet wird, ist der Zustand der Variablenliste unbestimmt. Eine Ausnahme davon ist die Abarbeitung einer Sequenz im Einzelschritt-Modus, hier wird der Zustand der Variablenliste zum Zeitpunkt des Funktionsaufrufes verwendet.

Die Funktion wird verwendet, um eine anschließende Abfrage der Variablenliste mit der Funktion VarGetName?() einzuleiten.

Lokale Variablen werden ignoriert.

Damit ein Variablenname in Formeln angegeben werden kann, muss er bestimmten Regeln gehorchen (z.B. keine Leerzeichen, erstes Zeichen keine Ziffer etc.). Falls dies nicht der Fall ist, muss der Name zusätzlich in geschweifte Klammern {...} eingeschlossen werden. Bei den Optionen 2 und 3 wird der bei VarGetName?() zurückgegebene Name ggf. automatisch um die geschweiften Klammern ergänzt, diese sind daher für diese Art der Anwendung vorzuziehen.

Mit Hilfe dieser beiden Funktionen können Sequenzen geschrieben werden, die beispielsweise alle momentan selektierten Variablen bearbeiten.

Die Funktion mit den Optionen 0-3 liefert den tatsächlich sichtbaren Zustand der Variablenliste, berücksichtigt also z.B. auch ein eingestelltes Anzeigefilter und ist auch nur anwendbar, wenn das FAMOS Hauptfenster mit Variablenliste überhaupt sichtbar ist (z.B. nicht gewährleistet bei Aufruf als Runtime oder bei der Fernsteuerung durch imc STUDIO). Wenn alle vorhandenen Variablen benötigt werden, ist die Funktion VarGetInit2() zu verwenden.

Die Option '4' wirkt nicht auf die Variablenliste, sondern hat eine spezielle Anwendung: Wenn die aktuell ausgeführte Sequenz mittels des Befehls SEQUENCE aufgerufen wurde, liefert dieser Aufruf die Anzahl der übergebenen Sequenz-Parameter, ansonsten -1.

Verweis: Die Funktion BoxVarSelector() zeigt eine Liste mit den zum Aufrufzeitpunkt vorhandenen Variablen, unter denen Sie dann interaktiv eine Auswahl treffen können.

Beispiele:

Für alle selektierten Einträge in der Variablenliste wird der Mittelwert gebildet:

```
count = VarGetInit(1)
FOR index = 1 TO count
   TxName = VarGetName?(index)
   TxNameMean = TxName + "_M"
   <TxNameMean> = Mean(<TxName>)
END
```

Eine Untersequenz erwartet eigentlich 2 Parameter. Falls nur ein Parameter übergeben wird, wird für den zweiten Parameter ein Standardwert angenommen.

```
ParCount = VarGetInit(4)
; oder, seit FAMOS 2024: ParCount = ParametersPassed?()
SWITCH ParCount
   CASE 2
      ; ok, PA1 und PA2 sind angegeben
   CASE 1
      ; PA2 fehlt, Standardwert annehmen
      PA2 = 2
   DEFAULT
      BoxMessage("Fehler", "Fehler beim Aufruf", "!1")
      EXITSEQUENCE
```

```
END
```

Siehe auch:

VarGetInit2, VarGetName?, ParametersPassed?, BoxVarSelector, VarExist?

VarGetInit2

Abfrage der zum Aufruf-Zeitpunkt vorhandenen FAMOS-Variablen. Initialisierung für nachfolgende VarGetName?()-Aufrufe.

Deklaration:

```
VarGetInit2 ( TxNamensMuster, EwOption ) -> EwAnzahl
```

Parameter:

TxNamensMuster	Namensmuster für die Auflistung der Variablen. Die Bedeutung ist abhängig vom Wert des Parameters [Option]. Die Jokerzeichen "*" und "?" können in Ihrer üblichen Interpretation verwendet werden.
EwOption	Optionsparameter
	0 : Es werden alle Variablen berücksichtigt, deren Name dem angegebenen Namensmuster entspricht. Um alle vorhandenen Variablen aufzulisten, geben Sie einen leeren Text oder "*" an.
	1 : Es werden alle Variablen berücksichtigt, deren Name NICHT dem angegebenen Namensmuster entspricht.
	2 : Es werden alle Variablen berücksichtigt, deren Name dem angegebenen Namensmuster entspricht. Variablennamen mit erweiterter Syntax werden für die direkte Verwendung in Formeln aufbereitet (siehe Anmerkungen).
	3 : Es werden alle Variablen berücksichtigt, deren Name NICHT dem angegebenen Namensmuster entspricht. Variablennamen mit erweiterter Syntax werden für die direkte Verwendung in Formeln aufbereitet (siehe Anmerkungen).
EwAnzahl	Anzahl der gefundenen Variablen.

Beschreibung:

Die Funktion wird verwendet, um eine anschließende Aufzählung der vorhandenen imc FAMOS-Variablen mit der Funktion VarGetName?() einzuleiten

Die Funktion VarGetInit2() listet die zum Aufrufzeitpunkt aktuell vorhandenen Variablen. Die Funktion VarGetInit() liefert dagegen die vorhandenen Variablen bei Beginn der (Sequenz-)Ausführung.

Zur Aufzählung der Variablennamen verwenden Sie anschließend die Funktion VarGetName?().

Lokale Variablen werden ignoriert.

Damit ein Variablenname in Formeln angegeben werden kann, muss er bestimmten Regeln gehorchen (z.B. keine Leerzeichen, erstes Zeichen keine Ziffer etc.). Falls dies nicht der Fall ist, muss der Name zusätzlich in geschweifte Klammern {...} eingeschlossen werden. Bei den Optionen 2 und 3 wird der bei VarGetName?() zurückgegebene Name ggf. automatisch um die geschweiften Klammern ergänzt, diese sind daher für diese Art der Anwendung vorzuziehen.

Verweis: Die Funktion BoxVarSelector() zeigt eine Liste mit den zum Aufrufzeitpunkt vorhandenen Variablen, unter denen Sie dann interaktiv eine Auswahl treffen können.

Beispiele:

Alle Variablen, deren Name mit dem Vorsatz "channel" beginnt, werden in einem Kurvenfenster angezeigt.

```
Count = VarGetInit2("channel*", 2)
FOR n = 1 TO Count
   TxVarName = VarGetName?(n)
   SHOW <TxVarName>
END
```

Alle Variablen, deren Name NICHT mit dem Zeichen "$" beginnt, werden zusammen in einer Datei gespeichert.

```
$fh = FileOpenDSF("z:\allvars.dat",1)
$Count = VarGetInit2("$*", 3)
FOR $i = 1 TO $Count
   $TxVarName = VarGetName?($i)
   FileObjWrite($fh, <$TxVarName>)
END
FileClose($fh)
```

Siehe auch:

VarGetInit, VarGetName?, BoxVarSelector, VarExist?

VarGetName?

Abfrage eines Eintrages der FAMOS-Variablenliste.

Deklaration:

```
VarGetName? ( EwIndex ) -> TxName
```

Parameter:

EwIndex	Index des Eintrags (1..)
TxName	Name des gewünschtes Variablen-Eintrags

Beschreibung:

Vor Verwendung dieser Funktion muss entweder die Funktion **VarGetInit()** oder **VarGetInit2()** zur Initialisierung aufgerufen werden.

Die Funktion VarGetInit2() listet die zum Aufrufzeitpunkt aktuell vorhandenen Variablen. Die Funktion VarGetInit() liefert dagegen den Inhalt der imc FAMOS-Variablenliste bei Beginn der (Sequenz-)Ausführung.

Beispiele:

Für alle selektierten Einträge in der Variablenliste wird der Mittelwert gebildet:

```
count = VarGetInit(1)
FOR index = 1 TO count
   TxName = VarGetName?(index)
   TxNameMean =  TxName + "_M"
   <TxNameMean> = Mean(<TxName>)
END
```

Alle Variablen, deren Name NICHT mit dem Zeichen "$" beginnt, werden zusammen in einer Datei gespeichert.

```
$fh = FileOpenDSF("z:\allvars.dat",1)
$Count = VarGetInit2("$*", 3)
FOR $i = 1 TO $Count
   $TxVarName = VarGetName?($i)
   FileObjWrite($fh, <$TxVarName>)
END
FileClose($fh)
```

Siehe auch:

VarGetInit, VarGetInit2, VarExist?, Name?

Verify

Verfügbar ab: Professional Edition

Prüft, ob ein Wert wahr, d.h. ungleich null ist. Gibt eine Fehlermeldung aus, falls nicht. Dann bricht auch die Sequenzausführung ab.

Deklaration:

```
Verify ( Value [, Fehlertext] )
```

Parameter:

Value	Dieser Wert wird auf ungleich 0.0 geprüft.
Fehlertext	Text, der im Fehlerfall ausgegeben wird. (optional)

Beschreibung:

Verify() wird benutzt, um Prüfungen von Eingaben oder Selbstüberprüfungen kompakt in Sequenzen einzubauen.

Beispiele:

Anzeige eines Fehlers, weil A und B nicht gleich sind und Equal() eine 0 liefert.

```
A=2
B=sqrt(2)^2 ; A <> B !
verify( equal( A, B ))
```

Siehe auch:

Equal

VerifyVar

Prüft, ob eine Variable gegebene Bedingungen erfüllt.

Deklaration:

```
VerifyVar ( Variable, TxBedingung1 [, TxBedingung2] [, B3] [, B4] [, B5] [, B6] [, B7] [, B8] [, B9] [, B10] [, B11] [, B12] [, B13] [, B14] )
-> OK
```

Parameter:

Variable	Zu prüfende Variable
TxBedingung1	1.Bedingung
	"SV" : Einzelwert (Normaler Datensatz der Länge 1)
	"ND" : Normaler Datensatz
	"ND(->)" : Normaler Datensatz, keine Events, keine Segmente
	"ND(..)" : Normaler Datensatz, Länge > 1
	"ND(..,->)" : Normaler Datensatz, keine Events, keine Segmente, Länge > 1
	"CX" : Komplexer Datensatz
	"!CX" : Datensatz, nicht komplex
	"CX(->)" : Komplexer Datensatz, keine Events, keine Segmente
	"RI" : Komplexer Datensatz in Real-/Imaginärteil-Darstellung
	"MP" : Komplexer Datensatz in Betrag/Phase-Darstellung
	"DP" : Komplexer Datensatz in Dezibel/Phase-Darstellung
	"XY" : XY-Datensatz
	"!XY" : Datensatz, kein XY
	"XY(->)" : XY-Datensatz, keine Events, keine Segmente
	"XY(/)" : XY-Datensatz mit monoton wachsender x-Spur.
	"XY(->,/)" : XY-Datensatz mit monoton wachsender x-Spur. Keine Events, keine Segmente.
	"TSA" : Datensatz, TimeStamp-ASCII
	"!TSA" : Datensatz, kein TimeStamp-ASCII
	"TSA(->)" : TimeStamp-ASCII, keine Events, keine Segmente
	"DS" : Datensatz (Typ beliebig), also kein Text, Textfeld oder Datengruppe
	"Evn" : Datensatz (Typ beliebig) mit Events
	"Seg" : Datensatz (Typ beliebig) mit Segmenten
	"SegEvn" : Datensatz (Typ beliebig) mit Segmenten und Events
	"!Evn" : Datensatz (Typ beliebig) ohne Events
	"!Seg" : Datensatz (Typ beliebig) ohne Segmente
	"!SegEvn" : Datensatz (Typ beliebig), keine Events, keine Segmente
	"->" : Abkürzung für "!SegEvn", Datensatz (Typ beliebig), keine Events, keine Segmente
	"Monotone" : Einzelwert, Normaler Datensatz oder XY-Datensatz mit monoton wachsender x-Spur.
	"/" : Abkürzung für "Monotone". Einzelwert, Normaler Datensatz oder XY-Datensatz mit monoton wachsender x-Spur.
	"Digital" : Digitaler Datensatz
	"!Digital" : Datensatz, nicht digital.
	"TXT" : Text
	"!TXT" : Kein Text, also Datensatz, Textfeld oder Datengruppe
	"TXA" : Textfeld
	"!TXA" : Kein Textfeld, also Datensatz, Text oder Datengruppe
	"TX*" : Text oder Textfeld
	"!TX*" : Kein Text oder Textfeld, also Datensatz oder Datengruppe
	"Group" : Datengruppe
	"!Group" : Keine Datengruppe
	"GrMember" : Element einer Datengruppe
	"!GrMember" : Kein Element einer Datengruppe
	"Empty" : Variable ist leer. Länge 0 bei Datensätzen, Textlänge 0 bei Texten, Elementanzahl 0 bei Textfeldern und TSA.
	"!Empty" : Variable ist nicht leer.
	"@M" : Die Variable ist einer Messung zugeordnet.
	"!@M" : Die Variable ist keiner Messung zugeordnet.
	"ReadOnly" : Schreibschutz
	"!ReadOnly" : Kein Schreibschutz

	"V74Compat" : Variable hat keine Eigenschaften, die in künftigen Versionen von FAMOS eingeführt werden und die Kompatibilität existierender Sequenzen möglicherweise gefährden können.
TxBedingung2	2.Bedingung (optional)
B3	3.Bedingung (optional)
B4	4.Bedingung (optional)
B5	5.Bedingung (optional)
B6	6.Bedingung (optional)
B7	7.Bedingung (optional)
B8	8.Bedingung (optional)
B9	9.Bedingung (optional)
B10	10.Bedingung (optional)
B11	11.Bedingung (optional)
B12	12.Bedingung (optional)
B13	13.Bedingung (optional)
B14	14.Bedingung (optional)
OK	Eine 1, wenn die Variable alle angegebenen Bedingungen erfüllt, 0 sonst.

Beschreibung:

Die Funktion prüft, ob eine Variable die angegebenen Bedingungen erfüllt. Sie ist hilfreich, um bereits am Beginn einer Auswertung zu prüfen, ob Eingangsvariablen den Erwartungen entsprechen. Dadurch können nachfolgende Fehler bei der Sequenzausführung verhindert werden, z.B. wenn eine Variable einen unerwarteten Datentyp besitzt und dieser von aufgerufenen Funktionen nicht verarbeitet werden kann.

Die angegebenen Bedingungen werden UND-verknüpft, das Ergebnis ist also genau dann 1, wenn ausnahmslos alle Bedingungen erfüllt sind.

Sonderfall: Wenn als **erste** Bedingung "Group" angegeben wird, gilt folgende Besonderheit:

Wenn die Variable vom Typ "Datengruppe" ist, werden alle nachfolgenden Bedingungen auf alle enthaltenen Kanäle angewendet (UND-verknüpft). So kann bequem geprüft werden, ob alle Elemente einer Datengruppe einer gegebenen Anforderung genügen.

Test, ob der Parameter eine Datengruppe ist und alle Elemente Einzelwerte sind:

```
ok = VerifyVar(MyGroup, "Group", "SV")
```

Test, ob der Parameter eine Datengruppe ist und alle Elemente leer sind:

```
ok = VerifyVar(MyGroup, "Group", "Empty")
```

Test, ob der Parameter eine Datengruppe ohne Elemente ist:

```
ok = VerifyVar(MyGroup, "Empty" "Group")
```

Beispiele:

Eine Sequenz berechnet das gleitende Maximum des 1. Parameters.

Am Beginn der Sequenz wird geprüft, ob der übergebene Parameter ein unstrukturierter Datensatz ist (keine Events, keine Segmente). Außerdem wird getestet, ob er ein äquidistant abgetasteter Datensatz oder ein XY-Datensatz mit monoton wachsender x-Achse ist. Falls nicht, wird eine Fehlermeldung angezeigt und die Sequenz beendet.

Wenn es sich um einen XY-Datensatz handelt, wird er vor der weiteren Verarbeitung äquidistant nachabgetastet. Damit ist sichergestellt, dass der nachfolgende Algorithmus fehlerfrei durchläuft.

```
par1 = PA1
IF NOT(VerifyVar(par1, "!SegEvn", "Monotone"))
   ThrowError("Ungültiger Datentyp des ersten Parameters.")
END

IF VerifyVar(par1, "XY")
   par1 = XYdt(Cmp1(par1), Cmp2(par1), 0.1)
END

par1 = Smo5(par1)
result = MvMax(par1, 1, 1)
```

Die nachfolgende Sequenzfunktion prüft, ob ein Datensatz komplett oberhalb eines gegebenen Referenzdatensatzes verläuft. Wenn die Funktion wegen unpassender Parameter nicht ausgeführt werden kann, wird dies durch einen speziellen Rückgabewert signalisiert. Der Aufrufer kann dann die Fehlerursache mittels GetLastError() abfragen.

```
; Deklaration: !CheckAbove(TestData, Lower) => Result
; Result = 1:  OK
; Result = 0:  Mindestens 1 Wert von [TestData] ist kleiner als der korrespondierende Wert in [Lower]
; Result = -1: Fehler: Eingangsdaten passen nicht zusammen (bzgl. ihrer x-Achse) oder haben Events/Segmente

OnError("Return", "Unbekannter Fehler", "Result", -1)
IF NOT(VerifyVar(TestData, "!SegEvn")) OR NOT(VerifyVar(Lower, "!SegEvn"))
   ThrowError("Die Parameter dürfen weder Segmente noch Events besitzen!")
END
Verify(Leng?(TestData)=Leng?(Lower) AND xdel?(TestData)=xdel?(Lower) AND xoff?(TestData)=xoff?(Lower), "Parameter: Inkompatible x-Skalierung!")
Result = Max(Lower - TestData) < 0
```

Nachfolgend wird geprüft, ob die Variable [test] entweder ein normaler Datensatz oder ein XY-Datensatz mit mindestens 10 Samples ist:

```
ok = VerifyVar(test, "ND")) OR VerifyVar(test, "XY"))
ok = ok AND (Leng?(test) >= 10)
```

Nachfolgend wird geprüft, ob die Variable [gr] eine Datengruppe repräsentiert, die ausschließlich normale Datensätze oder komplexe Datensätze in Betrag/Phase-Darstellung enthält:

```
ok = VerifyVar(gr, "Group", "ND")) OR VerifyVar(gr, "GROUP", "BP"))
```

Siehe auch:

Verify, OnError, ThrowError, DataFormat?

VFAppendCwSnapshot

Der Inhalt des aktuell selektierten Kurvenfensters wird in eine Bitmap exportiert und diese an eine vorher mit VFOpen() geöffnete Videodatei angehängt.

Deklaration:

```
VFAppendCwSnapshot ( Datei-ID ) -> Erfolg
```

Parameter:

Datei-ID	ID der geöffneten Videodatei. Entspricht dem Rückgabewert der Funktion VFOpen()
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Das Kurvenfenster muss vor dem Aufruf dieser Funktion erzeugt und selektiert werden. Dazu können die Funktionen des Kurvenfenster- Kits (CwSelectWindow() etc.) benutzt werden.

Das Bild wird mit der aktuellen Größe des Kurvenfenster-Inhalts auf dem Bildschirm erzeugt.

Wenn die Größe der resultierenden Bitmap nicht mit der Bildgröße des Videos übereinstimmt, wird die Bitmap zentriert auf den Videobild platziert.

Beispiele:

In einem Verzeichnis liegen diverse Messdateien-Dateien als Ergebnis eines Versuchs. Alle Dateien, die einem gegebenen Namensmuster entsprechen, werden nacheinander in ihrer Erzeugungsreihenfolge geladen, in einem Kurvenfenster mit fester Konfiguration angezeigt und die Kurvenbilder zu einem Video zusammengefügt. Die Videobildgröße entspricht der Größe des Kurvenbildes auf dem Schirm, das Videoformat ist MP4 in Standardqualität mit 2 Bildern pro Sekunde.

```
CwSelectMode("variable")
id = VFOpen( "c:\video\test_2019_04_18.mp4", 0, 0, 0, 0, 2)
IF id > 0
   CwLoadCCV(id, "data.ccv") ;curve configuration with fixed channel name "data"
   CwSelectWindow(id)
   fileNames = FsGetFileNames("c:\test\2019_04_18","ch*.raw", 0, 0, 2)
   FOREACH ELEMENT fileName in fileNames
      fh = FileOpenDSF(fileName, 0)
      IF fh > 0
         data = FileObjRead(fh, 1)
         FileClose(fh)
          VFAppendCwSnapshot(id )
      END
   END
   VFClose(id)
END
```

Siehe auch:

VFOpen, VFAppendFrame, VFAppendPanelSnapshot, VFAppendRGBData, VFClose

VFAppendFrame

Eine Bitmap wird aus einer Bilddatei geladen und an eine vorher mit VFOpen() geöffnete Videodatei angehängt.

Deklaration:

```
VFAppendFrame ( Datei-ID, BildDatei ) -> Erfolg
```

Parameter:

Datei-ID	ID der geöffneten Videodatei. Entspricht dem Rückgabewert der Funktion VFOpen()
BildDatei	Kompletter Pfad auf eine Bitmap-Bild-Datei.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die folgenden Dateitypen werden unterstützt: BMP, GIF, JPEG, PNG, TIFF mit Farbtiefen von 24 oder 32Bit per Bildpunkt sowie 8Bit-Bitmaps mit Farbtabelle.

Wenn die Größe der resultierenden Bitmap nicht mit der Bildgröße des Videos übereinstimmt, wird die Bitmap zentriert auf den Videobild platziert.

Beispiele:

In einem Verzeichnis liegen diverse PNG-Dateien, die zu einem Video (MP4 mit Auflösung 1280x720, 1fps) zusammen gebunden werden.

```
 id = VFOpen( "d:\data.mp4", 0, 0, 1280, 720, 1)
IF id > 0
   fileNames = FsGetFileNames("d:\report_exports","*.png", 0, 0, 2)
   FOREACH ELEMENT fileName in fileNames
      VFAppendFrame(id, fileName)
   END
   VFClose(id)
END
```

Siehe auch:

VFOpen, VFAppendPanelSnapshot, VFAppendCwSnapshot, VFAppendRGBData, VFClose

VFAppendPanelSnapshot

Eine Seite des aktiven Panels wird als Bitmap exportiert und diese an eine vorher mit VFOpen() geöffnete Videodatei angehängt.

Deklaration:

```
VFAppendPanelSnapshot ( Datei-ID, Seitenwahl, Größe ) -> Erfolg
```

Parameter:

Datei-ID	ID der geöffneten Videodatei. Entspricht dem Rückgabewert der Funktion VFOpen()
Seitenwahl	Auswahl der zu exportierenden Seite.
	-1 : Die aktive Seite wird exportiert.
	>=1 : Seitennummer der zu exportierenden Seite.
Größe	Größenangabe für die exportierte Bitmap. Nur bei Report-Seiten verwendet, bei Dialog-Seiten wird dieser Wert ignoriert und immer die aktuelle Größe auf dem Schirm verwendet.
	0 : Es werden die Abmessungen entsprechend der aktuellen Anzeige auf dem Bildschirm verwendet. Die resultierende Größe dadurch abhängig von der aktuellen Zoomstufe.
	>=72 : Der Wert gibt die zu wählende Auflösung in 'dpi' (Dots per Inch = Punkte pro Zoll) an. Übliche Werte sind z.B. 150dpi oder 300dpi. Der Wert muss im Bereich 72 - 600dpi liegen. Beispiel: Bei einer Report-Seite im A4-Quer-Format (297x210mm) mit jeweils 10mm Seitenrand (resultierende Größe also 277x190mm) und 150DPI ergibt sich eine Bitmapgröße von (277*150/25.4, 190*150/25.4) = (1636x1124) Pixeln.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Wenn die Größe der resultierenden Bitmap nicht mit der Bildgröße des Videos übereinstimmt, wird die Bitmap zentriert auf den Videobild platziert.

Bei Dialog-Seiten wird ein möglichst exaktes Abbild der Bildschirmausgabe erzeugt. Alle sichtbaren Widgets werden ausgegeben, die Widget-Eigenschaft 'Drucken/Export' wird ignoriert.

Bei Report-Seiten wird ein möglichst exaktes Abbild der Druckausgabe (ohne Seitenränder) erzeugt. Die Widget-Eigenschaft 'Drucken/Export' wird berücksichtigt.

Beispiele:

Ein Panel hat einen Knopf "Take Snapshot".Mit jeder Betätigung des Knopfes wird ein Abbild des aktuellen Inhaltes der ersten Panelseite an eine beim Laden des Panels geöffnete Video-Datei angehängt. Mit dem Schließen des Panels wird auch die Video-Datei geschlossen.

Ereignis-Sequenz 'Init':

```
; Öffnen der Video-Datei im WMV-Format mit 2 fps. Videobildgröße entspricht der Größe des ersten zugefügten Bildes.
videoID = VFOpen( "d:\evaluation.wmv", 0, 0, 0, 0, 2)
```

Ereignis-Sequenz 'Knopf gedrückt' für den 'Take Snapshot'-Knopf:

```
; Anhängen des aktuellen Panel-Bildes (Seite 1) in Originalgröße
VFAppendPanelSnapshot( videoID, 1, 0)
```

Ereignis-Sequenz 'Ende':

```
VFClose(videoID)
```

Siehe auch:

VFOpen, VFAppendFrame, VFAppendCwSnapshot, VFAppendRGBData, VFClose

VFAppendRGBData

Ein RGB-Datensatz wird in eine Bitmap bzw. eine Folge von Bitmaps konvertiert und diese an eine vorher mit VFOpen() geöffnete Videodatei angehängt.

Deklaration:

```
VFAppendRGBData ( Datei-ID, RGB-Datensatz ) -> Erfolg
```

Parameter:

Datei-ID	ID der geöffneten Videodatei. Entspricht dem Rückgabewert der Funktion VFOpen().
RGB-Datensatz	Datensatz mit RGB-Attribut. Die einzelnen Werte enthalten die Farbinformation für je ein Pixel.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Der übergebene Datensatz muss als Bild interpretierbar sein, wobei jedes Sample die Farbinformation für 1 Pixel im RGB-Format oder als Graustufe enthält. Solche Datensätze sind durch ein spezielles Bild-Attribut gekennzeichnet. Das Bild-Attribut wird standardmäßig nur von Importfiltern für Bilddateien, Spezialfunktionen wie VpGetImages() gesetzt oder explizit durch die Funktion SetFlag() gesetzt.

Der Datensatz darf Events besitzen, wobei jedes Event als 1 Bild interpretiert wird.

Wenn die Größe der resultierenden Bitmap nicht mit der Bildgröße des Videos übereinstimmt, wird die Bitmap zentriert auf den Videobild platziert.

Beispiele:

Ein Panel mit Videoplayer-Widget wird geladen und eine Videodatei angezeigt. Ab Bildnummer 400 werden 200 Bilder ausgeschnitten und in eine neue MP4-Datei gespeichert.

```
PnLoad( "videoplayer.panel")
VpVideoLoad("c:\videos\sample.avi",1)
VpSetPosFrames(400,1)
images = VpGetImages(200)
id = VFOpen( "c:videos\clipped.mp4", 0, 0, 0, 0, 25, 2)
IF id > 0
   VFAppendRGBData(id, images)
   VFClose(Id)
END
```

Siehe auch:

VFOpen, VFAppendPanelSnapshot, VFAppendCwSnapshot, RGB, VpGetImages, Flag?

VFClose

Eine Video-Datei wird geschlossen und der Inhalt auf den Datenträger geschrieben.

Deklaration:

```
VFClose ( Datei-ID ) -> Erfolg
```

Parameter:

Datei-ID	ID der geöffneten Videodatei. Entspricht dem Rückgabewert der Funktion VFOpen(). Durch Angabe von 0 werden alle zur Zeit geöffneten Videodateien geschlossen.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Beispiele:

Ein Panel mit Videoplayer-Widget wird geladen und eine Videodatei angezeigt. Ab Bildnummer 400 werden 200 Bilder ausgeschnitten und in eine neue MP4-Datei gespeichert.

```
PnLoad( "videoplayer.panel")
VpVideoLoad("c:\videos\sample.avi",1)
VpSetPosFrames(400,1)
images = VpGetImages(200)
id = VFOpen( "c:\videos\clipped.mp4", 0, 0, 0, 0, 25, 2)
IF id > 0
   VFAppendRGBData(id, images)
   VFClose(Id)
END
```

Siehe auch:

VFOpen, VFAppendPanelSnapshot, VFAppendCwSnapshot, VFAppendFrame, VFAppendRGBData

VFOpen

Eine Videodatei wird geöffnet und für das nachfolgende Anhängen von Einzelbildern vorbereitet.

Deklaration:

```
VFOpen ( Dateiname, Modus, Format, Breite, Höhe, Bildrate [, Qualität] ) -> Datei-ID
```

Parameter:

Dateiname	Kompletter Pfadname der Videodatei
Modus	
	0 : Die Datei wird neu angelegt. Eine bereit bestehende Datei mit dem selben Namen wird überschrieben.
	1 : Falls die Datei bereits existiert, wird sie zum Anhängen geöffnet. In diesem Fall werden alle Video-Parameter der bestehenden Datei beibehalten und die weiteren Parameter der Funktion ignoriert. Diese müssen aber trotzdem erlaubte Werte haben.
Format	Video-Format
	0 : Automatische Format-Bestimmung am Hand der Datei-Erweiterung. Unterstützte Formate sind ".mp4", ".wmv" und ".avi".
Breite	Breite des Videobildes in Bildpunkten. Muss ein Vielfaches von 2 sein und im Bereich von 100..4096 liegen.
Höhe	Höhe des Videobildes in Bildpunkten. Muss ein Vielfaches von 2 sein und im Bereich von 100..4096 liegen. Breite und Höhe dürfen auch beide 0 sein, in diesem Fall wird die Breite und Höhe des Videos automatisch durch die Abmessung des ersten zugefügten Bildes (siehe Funktionen VFAppend*) bestimmt.
Bildrate	Bildrate in Bilder pro Sekunde (Frames per second, fps).
Qualität	Bestimmt die Qualität (Kompression) des codierten Videos. Kleinere Werte bedeuten höhere Qualität, aber auch höhere Dateigröße. (optional , Standardwert: 3)
	1 :
	2 :
	3 :
	4 :
	5 :
Datei-ID	ID der geöffneten Videodatei (>=1) oder 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die folgenden Video-Formate werden unterstützt.

Dateierweiterung	Format
.mp4	Container: MP4 (MPEG-4 Base Media). Videocodec: MPEG-4 Visual (Advanced Simple@L1).
.avi	Container: AVI (Audio Video Interleaved). Videocodec: MPEG-4 Visual (Simple@L1).
.wmv	Container: WMV (Windows Media). Videocodec: WMV2 (Windows Media Video 8).

Beim Anhängen an eine bestehende Datei ([Modus]=1 ist zu beachten, dass die bestehende Datei zunächst wieder decodiert und in Einzelbilder aufgesplittet werden muss, bevor neue Bilder angehängt werden können. Dies ist rechenintensiv und kostet Speicherplatz. Wann immer möglich, sollten daher Videodateien immer in einem Durchlauf erzeugt werden und zwischenzeitliches Schließen vermieden werden.

Jede mit VFOpen() geöffnete Datei muss durch einen Aufruf von VFClose() wieder geschlossen werden.

Die ID der geöffneten Datei muss bei jedem folgenden Zugriff auf die Datei mit einer Video-Dateifunktion (VF*) angegeben werden. Sie ist gültig bis zum Aufruf von VFClose().

Es können maximal 10 Videodateien gleichzeitig geöffnet werden.

Durch fehlende Aufrufe von VFClose() kann diese Zahl schnell erreicht werden. Sie erhalten dann eine entsprechende Fehlermeldung.

Der Aufruf VFClose(0) schließt alle momentan geöffneten Dateien. Dies kann nützlich sein, wenn z.B. beim Austesten von Sequenzen im Einzelschrittmodus Dateien offen geblieben sind, deren ID's Sie nicht mehr kennen.

Alle offenen Videodateien werden außerdem automatisch geschlossen beim Menübefehl 'Datei'/'Neubeginn' und beim manuellen Neustart einer Sequenz im Editorfenster (z.B. mit Tastenkombination F7).

Multithreading: Die von der Funktion zurückgegebene Datei-ID ist nur für den aktuellen Ausführungs-Thread gültig. Dateien werden am Thread-Ende (also z.B. am Ende einer mittels BEGIN_PARALLEL ausgeführten Sequenzfunktion) automatisch geschlossen.

Beispiele:

In einem Verzeichnis liegen diverse Messdateien-Dateien als Ergebnis eines Versuchs. Alle Dateien, die einem gegebenen Namensmuster entsprechen, werden nacheinander in ihrer Erzeugungsreihenfolge geladen, in einem Kurvenfenster mit fester Konfiguration angezeigt und die Kurvenbilder zu einem Video zusammengefügt. Die Videobildgröße entspricht der Größe des Kurvenbildes auf dem Schirm, das Videoformat ist MP4

in Standardqualität mit 2 Bildern pro Sekunde.

```
CwSelectMode("variable")
id = VFOpen( "c:\video\test_2019_04_18.mp4", 0, 0, 0, 0, 2)
IF id > 0
   CwLoadCCV(id, "data.ccv") ;curve configuration with fixed channel name "data"
   CwSelectWindow(id)
   fileNames = FsGetFileNames("c:\test\2019_04_18","ch*.raw", 0, 0, 2)
   FOREACH ELEMENT fileName in fileNames
      fh = FileOpenDSF(fileName, 0)
      IF fh > 0
         data = FileObjRead(fh, 1)
         FileClose(fh)
          VFAppendCwSnapshot(id )
      END
   END
   VFClose(id)
END
```

Ein Panel hat einen Knopf "Take Snapshot".Mit jeder Betätigung des Knopfes wird ein Abbild des aktuellen Inhaltes der ersten Panelseite an eine beim Laden des Panels geöffnete Video-Datei angehängt. Mit dem Schließen des Panels wird auch die Video-Datei geschlossen.

Ereignis-Sequenz 'Init':

```
; Öffnen der Video-Datei im WMV-Format mit 2 fps. Videobildgröße entspricht der Größe des ersten zugefügten Bildes.
videoID = VFOpen( "d:\evaluation.wmv", 0, 0, 0, 0, 2)
```

Ereignis-Sequenz 'Knopf gedrückt' für den 'Take Snapshot'-Knopf:

```
; Anhängen des aktuellen Panel-Bildes (Seite 1) in Originalgröße
VFAppendPanelSnapshot( videoID, 1, 0)
```

Ereignis-Sequenz 'Ende':

```
VFClose(videoID)
```

Siehe auch:

VFClose, VFAppendPanelSnapshot, VFAppendCwSnapshot, VFAppendRGBData, VFAppendFrame

VibrationFilter

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Filterung für die Bewertung von Schwingungen. Die Funktion führt die Filterung aus entsprechend einer gewünschten Frequenzbewertung. Danach wird das Ergebnis einer gleitenden exponentiellen Effektivwertbildung unterzogen (Zeitbewertung). Danach erfolgt eine Nachabtastung, die Datenmenge wird dabei um einen Faktor reduziert.

Deklaration:

```
VibrationFilter ( Eingangskanal, Frequenz-Bewertung, Zeitkonstante, Reduktion ) -> Ergebnis
```

Parameter:

Eingangskanal	Der zu filternde Zeitdatensatz, die Zeit in Sekunden skaliert.
Frequenz-Bewertung	Welche Frequenz-Bewertung wird benutzt?
	10 : Wk, z-Richtung und senkrechte Richtung liegend, außer Kopf (z direction and for vertical recumbent direction, except head). Nach ISO 2631-1:1997
	11 : Wd, x- und y-Richtung und horizontale Richtung liegend (x and y directions and for horizontal recumbent direction). Nach ISO 2631-1:1997
	12 : Wf, Bewegungskrankheit (Motion sickness). Nach ISO 2631-1:1997
	13 : Wc, Messung am Rücken (seat-back measurement). Nach ISO 2631-1:1997
	14 : We, Rotierende Schwingungen (measurement of rotational vibration). Nach ISO 2631-1:1997
	15 : Wj, Messung unter Kopf, liegend (vibration under the head of a recumbent person). Nach ISO 2631-1:1997
	16 : Hx, Ganzkörperschwingungen, stehende, sitzende Haltung, Messrichtung x, y. Liegende Haltung, Messrichtung y, z. Nach DIN 45671-1:1990-09
	17 : Hz, Ganzkörperschwingungen, stehende, sitzende Haltung, Messrichtung z. Nach DIN 45671-1:1990-09
	18 : Hxl, Ganzkörperschwingungen, liegende Haltung, Messrichtung x. Nach DIN 45671-1:1990-09
	19 : Hb, Ganzkörperschwingungen, nicht vorgegebene Körperhaltung. Nach DIN 45671-1:1990-09
	20 : Hh, Hand-Arm-Schwingungen, für alle Messrichtungen. Nach DIN 45671-1:1990-09
	20 : Handübertragene Vibration, Gewichtungsfilter. Nach ISO 7505:1986-05
	21 : Gewichtungsfaktoren für Querschwingungen (x, y), gemäß zurückgezogener ISO 2631-1:1985 Tabelle 3
	22 : Gewichtungsfaktoren für Längsschwingungen (z), gemäß zurückgezogener ISO 2631-1:1985 Tabelle 3
	23 : Wb (Fahrgast und Besatzungskomfort in Beförderungssystemen mit fester Führung). Nach ISO 2631-4:2001
	24 : Wm (Exposition des Menschen gegenüber Vibrationen in Gebäuden). Nach ISO 2631-2:2003
	25 : Beschleunigungseingang. Nach ISO 6954:2000
	26 : Drehzahleingang. Nach ISO 6954:2000
	27 : Wh (handübertragene Vibration, Belastungsfilter). Nach ISO 5349-1:2001
	28 : Wb (Fahrgast und Besatzungskomfort in Beförderungssystemen mit fester Führung). Nach ISO 8041:2005
	29 : Wc (Sitz-Rücken-Messung). Nach ISO 8041:2005
	30 : Wd (x- und y-Richtung und für horizontale Liegerichtung). Nach ISO 8041:2005
	31 : We (Messung der Drehschwingungen). Nach ISO 8041:2005
	32 : Wf (Ganzkörper tieffrequent, Bewegungskrankheit, Motion sickness). Nach ISO 8041:2005
	33 : Wh (von Hand übertragene Vibration). Nach ISO 8041:2005
	34 : Wj (Vibration unter dem Kopf einer liegenden Person). Nach ISO 8041:2005
	35 : Wk (z-Richtung und für vertikale Liegerichtung, außer Kopf). Nach ISO 8041:2005
	36 : Wm (Exposition des Menschen gegenüber Vibrationen in Gebäuden). Nach ISO 8041:2005
	37 : Wb (Bahnanwendungen, Fahrkomfort für Fahrgäste, Z Boden, Z Sitzschale). Nach EN 12299:2009
	38 : Wc (Bahnanwendungen, Fahrkomfort für Fahrgäste, X Sitzrücklehne). Nach EN 12299:2009
	39 : Wd (Bahnanwendungen, Fahrkomfort für Fahrgäste, X Boden, Y Boden, Y Sitzschale). Nach EN 12299:2009

	40 : Wp (Bahnanwendungen, Fahrkomfort für Fahrgäste, Y Boden, phi Boden). Nach EN 12299:2009
Zeitkonstante	Die Zeitkonstante für die exponentielle Effektivwertbildung. In Sekunden angegeben, >= 0.0. Z.B. 1.0s für SLOW Bewertung, 0.125s für FAST Bewertung. Falls = 0, dann keine Effektivwertbildung. Dann wird lediglich das gefilterte Signal bestimmt.
Reduktion	>=1, Nur jeder soundsovielte Wert des Ergebnisses wird ausgegeben. Damit kann bei großer Zeitkonstante die Datenmenge für das Resultat sinnvoll eingeschränkt werden.
Ergebnis	Gefilterter Datensatz

Beschreibung:

Die Abtastzeit des Eingangskanals muss klein genug sein, sodass die entscheidenden Knicke der Frequenzgänge realisiert werden können.

Beispiele:

```
wc = VibrationFilter ( Vibration, 10, 0.125, 20 )
```

Wk Bewertung nach ISO 2631. Gleitender Effektivwert mit Zeitkonstante 0.125 (FAST). Das Eingangssignal Vibration ist ein Beschleunigung und ist mit 1 ms abgetastet. Jeder 20. Wert wird nur für das Ergebnis benutzt.

```
a = diff ( v )
hx = VibrationFilter ( a, 16, 0, 1 )
```

eine Hx Bewertung nach DIN 45671-1:1990-09 wird durchgeführt. Da eine Beschleunigung benutzt werden soll, aber eine Geschwindigkeit gemessen wurde, wird vorher einmal abgeleitet. Nur die Filterung wird durchgeführt ohne Bestimmung eines Effektivwertes.

Siehe auch:

FilterAnalog, FiltTP, dFilt, ExpoRms

VpBackStep

Verfügbar ab: Professional Edition (Video-Kit)

Zeigt das Bild an, dem das gegenwärtige Bild folgt.

Deklaration:

```
VpBackStep ( Player ) -> Fehlercode
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das letzte Bild der Video-Datei wird als Standbild angezeigt. Ist die Datei am Anfang angelangt, gibt die Funktion einen Fehler zurück.

Beispiele:

Im folgenden Beispiel wird eine Datei namens BEISPIEL.AVI im Pluginfenster geladen. Es wird das zweite Bild angezeigt, und nach einer Pause von 10 Sekunden das erste.

```
VpVideoLoad ("C:\VIDEO\BEISPIEL.AVI", 0)
VpSingleStep(0)
sleep(10)
VpBackStep(0)
```

Siehe auch:

VpSingleStep

VpContinue

Verfügbar ab: Professional Edition (Video-Kit)

Setzt das Abspielen der Video-Datei an der aktuellen Position fort.

Deklaration:

```
VpContinue ( Player ) -> Fehlercode
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Wenn das Video mit Hilfe des Pause-Schalters oder der Kit-Funktion VpPause() angehalten wurde, kann das Abspielen mit der Funktion VpContinue() fortgesetzt werden. Mehrfaches Ausführen dieser Funktion führt nicht zum Anhalten des Videos, es wird auch kein Fehlercode < 0 ausgegeben. Wurde das Video nicht aus dem Abspielen heraus angehalten, sondern befindet sich im zurückgesetzten Zustand (direkt nach dem Laden oder nach Stop), hat ein Aufruf von VpContinue() keine Wirkung.

Beispiele:

Im folgenden Beispiel wird angenommen, daß sich die Datei BEISPIEL.AVI im Verzeichnis C:\VIDEO befindet. Diese Datei wird geladen, 10 Sekunden lang abgespielt, pausiert dann 10 Sekunden, und setzt danach das Abspielen fort.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpPlay(0)
sleep(10)
VpPause(0)
sleep(10)
VpContinue(0)
```

Das Video bleibt 10 Sekunden bei dem Bild stehen, das nach 10 Sekunden erreicht war, und wird dann weiter abgespielt.

Siehe auch:

VpPause

VpDelLink

Verfügbar ab: Professional Edition (Video-Kit)

Löscht die Verbindung mit einem Kurvenfenster.

Deklaration:

```
VpDelLink ( Player ) -> Fehlercode
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll. Reserviert, immer auf Null zu setzen. Die Funktion ist nur auf den Videoplayer im FAMOS-Plugin-Fenster anwendbar.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Ist eine Verbindung mit einem Kurvenfenster oder dem Dateneditor vorhanden, wird diese gelöscht. Der Zustand der Video-Players wird davon nicht beeinflußt.

Beispiele:

Im folgenden Beispiel wird die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und eine Verbindung zu einem Kurvenfenster aufgebaut, das den Zeitverlauf der Variablen test darstellt. Das Video wird 10 Sekunden lang abgespielt, dann wird die Verbindung gelöst.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
id=FileOpenDSF("TEST.DAT", 0)
test = FileObjRead(id, 1)
KvKonfig(test, "TEST.CCV")
VpSetLink(test, 0)
VpPlay(0)
sleep(10)
VpDelLink(0)
```

Siehe auch:

VpSetLink, VpLinkExists

VpGetAbsStartTime

Verfügbar ab: Professional Edition (Video-Kit)

Gibt Datum und Uhrzeit der Erstellung der aktuell geladenen Video-Datei an.

Deklaration:

```
VpGetAbsStartTime ( Player ) -> Zeitpunkt
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll. Reserviert, immer auf Null zu setzen. Die Funktion ist nur auf den Videoplayer im FAMOS-Plugin-Fenster anwendbar.
Zeitpunkt	Datum und Uhrzeit der Erstellung.

Beschreibung:

Benutzt man in einem verbundenen Kurvenfenster absolute Zeitangaben, ist es notwendig, die absolute Startzeit der Videoaufzeichnung zu berücksichtigen. Standardmäßig wird dafür der Erstellungszeitpunkt des Videos benutzt. Die Funktion VpGetAbsStartTime() gibt den eingestellten absoluten Startzeitpunkt zurück.

Die absolute Startzeit wird als Text der Form "tt.mm.jj hh:mm:ss" (Tag.Monat.Jahr Stunden:Minuten:Sekunden) angegeben.

Die Funktion hat keine Fehlercodes. Tritt ein Fehler auf, wird ein leerer Text zurückgegeben.

Beispiele:

Im folgenden Beispiel wird die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und ihr Erstellungszeitpunkt in der Variablen build abgelegt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
build=VpGetAbsStartTime(0)
```

Siehe auch:

VpSetAbsStartTime, VpGetXOffset, VpSetXOffset

VpGetAbsStartTime2

Verfügbar ab: Professional Edition (Video-Kit)

Gibt Datum und Uhrzeit der Erstellung der aktuell geladenen Video-Datei an.

Deklaration:

```
VpGetAbsStartTime2 ( Player ) -> Zeitpunkt
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll. Reserviert, immer auf 1 zu setzen. Die Funktion ist nur auf den ausgewählten Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers anwendbar. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Zeitpunkt	Zeitpunkt der Erstellung (FAMOS-Zeitformat).

Beschreibung:

Benutzt man in einem verbundenen Kurvenfenster absolute Zeitangaben, ist es notwendig, die absolute Startzeit der Videoaufzeichnung zu berücksichtigen. Standardmäßig wird dafür der Erstellungszeitpunkt des Videos benutzt. Die Funktion VpGetAbsStartTime2() gibt den eingestellten absoluten Startzeitpunkt zurück.

Der Rückgabewert im FAMOS-Zeitformat kann mit den Funktionen der Gruppe '18> Datum, Uhrzeit' weiter verarbeitet werden.

Tritt ein Fehler auf, wird ein Fehlercode (immer negativ) zurückgegeben.

Sowohl das Videoplayer-Anzeigeelement als auch eine geladene Videodatei können jeweils eine explizit zugewiesene Startzeit besitzen. Die Startzeit der Videodatei kann je nach Typ entweder im Dateikopf oder in einer parallelen Konfigurationsdatei (*.ivi) eingetragen sein, entweder direkt bei der Aufnahme oder nachträglich durch den Anwender (Dialog 'Video-Eigenschaften'). Am Videoplayer-Element selbst gibt es ebenfalls eine entsprechende Eigenschaft, die beim Design festgelegt werden kann.

Bei der Bestimmung der wirksamen Startzeit ist die Einstellung am Anzeigeelement dominant, nur wenn diese auf 'automatisch' steht, wird ggf. die private Startzeit des angezeigten Videos berücksichtigt. Ist auch diese nicht vorhanden, ist die resultierende Startzeit gleich der Datei-Erstellungszeit der Videodatei.

Beispiele:

Im folgenden Beispiel wird die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und ihr Erstellungszeitpunkt abgefragt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 1)
StartTime = VpGetAbsStartTime2(1)
TxStartTime = ZeitInText(StartTime, 0)
```

Siehe auch:

VpSetAbsStartTime2, VpGetXOffset, VpSetXOffset

VpGetErrorText

Verfügbar ab: Professional Edition (Video-Kit)

Gibt zum übergebenen Fehlercode den zugehörigen Fehlertext an.

Deklaration:

```
VpGetErrorText ( Fehlercode ) -> Fehlertext
```

Parameter:

Fehlercode	Von den Funktionen des Video-Kits im Fehlerfall ausgegebene Nummer.
Fehlertext	Fehlertext zum Fehlercode.

Beschreibung:

Viele Video-Player-Funktionen liefern als Rückgabewert eine Zahl, die im Fehlerfall Auskunft über die Fehlerursache gibt. Bei Erfolg der Funktion wird 0 zurückgegeben. Zu jedem Fehlercode gibt es einen Text, der die Fehlerursache benennt. Dieser Text kann mit VpGetErrorText() erhalten werden.

Die Angabe des negativen Vorzeichens des Fehlercodes ist für diese Funktion nicht notwendig.

Beispiele:

Folgende Sequenz ist fehlerhaft, da versucht wird, an eine zu hohe Position zu springen. Es wird hier angenommen, daß die Video-Datei BEISPIEL.AVI im Verzeichnis C:\VIDEO liegt.

```
VpVideoLoad( "C:\VIDEO\BEISPIEL.AVI", 0)
erg  = VpGetLengthFrames(0)
erg = erg + 1
num = VpSetPosFrames(erg, 0)
txt = VpGetErrorText(num)
```

Diese Sequenz setzt txt = "Kann nicht zur angegebenen Position springen".

Siehe auch:

VpGetFileName

Verfügbar ab: Professional Edition (Video-Kit)

Gibt den Namen und Pfad der aktuell geladenen Video-Datei an.

Deklaration:

```
VpGetFileName ( Player ) -> Name
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Name	Name und Pfad der aktuellen Datei

Beschreibung:

Die Funktion VpGetFileName() gibt den Namen und Pfad der aktuell geladenen Video-Datei zurück.

Diese Funktion hat keine Fehlercodes. Tritt ein Fehler auf, wird ein leerer Text zurückgegeben.

Beispiele:

Im folgenden Beispiel wird davon ausgegangen, daß vor dem Ausführen der Sequenz die Videodatei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO von Hand geladen wurde. Nach dem Ausführen der Sequenz enthält die Variable name den Text "C:\VIDEO\BEISPIEL.AVI".

```
name = VpGetFileName(0)
```

Siehe auch:

VpGetImages

Verfügbar ab: Professional Edition (Video-Kit)

Holt ein oder mehrere Bilder aus der aktuell angezeigten Video-Datei im aktiven Panel-Videoplayer.

Deklaration:

```
VpGetImages ( Bildanzahl ) -> Bilddaten
```

Parameter:

Bildanzahl	Gibt die Anzahl der Bilder an, welche aus der aktuell geladenen Video-Datei geholt werden sollen.
Bilddaten	Datensatz mit den angeforderten Bilddaten.

Beschreibung:

Die Funktion exportiert fortlaufend von der aktuellen Position die mit dem Parameter "Bildanzahl" angegebene Menge an Videobildern. Wird das Video zum Zeitpunkt des Aufrufes abgespielt, so wird es ohne Rückmeldung pausiert und auch nicht wieder fortgesetzt.

Der Ergebnisdatensatz besitzt das Dateiformat "4 Byte ganzzahlig ohne Vorzeichen". Jeder Bildpunkt wird durch einen 4-Byte-Wert repräsentiert, dabei wird das "RGB" Format verwendet. Das höchstwertige Byte ist immer 0, danach kommen die Werte für die Farbanteile Blau, Grün und Rot im Wertebereich 0 ... 255. Ein 4-Byte-Wert von 0x00000000 entspricht schwarz, ein Wert von 0x00FFFFFF (dezimal => 16777215) entspricht weiß. Der Ergebnissdatensatz ist segmentiert, die Segmentlänge entspricht der Breite eines Videobildes in Pixeln. Die Segmentzahl pro Bild entspricht der Höhe eines Videobildes in Pixeln. Das erste Segment entpricht der untersten Bildzeile. Das erste Sample des ersten Segments entspricht dem Bildpunkt links unten.

Wird nur ein Bild exportiert, so wird ein einfacher Datensatz angelegt, ansonsten wird ein eventierter Datensatz erzeugt in dem jedes Event ein Bild repräsentiert. Werden mehr zu exportierende Bilder angegeben als ab der aktuellen Position noch Bilder im Video vorhanden sind, so wird ohne Rückmeldung nur die noch verfügbare Menge an Bildern exportiert. Die Gesamtlänge des Datensatzes entpricht somit der Bildbreite x Höhe (Pixel) x Anzahl der gelieferten Frames.

Im Fehlerfall wird ein Datensatz der Länge 1 geliefert, der den Fehlercode enthält. Der zugehörige Fehlertext kann mit der Funktion "VpGetErrorText" ermittelt werden.

Die Funktion ist nur auf den ausgewählten Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers anwendbar. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.

Beispiele:

Im folgenden Beispiel wird die Datei BEISPIEL.AVI in den aktuellen Panel-Videoplayer geladen. VpSetPosFrames() setzt das Video an die Position, die der übergebenen Bildnummer (100) entspricht. Anschließend werden fünf Bilder aus der aktuell im Video-Player geladen Video-Datei geholt. Das erste der fünf zu holenden Bilder wird ab dem aktuellen Zeitpunkt geholt. Zum Schluß werden noch die einzelnen Farbanteile aus dem ersten Pixel (links, unten) des ersten Bildes extrahiert.

```
VpVideoLoad("c:\BEISPIEL.AVI",1)
VpSetPosFrames(100,1)
Bilder = VpGetImages(5)
Bild1 = Bilder[1]
Zeile1 = Bild1[1]
Pixel1 = Zeile1[1]
RedValue = BitAnd(Pixel1, 0x000000FF, 32)
GreenValue = BitShift(BitAnd(Pixel1, 0x0000FF00, 32), -8, 32)
BlueValue = BitShift(BitAnd(Pixel1, 0x00FF0000, 32), -16, 32)
```

Siehe auch:

RGB

VpGetLengthFrames

Verfügbar ab: Professional Edition (Video-Kit)

Gibt die Anzahl der Bilder in der aktuell geladenen Video-Datei an.

Deklaration:

```
VpGetLengthFrames ( Player ) -> Länge
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Länge	Länge des Videos in Bildern
	>= 0 : Nummer des Bildes
	< 0 : Fehlercode

Beschreibung:

Die Funktion VpGetLengthFrames() gibt die Anzahl der Bilder in der aktuell geladenen Video-Datei zurück.

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:VIDEO wird geladen. Da die Zählung der Bilder bei 1 beginnt, kann man mit VpGetLengthFrames() das letzte Bild ermitteln und anzeigen lassen.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
last = VpGetLengthFrames(0)
VpSetPosFrames(last, 0)
```

Siehe auch:

VpGetLengthSeconds

VpGetLengthSeconds

Verfügbar ab: Professional Edition (Video-Kit)

Gibt die Länge der aktuell geladenen Video-Datei in Sekunden an.

Deklaration:

```
VpGetLengthSeconds ( Player ) -> Länge
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Länge	Länge des Videos in Sekunden
	>= 0 : Länge des Videos
	< 0 : Fehlercode

Beschreibung:

Die Funktion VpGetLengthSeconds() gibt die Länge der aktuell geladenen Video-Datei in Sekunden an.

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO wird geladen, ihre Länge in Sekunden wird ermittelt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
length = VpGetLengthSeconds(0)
```

Siehe auch:

VpGetLengthFrames

VpGetPlayRate

Verfügbar ab: Professional Edition (Video-Kit)

Gibt die Abspielgeschwindigkeit des gegenwärtig geladenen Videos an.

Deklaration:

```
VpGetPlayRate ( Player ) -> Abspielgeschwindigkeit
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Abspielgeschwindigkeit	Abspielgeschwindigkeit des Videos.
	>= 0 : Abspielgeschwindigkeit
	< 0 : Fehlercode

Beschreibung:

Das gegenwärtig geladene Video wird mit voreingestellter Geschwindigkeit abgespielt. Diese Geschwindigkeit wird in Bildern pro Sekunde angegeben (frames per second, FPS); sie ist in der Videodatei gespeichert. Die Abspielgeschwindigkeit läßt sich vorübergehend ändern; siehe VpSetPlayRate(). Die Funktion VpGetPlayRate() gibt die gegenwärtig eingestellte Abspielgeschwindigkeit zurück.

Beispiele:

Im folgenden Beispiel wird das Video BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und seine standardmäßig eingestellte Abspielgeschwindigkeit ermittelt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
rate = VpGetPlayRate(0)
```

Siehe auch:

VpSetPlayRate, VpGetRecordRate, VpSetRecordRate

VpGetPosFrames

Verfügbar ab: Professional Edition (Video-Kit)

Gibt die Nummer des gegenwärtig angezeigten Bildes an.

Deklaration:

```
VpGetPosFrames ( Player ) -> Bildnummer
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Bildnummer	Nummer des aktuellen Bildes.
	>= 0 : Nummer des Bildes
	< 0 : Fehlercode

Beschreibung:

Videos sind als (komprimierte) Einzelbilder in einer festen Reihenfolge gespeichert. Im Video-Player beginnt die Zählung mit 1; Bild 1 ist also das aktuelle Bild, wenn die Position (in Sekunden) im Bereich von 0.0000s bis (Länge des Videos in Sekunden / Anzahl der Bilder) liegt. VpGetPosFrames() gibt die Nummer des aktuellen Bildes zurück.

Beispiele:

Im folgenden Beispiel wird das Video BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO im FAMOS Pluginfenster geladen, 10s lang abgespielt und dann die zu diesem Zeitpunkt aktuelle Bildnummer ermittelt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpPlay(0)
sleep(10)
position = VpGetPosFrames(0)
```

Nimmt man an, daß die Abspielgeschwindigkeit 10 Bilder pro Sekunde (10 FPS) betrug, hat die Variable position nach dem Ausführen der Sequenz den Wert 101.

Siehe auch:

VpSetPosFrames, VpGetPosSeconds, VpSetPosSeconds

VpGetPosSeconds

Verfügbar ab: Professional Edition (Video-Kit)

Gibt die Position des aktuell angezeigten Bildes in Sekunden an.

Deklaration:

```
VpGetPosSeconds ( Player ) -> Position
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Position	Position des aktuellen Bildes in Sekunden.

Beschreibung:

Beim Aufnehmen eines Videos werden die Einzelbilder mit einer bestimmten Aufnahmegeschwindigkeit gespeichert. Die Position eines Bildes in Sekunden ist dann die Bildnummer dividiert durch die Aufnahmegeschwindigkeit. Die Funktion VpGetPosSeconds() gibt die Position des aktuellen Bildes in Sekunden zurück.

Die Funktion hat keine Fehlercodes. Tritt ein Fehler auf, wird 0.000 zurückgegeben.

Beispiele:

Im folgenden Beispiel wird das Video BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen, bis zum Bild mit der Nummer 20 abgespielt und dann die Position dieses Bildes in Sekunden ermittelt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpPlaySync(20, 0)
position = VpGetPosSeconds(0)
```

Nimmt man an, daß die Aufnahmegeschwindigkeit 10 Bilder pro Sekunde (10 FPS) betrug, hat die Variable position nach dem Ausführen der Sequenz den Wert 1.9.

Siehe auch:

VpSetPosSeconds, VpGetPosFrames, VpSetPosFrames

VpGetRecordRate

Verfügbar ab: Professional Edition (Video-Kit)

Gibt die Aufnahmegeschwindigkeit der aktuell geladenen Video-Datei in Bildern pro Sekunde an.

Deklaration:

```
VpGetRecordRate ( Player ) -> Rate (FPS)
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Rate (FPS)	Aufnahmegeschwindigkeit in Bildern pro Sekunde
	>= 0 : Aufnahmegeschwindigkei
	< 0 : Fehlercode

Beschreibung:

Die Videodatei besteht aus (komprimierten) Einzelbildern, die in einer bestimmten Geschwindigkeit aufgenommen wurden. Diese wird in Bildern pro Sekunde (frames per second, FPS) angegeben. Standardmäßig wird angenommen, daß Aufnahmegeschwindigkeit und Abspielgeschwindigkeit gleich sind. Sie können jedoch auch verschiedene Werte haben, z.B. wenn Hochgeschwindigkeitsaufnahmen in .AVI-Dateien umgewandelt werden. Mit Hilfe der Funktion VpGetRecordRate() kann die gegenwärtig eingestellte Aufnahmegeschwindigkeit ermittelt werden.

Videoplayer-Element im Datenbrowser:

Sowohl das Videoplayer-Anzeigeelement als auch eine geladene Videodatei können jeweils eine explizit zugewiesene Aufnahmerate besitzen. Der Aufnahmerate der Videodatei kann je nach Typ entweder im Dateikopf oder in einer parallelen Konfigurationsdatei (*.ivi) eingetragen sein, entweder direkt bei der Aufnahme oder nachträglich durch den Anwender (Dialog 'Video-Eigenschaften'). Am Videoplayer-Element selbst gibt es ebenfalls eine entsprechende Eigenschaft, die beim Design festgelegt werden kann.

Bei der Bestimmung der wirksamen Rate ist die Einstellung am Anzeigeelement dominant, nur wenn diese auf 'automatisch' steht, wird ggf. die private Aufnahmerate des angezeigten Videos berücksichtigt. Ist auch diese nicht vorhanden, wird die originale Rate der Videoaufnahme verwendet.

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO wird geladen, ihre Aufnahmegeschwindigkeit wird ermittelt. Nach dem Ausführen der Sequenz steht in der Variablen rate der Wert, der standardmäßig für die Wiedergabe vorgesehen ist.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
rate = VpGetRecordRate(0)
```

Siehe auch:

VpSetRecordRate, VpGetPlayRate, VpSetPlayRate

VpGetState

Verfügbar ab: Professional Edition (Video-Kit)

Gibt eine Zahl zurück, die den Zustand des Video-Players beschreibt.

Deklaration:

```
VpGetState ( Player ) -> Zustand
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Zustand	Zahl, die den Zustand des Video-Players beschreibt.

Beschreibung:

Der Video-Player kann durch Kit-Funktionen oder die Bedienung der Steuerelemente seinen Zustand ändern. Die Funktion VpGetState() gibt eine Zahl zurück, die den gegenwärtigen Zustand beschreibt. Folgende Werte sind möglich:

0: Es ist kein Video geladen.

1: Der Video-Player ist zurückgesetzt, das Video ist in Anfangsposition (neu geladen oder Stop).

3: Das Video wird abgespielt (Play).

5: Das Video wurde angehalten, aber vorher nicht abgespielt. (Dieser Zustand tritt ein, wenn die Bildposition nach dem Laden oder Stop von Hand verstellt wurde).

7: Das Video wurde angehalten (Pause).

Beispiele:

Mit Hilfe der Funktion VpGetState() wird überprüft, ob im Pluginfenster ein Video geladen ist. Ist das der Fall, wird es geschlossen und danach BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und abgespielt.

```
erg = VpGetState(0)
WENN erg > 0
   VpVideoClose(0)
ENDE
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpPlay(0)
```

Bemerkung: Dieses Vorgehen ist nicht notwendig; siehe VpVideoLoad().

Siehe auch:

VpGetStateText

VpGetStateText

Verfügbar ab: Professional Edition (Video-Kit)

Gibt einen Text zurück, der den gegenwärtigen Zustand des Video-Players beschreibt.

Deklaration:

```
VpGetStateText ( Player ) -> Zustand
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Zustand	Zustand des Video-Players

Beschreibung:

Der Video-Player kann durch Kit-Funktionen oder die Bedienung der Steuerelemente seinen Zustand ändern. Die Funktion VpGetStateText() gibt einen Text zurück, der den gegenwärtigen Zustand beschreibt. Folgende Werte der Textvariablen sind möglich:

"Kein Video geladen."

"Video ist zurückgesetzt (Neu geladen oder Stop)."

"Video wird abgespielt (Play)."

"Video ist zurückgesetzt und angehalten (Neu geladen/Stop sowie Pause)."

"Video wurde angehalten (Pause)."

Die Funktion hat keine Fehlercodes. Tritt ein Fehler auf, wird ein leerer Text zurückgegeben.

Beispiele:

Das Video BEISPIEL.AVI im Verzeichnis C:\VIDEO wird im Pluginfenster geladen und 30 Sekunden lang abgespielt. Dann wird mit VpGetStateText() überprüft, ob das Video noch gespielt wird oder nicht.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpPlay(0)
sleep(30)
erg = VpGetStateText(0)
```

Abhängig von der Länge des Videos enthält die Variable erg den Text "Video wird abgespielt (Play)." oder "Video wurde angehalten (Pause).".

Siehe auch:

VpGetState

VpGetXOffset

Verfügbar ab: Professional Edition (Video-Kit)

Gibt den zeitlichen Versatz der aktuell geladenen Video-Datei in Sekunden an.

Deklaration:

```
VpGetXOffset ( Player ) -> Zeitlicher Versatz
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Zeitlicher Versatz	Zeitlicher Versatz des Videos in Sekunden.

Beschreibung:

Standardmäßig beginnt die Wiedergabe einer Videodatei zum Zeitpunkt 0.000s. Die Funktion VpGetXOffset() gibt die relative Startzeit des Videos an. Diese relative Startzeit bestimmt die Position (in Sekunden), die für das erste Bild des Videos angezeigt wird.

Die Funktion hat keine Fehlercodes. Tritt ein Fehler auf, wird der Wert 0.000 zurückgegeben.

Videoplayer-Element im Datenbrowser:

Sowohl das Videoplayer-Anzeigeelement als auch eine geladene Videodatei können jeweils einen explizit zugewiesenen Offset besitzen. Der Offset der Videodatei kann je nach Typ entweder im Dateikopf oder in einer parallelen Konfigurationsdatei (*.ivi) eingetragen sein, entweder direkt bei der Aufnahme oder nachträglich durch den Anwender (Dialog 'Video-Eigenschaften'). Am Videoplayer-Element selbst gibt es ebenfalls eine entsprechende Eigenschaft, die beim Design festgelegt werden kann.

Bei der Bestimmung des wirksamen Offsets ist die Einstellung am Anzeigeelement dominant, nur wenn diese auf 'automatisch' steht, wird ggf. der private Offset des angezeigten Videos berücksichtigt. Ist auch dieser nicht vorhanden, ist der resultierende Offset gleich 0.

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO wird geladen, ihre relative Startzeit steht nach dem Ausführen der Sequenz in der Variablen xoff.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
xoff=VpGetXOffset(0)
```

Siehe auch:

VpSetXOffset, VpGetAbsStartTime, VpSetAbsStartTime

VpLinkExists

Verfügbar ab: Professional Edition (Video-Kit)

Gibt an, ob eine Verbindung mit einem Kurvenfenster oder dem Famos-Dateneditor besteht.

Deklaration:

```
VpLinkExists ( Player ) -> Wert
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll. Reserviert, immer auf Null zu setzen. Die Funktion ist nur auf den Videoplayer im FAMOS-Plugin-Fenster anwendbar.
Wert	1, wenn Verbindung existiert, 0 sonst.

Beschreibung:

Gibt an, ob eine Verbindung mit einem Kurvenfenster oder dem Famos-Dateneditor besteht. Ist eine Verbindung vorhanden, gibt die Funktion VpLinkExists() den Wert 1 zurück, sonst den Wert 0.

Die Funktion hat keine Fehlercodes. Bei Auftreten eines Fehlers gibt sie 0 zurück.

Beispiele:

Im folgenden Beispiel wird geprüft, ob eine Verbindung mit einem Kurvenfenster besteht; in diesem Fall wird sie gelöst.

```
link = VpLinkExists(0)
WENN link > 0
   VpDelLink(0)
ENDE
```

Siehe auch:

VpSetLink, VpDelLink

VpPause

Verfügbar ab: Professional Edition (Video-Kit)

Hält die Video-Datei an der aktuellen Position an.

Deklaration:

```
VpPause ( Player ) -> Fehlercode
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das Abspielen der Video-Datei wird an der aktuellen Position unterbrochen, das aktuelle Bild wird als Standbild angezeigt. Im Gegensatz zur Tastenbedienung führt mehrfaches Ausführen der Funktion VpPause() nicht zu einem Weiterspielen des Videos; auch ein Fehlercode < 0 wird nicht ausgegeben.

Beispiele:

Im folgenden Beispiel wird ein Panel mit Videoplayer geladen, das Video BEISPIEL.AVI geladen, 10 Sekunden lang abgespielt und dann pausiert.

```
DbLoadPanel(" c:\VIDEO\VIDEO.PANEL", 0)
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 1)
VpPlay(1)
sleep(10)
VpPause(1)
```

Das Video bleibt bei dem Bild stehen, das nach 10 Sekunden erreicht war.

Siehe auch:

VpContinue

VpPlay

Verfügbar ab: Professional Edition (Video-Kit)

Spielt die geöffnete Video-Datei ab.

Deklaration:

```
VpPlay ( Player ) -> Fehlercode
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Die zur Zeit geladene Video-Datei wird abgespielt. Dabei wird die Einstellung für die Abspielgeschwindigkeit berücksichtigt. Die Einstellungen für den Startzeitpunkt und die Aufnahmegeschwindigkeit gehen in die Anzeige der Position (in Sekunden) ein. Das Abspielen erfolgt asynchron, d.h. auch während des Spielens ist es möglich, das Programm zu bedienen.

Das Verhalten der Funktion ist abhängig vom gewählten Player. Beim Plugin wird die Wiedergabe ab der aktuellen Position fortgesetzt. Beim Player im Panel erfolgt die Wiedergabe stets ab Beginn des Videos, verwenden Sie hier VpContinue(), um ab der aktuellen Position abzuspielen.

Hinweis: Ein Aufruf der Funktion VpPlay(0) entspricht einem Betätigen der Play-Taste.

Beispiele:

Im folgenden Beispiel wird eine Datei namens BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und mit der voreingestellten Abspielgeschwindigkeit ab dem ersten Bild abgespielt.

```
VpVideoLoad ("C:\VIDEO\BEISPIEL.AVI", 0)
VpPlay(0)
```

Im nächsten Beispiel wird ein Panel mit Videoplayer im Datenbrowser geladen, das angegebene Video geladen und die Wiedergabe gestartet.

```
DbLoadPanel(" c:\VIDEO\VIDEO.PANEL", 0)
VpVideoLoad ( "C:\VIDEO\BEISPIEL.AVI", 1)
VpPlay(1)
```

Im folgenden Beispiel wird sleep() benutzt, um VpPause() 10 Sekunden lang an der Ausführung zu hindern. Allerdings kann das Abspielen während dieser 10 Sekunden nicht unterbrochen werden, da der sleep()-Befehl dies verhindert.

```
VpVideoLoad ( "C:\VIDEO\BEISPIEL.AVI", 0)
VpPlay(0)
sleep(10)
VpPause(0)
```

Siehe auch:

VpPlaySync

VpPlaySync

Verfügbar ab: Professional Edition (Video-Kit)

Spielt das Video ab, bis die vorgegebene Position erreicht ist

Deklaration:

```
VpPlaySync ( Bildnummer, Player ) -> Fehlercode
```

Parameter:

Bildnummer	Nummer des Bildes, bei dem das Abspielen beendet werden soll
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll. Reserviert, immer auf Null zu setzen. Die Funktion kann nur auf den Videoplayer im FAMOS-Pluginfenster angwendet werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Die zur Zeit geladene Video-Datei wird abgespielt. Dabei werden die gegenwärtige Position und die Abspielgeschwindigkeit berücksichtigt. Das Abspielen erfolgt synchron, d.h. es ist während des Abspielens nicht möglich, weitere Befehle zu geben; auch die Anzeigen und die Verbindung zum Kurvenfenster werden während des Abspielens nicht aktualisiert. VpPlaySync() erwartet die Angabe einer Position (als Bildnummer), an der das Abspielen beendet werden soll.

Beispiele:

Im folgenden Beispiel wird eine Datei namens BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und mit der voreingestellten Abspielgeschwindigkeit ab dem ersten Bild abgespielt. Das Abspielen läßt sich nicht unterbrechen, bis das letzte Bild erreicht ist.

```
VpVideoLoad ("C:\VIDEO\BEISPIEL.AVI", 0)
length = VpGetLengthFrames(0)
VpPlaySync(length, 0)
```

Siehe auch:

VpPlay

VpSelect

Verfügbar ab: Professional Edition (Video-Kit)

Auswahl des Video-Players im aktuellen Datenbrowser-Panel, auf den nachfolgende Aufrufe der Funktionen des Video-Kits wirken sollen.

Deklaration:

```
VpSelect ( TxVideoplayerName )
```

Parameter:

TxVideoplayerName	Name des auszuwählenden Videoplayer-Elements.

Beschreibung:

Die Funktionen des Video-Kits besitzen einen Parameter, mit dem angegeben wird, ob der Videoplayer im integrierten FAMOS-Plugin (0) oder ein Videoplayer im aktuellen Panel (1) gemeint ist. Wenn das Panel mehrere Videoplayer-Elemente besitzt, wird über diese Funktion das anzusprechende Element ausgewählt. Ohne Aufruf dieser Funktion wird immer der zuerst gefundene Videoplayer angesteuert.

Die Auswahl des Videoplayers kann entweder in der Form [Seitenname].[Elementname] oder nur über den Elementnamen erfolgen. Wenn die Seite nicht explizit angegeben ist, wird der Videoplayer wie folgt gesucht:

- Wenn die Funktion innerhalb einer Ereignissequenz aufgerufen wird und das Ereignis einer Seite zugeordnet werden kann (z.B. das 'Gedrückt'-Ereignis eines Knopfes), so wird nach einem Videoplayer mit dem angegebenen Namen auf dieser Seite gesucht.
- Anderenfalls wird auf der aktiven (sichtbaren) Seite gesucht.

Falls kein Panel im Datenbrowser geladen ist, sich das aktuelle Panel im Design-Modus befindet oder kein Videoplayer mit dem angegebenen Namen vorhanden ist, bricht die Funktion mit Fehlermeldung ab.

Multithreading: Die Funktionen des Video-Kits dürfen überall aufgerufen werden und wirken global. Der hier selektierte Player ist also für alle Ausführungs-Threads gültig.

Beispiele:

Auf einer Panelseite sind 2 Videoplayer und ein Knopf vorhanden. Auf Knopfdruck werden 2 Videodateien geladen und abgespielt.

Ereignis-Sequenz 'Gedrückt' des 'Start'-Knopfes

```
VpSelect( "Player1")
VpVideoLoad( "c:\video\sample1.avi", 1)
VpPlay(1)
VpSelect( "Player2")
VpVideoLoad( "c:\video\sample2.avi", 1)
VpPlay(1)
```

VpSetAbsStartTime

Verfügbar ab: Professional Edition (Video-Kit)

Setzt Datum und Uhrzeit der Erstellung der aktuell geladenen Video-Datei.

Deklaration:

```
VpSetAbsStartTime ( Zeitpunkt, Player ) -> Fehlercode
```

Parameter:

Zeitpunkt	Erstellungszeitpunkt
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll. Reserviert, immer auf Null zu setzen. Die Funktion ist nur auf den Videoplayer im FAMOS-Plugin-Fenster anwendbar.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Benutzt man in einem verbundenen Kurvenfenster absolute Zeitangaben, ist es notwendig, die absolute Startzeit der Videoaufzeichnung zu berücksichtigen. Standardmäßig wird dafür der Erstellungszeitpunkt des Videos benutzt. Mit der Funktion VpSetAbsStartTime() können Datum und Uhrzeit des Starts der Videoaufzeichnung gesetzt werden. Das Format ist ein Text der Form "tt.mm.jj hh:mm:ss" (Tag.Monat.Jahr Stunden:Minuten:Sekunden).

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO wird geladen, ihre absolute Startzeit wird auf den 1. Januar 2000 0:00 Uhr gesetzt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpSetAbsStartTime("01.01.00 00:00:00", 0)
```

Siehe auch:

VpGetAbsStartTime, VpGetXOffset, VpSetXOffset

VpSetAbsStartTime2

Verfügbar ab: Professional Edition (Video-Kit)

Setzt Datum und Uhrzeit der Erstellung der aktuell geladenen Video-Datei.

Deklaration:

```
VpSetAbsStartTime2 ( Zeitpunkt, Player ) -> Fehlercode
```

Parameter:

Zeitpunkt	Erstellungszeitpunkt (FAMOS-Zeitformat). -1 für automatisch.
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll. Reserviert, immer auf 1 zu setzen. Die Funktion ist nur auf den ausgewählten Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers anwendbar. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Benutzt man in einem verbundenen Kurvenfenster absolute Zeitangaben, ist es notwendig, die absolute Startzeit der Videoaufzeichnung zu berücksichtigen. Standardmäßig wird dafür der Erstellungszeitpunkt des Videos benutzt. Mit der Funktion VpSetAbsStartTime2() kann Datum und Uhrzeit des Starts der Videoaufzeichnung angepasst werden.

Die Zeitangabe im FAMOS-Zeitformat kann mit den Funktionen der Gruppe '18> Datum, Uhrzeit' erzeugt werden.

Die Funktion setzt die entsprechende Eigenschaft am Videoplayer, welche gegenüber einer ggf. auch vorhandenen privaten Videodatei-Eigenschaft (gespeichert im Dateikopf oder in einer parallelen Konfigurationsdatei *.ivi) dominant ist. Um die Videoplayer-Eigenschaft auf 'automatisch' zurückzusetzen (und damit wieder ggf. die Einstellung der Videodatei wirksam werden zu lassen), geben Sie den Wert -1 an.

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO wird geladen, ihre absolute Startzeit wird auf den 1. Januar 2000 0:00 Uhr gesetzt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 1)
time = ZeitBinde( 1, 1, 2000, 0, 0, 0)
VpSetAbsStartTime2(time, 1)
```

Siehe auch:

VpGetAbsStartTime2, VpGetXOffset, VpSetXOffset

VpSetLink

Verfügbar ab: Professional Edition (Video-Kit)

Baut eine Verbindung mit einem vorhandenen Kurvenfenster oder dem Famos-Dateneditor auf.

Deklaration:

```
VpSetLink ( Bezugsdatensatz, Player ) -> Fehlercode
```

Parameter:

Bezugsdatensatz	Kanal, mit dem die Verbindung aufgebaut wird.
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll. Reserviert, immer auf Null zu setzen. Die Funktion ist nur auf den Videoplayer im FAMOS-Plugin-Fenster anwendbar.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Ein gegebener Datensatz, dessen x-Achse als Zeit interpretiert werden kann, kann in einem Kurvenfenster oder im Famos-Dateneditor dargestellt werden. Mit der Funktion VpSetLink() ist es möglich, eine Verbindung zwischen einem Video und diesem Bezugsdatendatz herzustellen. Das hat zur Folge, daß Video und Datensatz synchronisiert werden, d.h. das Abspielen des Videos oder Ziehen des Positionscursors in x-Richtung führt zu einer entsprechenden Anpassung der Position im verbundenen Fenster. Der Startzeitpunkt wird dabei vom verbindenden Fenster bestimmt. Ist für den Datensatz die absolute Zeitdarstellung gewählt, wird diese auch beim Abspielen des Videos berücksichtigt. Der relative Startzeitpunkt (Versatz, x-Offset) wird in jedem Fall beachtet.

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO wird geladen und mit einer Variablen namens test verbunden, die aus dem Datensatz TEST.DAT mit der Kurvenfenster-Konfiguration TEST.CCV erzeugt wird. Anschließend wird das Video abgespielt; das Kurvenfenster zeigt die zugehörigen Meßwerte.

```
id=FileOpenDSF("TEST.DAT", 0)
test = FileObjRead(id, 1)
KvKonfig(test, "TEST.CCV")
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpSetLink(test, 0)
VpPlay(0)
```

Siehe auch:

VpDelLink, VpLinkExists

VpSetPlayRate

Verfügbar ab: Professional Edition (Video-Kit)

Setzt die Abspielgeschwindigkeit auf den angegebenen Wert.

Deklaration:

```
VpSetPlayRate ( Abspielgeschwindigkeit, Player ) -> Fehlercode
```

Parameter:

Abspielgeschwindigkeit	Setzt die Abspielgeschwindigkeit auf den angegebenen Wert.
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das gegenwärtig geladene Video wird mit voreingestellter Geschwindigkeit abgespielt. Diese Geschwindigkeit wird in Bildern pro Sekunde angegeben (frames per second, FPS); sie ist in der Videodatei gespeichert. Mit der Funktion VpSetPlayRate() kann die Abspielgeschwindigkeit auf einen anderen Wert gesetzt werden. Dieser Wert gilt, solange das Video geladen ist; er wird nicht gespeichert.

Die maximale Abspielgeschwindigkeit beträgt das 200fache des voreingestellten Wertes. Wobei diese in Abhängigkeit zum Video- bzw. Audio-CODEC steht, bei den meisten Videos mit Audio ist somit nur eine maximale Abspielgeschwindigkeit vom 2fachen des voreingestellten Wertes möglich.

Hinweis: Durch Übergeben einer Null für die gewünschte Abspielgeschwindigkeit wird der voreingestellte Wert gesetzt.

Beispiele:

Im folgenden Beispiel wird das Video BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und seine Abspielgeschwindigkeit auf den Wert 5 FPS gesetzt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpSetPlayRate(5, 0)
VpPlay(0)
```

Beträgt die voreingestellte Abspielgeschwindigkeit der Datei z.B. 10 FPS, dann wird sie nach dem Ausführen der Sequenz mit halber Geschwindigkeit abgespielt.

Siehe auch:

VpGetPlayRate, VpGetRecordRate, VpSetRecordRate

VpSetPosFrames

Verfügbar ab: Professional Edition (Video-Kit)

Zeigt das Bild mit der übergebenen Nummer an.

Deklaration:

```
VpSetPosFrames ( Bildnummer, Player ) -> Fehlercode
```

Parameter:

Bildnummer	Nummer des Bildes, zu dessen Position gesprungen werden soll.
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Videos sind als (komprimierte) Einzelbilder in einer festen Reihenfolge gespeichert. Im Video-Player beginnt die Zählung mit 1; Bild 1 ist also das aktuelle Bild, wenn die Position (in Sekunden) im Bereich von 0.0000s bis (Länge des Videos in Sekunden / Anzahl der Bilder) liegt. VpSetPosFrames() setzt das Video an die Position, die der übergebenen Bildnummer entspricht. Das Video wird angehalten, wenn es zum Zeitpunkt der Ausführung des Befehls abgespielt wurde.

Beispiele:

Im folgenden Beispiel wird ein Panel mit Videoplayer geladen, das Video BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO 10s lang abgespielt, danach wird das Bild mit der Nummer 5 angezeigt.

```
DbLoadPanel("C:\VIDEO\VIDEO.PANEL", 0)
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 1)
VpPlay(1)
sleep(10)
VpSetPosFrames(5, 1)
```

Siehe auch:

VpGetPosFrames, VpGetPosSeconds, VpSetPosSeconds

VpSetPosSeconds

Verfügbar ab: Professional Edition (Video-Kit)

Zeigt das Bild an, das zur angegebenen Zeit das aktuelle Bild ist.

Deklaration:

```
VpSetPosSeconds ( Sekunden, Player ) -> Fehlercode
```

Parameter:

Sekunden	Position des Bildes, das angezeigt werden soll, in Sekunden.
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Beim Aufnehmen eines Videos werden die Einzelbilder mit einer bestimmten Aufnahmegeschwindigkeit gespeichert. Die Position eines Bildes in Sekunden ist dann die Bildnummer dividiert durch die Aufnahmegeschwindigkeit. Die Funktion VpSetPosSeconds() stellt das Video auf das Bild, das zur angegebenen Zeit aufgenommen wurde.

Beispiele:

Im folgenden Beispiel wird ein Panel mit Videoplayer geöffnet, das Video BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO geladen und ab dem Bild abgespielt, das nach 10s aufgenommen wurde.

```
DbLoadPanel("C:\VIDEO\VIDEO.PANEL", 0)
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 1)
VpSetPosSeconds(10, 1)
VpContinue(1)
```

Siehe auch:

VpGetPosSeconds, VpGetPosFrames, VpSetPosFrames

VpSetRecordRate

Verfügbar ab: Professional Edition (Video-Kit)

Setzt die Aufnahmegeschwindigkeit der aktuell geladenen Video-Datei in Bildern pro Sekunde.

Deklaration:

```
VpSetRecordRate ( Rate, Player ) -> Fehlercode
```

Parameter:

Rate	Ursprüngliche Geschwindigkeit, mit der das Video aufgenommen wurde.
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Die Videodatei besteht aus (komprimierten) Einzelbildern, die in einer bestimmten Geschwindigkeit aufgenommen wurden. Diese wird in Bildern pro Sekunde (frames per second, FPS) angegeben. Standardmäßig wird angenommen, daß Aufnahmegeschwindigkeit und Abspielgeschwindigkeit gleich sind. Sie können jedoch auch verschiedene Werte haben, z.B. wenn Hochgeschwindigkeitsaufnahmen in .AVI-Dateien umgewandelt werden. Mit Hilfe der Funktion VpSetRecordRate() kann die Aufnahmegeschwindigkeit eingestellt werden. Zeitangaben für Position und Länge werden automatisch angepaßt. Die Abspielgeschwindigkeit ist jedoch unabhängig von der Aufnahmegeschwindigkeit.

Videoplayer-Element im Datenbrowser:

Die Funktion setzt die entsprechende Eigenschaft am Videoplayer, welche gegenüber einer ggf. auch vorhandenen privaten Videodatei-Eigenschaft (gespeichert im Dateikopf oder in einer parallelen Konfigurationsdatei *.ivi) oder der originalen Aufnahmerate dominant ist. Um die Videoplayer-Eigenschaft auf 'automatisch' zurückzusetzen (und damit wieder ggf. die Einstellung an der Videodatei wirksam werden zu lassen), geben Sie den Wert 0 an.

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO wird geladen und ihre Aufnahmegeschwindigkeit auf 1000 Bilder pro Sekunde eingestellt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpSetRecordRate(1000, 0)
length=VpGetLengthSeconds(0)
```

Nimmt man an, daß die Datei aus 1000 Bildern besteht, dann hat die Variable length nach dem Ausführen der Sequenz den Wert 1. Ist die Abspielgeschwindigkeit auf 10 FPS eingestellt, dauert das vollständige Abspielen des Videos 100s. Das Abspielen ist also gegenüber der Aufnahme auf 1/100 verlangsamt.

Siehe auch:

VpGetRecordRate, VpGetPlayRate, VpSetPlayRate

VpSetXOffset

Verfügbar ab: Professional Edition (Video-Kit)

Setzt den zeitlichen Versatz der aktuell geladenen Video-Datei in Sekunden.

Deklaration:

```
VpSetXOffset ( Rate, Player ) -> Fehlercode
```

Parameter:

Rate	Anfangszeitpunkt des Videos in Sekunden
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Standardmäßig beginnt die Wiedergabe einer Videodatei zum Zeitpunkt 0.000s. Mit Hilfe der Funktion VpSetXOffset() kann dieser Zeitpunkt auf einen anderen Wert gesetzt werden.

Videoplayer-Element im Datenbrowser:

Die Funktion setzt die entsprechende Eigenschaft am Videoplayer, welche gegenüber einer ggf. auch vorhandenen privaten Videodatei-Eigenschaft (gespeichert im Dateikopf oder in einer parallelen Konfigurationsdatei *.ivi) dominant ist. Um die Videoplayer-Eigenschaft auf 'automatisch' zurückzusetzen (und damit wieder ggf. die Einstellung an der Videodatei wirksam werden zu lassen), geben Sie den Wert 1e35 an.

Beispiele:

Die Datei BEISPIEL.AVI aus dem Verzeichnis C:\VIDEO wird geladen und mit einer Variablen namens test verbunden, die aus dem Datensatz TEST.DAT mit der Kurvenfenster-Konfiguration TEST.CCV erzeugt wird. Um einen zeitlichen Versatz zwischen den Anfangszeitpunkten beider Dateien auszugleichen, wird die relative Startzeit des Videos auf 1s gesetzt, so daß das erste Bild des Videos mit dem Meßwert zusammenfällt, der nach einer Sekunde aufgenommen wurde.

```
id=FileOpenDSF("TEST.DAT", 0)
test = FileObjRead(id, 1)
KvKonfig(test, "TEST.CCV")
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpSetLink(test, 0)
VpSetXOffset(1, 0)
```

Siehe auch:

VpGetXOffset, VpGetAbsStartTime, VpSetAbsStartTime

VpSingleStep

Verfügbar ab: Professional Edition (Video-Kit)

Zeigt das Bild an, das dem gegenwärtigen Bild folgt.

Deklaration:

```
VpSingleStep ( Player ) -> Fehlercode
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das nächste Bild der Video-Datei wird als Standbild angezeigt. Ist die Datei am Ende angelangt, gibt die Funktion einen Fehler zurück.

Beispiele:

Im folgenden Beispiel wird eine Datei namens BEISPIEL.AVI im Pluginfenster geladen. Es wird das zweite Bild angezeigt, und nach einer Pause von 10 Sekunden das dritte.

```
VpVideoLoad ("C:\VIDEO\BEISPIEL.AVI", 0)
VpSingleStep(0)
sleep(10)
VpSingleStep(0)
```

Siehe auch:

VpBackStep

VpStop

Verfügbar ab: Professional Edition (Video-Kit)

Hält die Video-Datei an und setzt sie an den Anfang zurück.

Deklaration:

```
VpStop ( Player ) -> Fehlercode
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Das Video wird unabhängig vom gegenwärtigen Zustand angehalten; es wird das erste Bild gezeigt. Einstellungen für die Aufnahme- und Abspielgeschwindigkeit sowie relative und absolute Startzeit (im Info-Dialog) bleiben erhalten.

Beispiele:

Im folgenden Beispiel wird angenommen, daß sich die Datei BEISPIEL.AVI im Verzeichnis C:\VIDEO befindet. Diese Datei wird im Pluginfenster geladen, 10 Sekunden lang abgespielt, und dann zurückgesetzt.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpPlay(0)
sleep(10)
VpStop(0)
```

Siehe auch:

VpVideoClose

Verfügbar ab: Professional Edition (Video-Kit)

Schließt die zur Zeit geöffnete Video-Datei.

Deklaration:

```
VpVideoClose ( Player ) -> Fehlercode
```

Parameter:

Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Video-Player kann zu jedem Zeitpunkt höchstens eine Video-Datei geöffnet haben. Diese wird mit VpVideoClose() wieder geschlossen, und die Steuerelemente (abgesehen vom Schalter zum Laden einer Video-Datei) werden in den inaktiven Zustand versetzt.

Beispiele:

Im folgenden Beispiel wird angenommen, daß sich im Verzeichnis C:\VIDEO eine Datei mit dem Namen BEISPIEL.AVI befindet. Diese Datei wird im Pluginfenster geladen, 10 Sekunden lang abgespielt und wieder geschlossen.

```
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
VpPlay(0)
sleep(10)
VpVideoClose(0)
```

Siehe auch:

VpVideoLoad

VpVideoLoad

Verfügbar ab: Professional Edition (Video-Kit)

Lädt die übergebene Video-Datei.

Deklaration:

```
VpVideoLoad ( Dateiname, Player ) -> Fehlercode
```

Parameter:

Dateiname	Name und Pfad der Video-Datei, die geladen werden soll.
Player	Auswahl des Video-Players, für den die Funktion ausgeführt werden soll.
	0 : Videoplayer im FAMOS-Plugin-Fenster
	1 : Videoplayer im aktuellen Panel des FAMOS-Datenbrowsers. Sind mehrere Player enthalten, kann durch einen vorherigen Aufruf der Funktion VpSelect() der gewünschte Player festgelegt werden.
Fehlercode	Erfolg der Funktion (Optional).
	0 : Funktion erfolgreich ausgeführt
	< 0 : Fehlercode

Beschreibung:

Der Parameter "Dateiname" erwartet den Namen und den Pfad einer Datei, die der Video-Player abspielen kann. Die Endung (.avi, .mpg, .mov oder andere) muß im Namen enthalten sein. Der Video-Player öffnet dann diese Datei und springt an den Anfang des Videos. Dieses kann danach mit den Steuerelementen oder mit weiteren Funktionen des Video-Kits abgespielt werden.

Ist zum Zeitpunkt des Aufrufs von VpVideoLoad() eine andere Videodatei geöffnet, wird diese geschlossen.

Multithreading: Die Funktionen des Video-Kits dürfen überall aufgerufen werden und wirken global. Das hier geladene Video ist also für alle Ausführungs-Threads gültig.

Beispiele:

Im folgenden Beispiel wird angenommen, daß sich im Pfad C:\VIDEO eine AVI-Datei namens BEISPIEL.AVI befindet. Die Datei wird im Video-Plugin geladen.

```
erg = VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 0)
```

Im folgenden Beispiel wird ein Panel mit Videoplayer geöffnet, ein Video in den Player geladen und die Wiedergabe gestartet.

```
DbLoadPanel("C:\VIDEO\VIDEO.PANEL", 0)
VpVideoLoad("C:\VIDEO\BEISPIEL.AVI", 1)
```

Siehe auch:

VpVideoClose

WFTLOAD

Laden einer Nicolet WFT-Datei (Copyright 1988 Nicolet Instrument Corporation).
Alternativer Name: **WFTLADEN**

Deklaration:

```
WFTLOAD EwSegment EwZeitbasis Dateiname Variablenname
```

Parameter:

EwSegment	Zu ladendes Segment (1..1024)
EwZeitbasis	Zu ladende Zeitbasis (1..3)
Dateiname	Name der zu ladenden Datei
Variablenname	Variable, in die der Dateiausschnitt eingetragen wird

Beschreibung:

Eine WFT-Datei der Firma Nicolet wird in imc FAMOS geladen. Eine WFT-Datei kann 1 bis 1024 Segmente und 1 bis 3 Zeitbasen enthalten. Über die Parameter [Segment] und [ZeitBasis] können ein bestimmtes Segment und eine Zeitbasis eingelesen werden. Der Parameter [Segment] darf Werte zwischen 1 und 1024 annehmen, der Parameter [ZeitBasis] Werte zwischen 1 und 3.

Der ausgewählte Dateiausschnitt wird mit der unter [Variablenname] angegebenen Bezeichnung geladen.

Für weitere Dateiformate der Firma Nicolet, wie z.B. "Nicolet Team" und "Nicolet Frequency Domain (WFF)" stehen externe Importfilter zur Verfügung, solche Dateien können mit der Funktion FileLoad() oder FileOpenFAS() geladen werden:

```
FileLoad("c:\data\channel1.wff", "Nicolet_FreqDomain.FAS", 0)
```

- Der Dateiname kann ein vollständiger Pfadname inklusive Verzeichnis und Dateinamen-Erweiterung sein, darf aber auch ohne beides angegeben werden. Dann wird die Datei in dem Verzeichnis gesucht, aus dem FAMOS aktuell Daten lädt. Die Erweiterung wird automatisch gewählt.
- Der Dateiname darf auch in Anführungszeichen angegeben werden. Dies ist z.B. dann notwendig, wenn der Pfad Leerzeichen enthält.

Beispiele:

```
WFTLOAD 4 1 WAVE0001 seg4
WFTLOAD 4 1 c:\data\WAVE0001 seg4
```

In beiden Fällen wird aus der Datei "WAVE0001.WFT" die erste Zeitbasis des vierten Segments gelesen und in imc FAMOS unter der Bezeichnung "seg4" eingetragen.

```
seg = 4
bas = 1
WFTLOAD seg bas WAVE01 daten
```

Zwei Variablen definieren den gewünschten Bereich der Datei "WAVE01". Eingetragen wird er unter der Bezeichnung "Daten".

```
WFTLOAD 4 1 "c:\Meine Daten\WAVE0001" seg4
```

Der Pfadname enthält Leerzeichen und muss deshalb in Anführungszeichen angegeben werden.

Siehe auch:

FileLoad, FileOpenFAS, LDIR

WHILE

Bedingungs-gesteuerte Schleife. Die nachfolgenden Anweisungen werden solange zyklisch abgearbeitet, wie der hier angegebene Ausdruck einen Wert > 0 liefert.
Alternativer Name: **SOLANGE**

Deklaration:

```
WHILE Bedingung
```

Parameter:

Bedingung	Solange die Bedingung erfüllt ist (Auswertung liefert einen Wert >0), werden die Anweisungen des folgenden Blocks wiederholt.

Beschreibung:

Das Ende der Schleife wird mit dem Befehl END markiert.

Als Bedingung kann z.B. eine Einzelwert-Variable oder ein komplexerer Ausdruck unter Verwendung von logischen Operatoren (AND, OR..) und/oder Vergleichsoperatoren (<, =, ...) angegeben werden.

In der Schleife können die Befehle BREAK und CONTINUE verwendet werden, um die Abarbeitung der inneren Anweisungen vorzeitig abzubrechen.

Statt einer WHILE-Schleife kann in vielen Anwendungsfällen auch bequemer eine FOR- oder FOREACH-Schleife verwendet werden.

Beispiele:

Ein Datensatz soll so oft geglättet werden, bis seine Standardabweichung nicht mehr größer als 0.2 ist.

```
WHILE StDev(data) > 0.2
   Data = Smo5(data)
END
```

Aus einer Datei mit mehreren Datenobjekten wird das erste gefundene Textobjekt ausgelesen.

```
fh = FileOpenDSF("test.dat", 0)
IF fh > 0
   n = FileObjNum?(fh)
   i = 1
   WHILE i <= n
      IF FileObjType?(fh, i) = 2
         text = FileObjRead(fh, i)
         BREAK ; Schleife abbrechen
      END
      i = i + 1
   END
   FileClose(fh)
END
```

Hinweis: Vorstehende Aufgabe lässt sich mit einer FOR-Schleife eleganter lösen.

Siehe auch:

FOREACH, FOR

XDel

Das Inkrement eines Datensatzes in x-Richtung (Delta-X, Abtastzeit) wird gesetzt.

Deklaration:

```
XDel ( Daten, EwXDelta ) -> Ergebnis
```

Parameter:

Daten	Datensatz, dessen x-Inkrement gesetzt werden soll.
EwXDelta	Neues Delta-X (>0)
Ergebnis	Datensatz-Kopie mit neuem Delta-X.

Beschreibung:

Es wird eine Kopie der Eingangsdaten erzeugt und das angegebene x-Inkrement eingetragen. Alle anderen Zahlen- und Kennwerte bleiben unberührt.

Delta-X entspricht der Differenz zwischen den x-Koordinaten benachbarter Punkte des Datensatzes.

Bei Messdaten, die äquidistant über der Zeit abgetastet werden, entspricht dies der **Abtastzeit**.

- Das neue Delta-X sollte die x-Einheit des Datensatzes haben.
- Das neue Delta-X sollte in der Größenordnung nicht zu viele Zehnerpotenzen unterhalb des x-Offset liegen. Ansonsten können Unterschiede in den x-Koordinaten der Punkte des Datensatzes nicht (ausreichend) aufgelöst werden.

Beispiele:

Die Abtastzeit eines Datensatzes, der in Form von ASCII-Daten ohne Zeitbasisinformation gelesen wurde, wird auf 2ms gesetzt:

```
SetUnit(NDdata,0, "s")
NDdata = XDel(NDdata, 2e-3)
```

Siehe auch:

XDel?, XOff, Leng, XDELTA

XDel?

Das Inkrement eines Datensatzes in x-Richtung (Delta-X, Abtastzeit) wird ermittelt.

Deklaration:

```
XDel? ( Daten ) -> EwXDelta
```

Parameter:

Daten	Datensatz, dessen Delta-X ermittelt werden soll.
EwXDelta	Delta-X

Beschreibung:

Das Delta-X entspricht der Differenz zwischen den x-Koordinaten benachbarter Punkte des Datensatzes.

Bei Messdaten, die äquidistant über der Zeit abgetastet werden, entspricht dies der **Abtastzeit**.

- Das Ergebnis hat die x-Einheit des übergebenen Datensatzes.

Beispiele:

Die Dauer eines Datensatzes ist das Produkt aus seiner Länge und Abtastzeit:

```
duration = XDel?(NDdata) * Leng?(NDdata)
```

Die Abtastfrequenz ist der Kehrwert der Abtastzeit:

```
freq = 1 / XDel?(NDdata)
```

Siehe auch:

XDel, XOff?, Leng?

XDELTA

Abtastzeit, x-Delta setzen

Deklaration:

```
XDELTA Variablenname EwDeltaX
```

Parameter:

Variablenname	Name der Variablen, deren x-Achse neu geteilt werden soll.
EwDeltaX	Abstand zwischen 2 Datenpunkten in x-Einheiten

Beschreibung:

Das Kommando XDELTA ist veraltet, statt dessen sollte in neu zu erstellenden Sequenzen die Funktion XDel() verwendet werden.

Die Teilung der x-Achse (i.Allg. die Abtastzeit) wird betragsmäßig umdefiniert.

Beispiele:

```
XUNIT Spann s
XDELTA Spann 60
```

Die Variable "Spann" erhält zuerst die x-Einheit "s". Anschließend wird der Abstand zwischen zwei Datenpunkten zu "60" definiert. Er beträgt also 60 s.

Siehe auch:

XDel, XDel?, XOff

XlBuildA1Ref

Anwendungsbereich: Excel-Fernsteuerung

Bildet aus Spalten- und Zeilennummer(n) eine Zellen- bzw. Bereichsreferenz im A1-Stil.

Deklaration:

```
XlBuildA1Ref ( Zeile, Spalte, Bereichshöhe, Bereichsbreite [, BezugsZelle] ) -> TxA1Referenz
```

Parameter:

Zeile	Zeilennummer (1..)
Spalte	Spaltennummer (1..)
Bereichshöhe	Höhe (1..) des Bereichs
Bereichsbreite	Breite (1..) des Bereichs
BezugsZelle	Zellenreferenz für Bezugszelle im A1-Stil. Wennn angegeben, werden [Zeile] und [Spalte] nicht als absolute Angabe, sondern als relativer Offset zu dieser Zelle betrachtet. (optional)
TxA1Referenz	Zellen/Bereichs-Referenz im A1-Stil.

Beschreibung:

Einige Funktionen dieses Kits verlangen die Angabe einer Zelle mit einer Referenz im 'A1'-Stil (oder eines Bereichs in der Form 'A1:B2'), wobei die Buchstaben die Spalte (A=1, Z = 26, AA = 27 usw.) und die Ziffern die Zeilennummer angeben.

Mit dieser Funktion kann aus Spalten-/Zeilennummer und der Größe des gewünschten Bereichs eine solche Zellenreferenz konstruiert werden.

Wenn als 5. Parameter eine Bezugszelle angegeben wird, werden Spalten- und Zeilennummer als Offset zu dieser Zelle interpretiert.

Beispiele:

```
ref = XlBuildA1Ref( 1, 1, 1, 1)           ; Ergebnis: "A1"
ref = XlBuildA1Ref( 3, 2, 1, 1)           ; "B3"
ref = XlBuildA1Ref( 1, 2, 100, 1)         ; "B1:B100"
ref = XlBuildA1Ref( 2, 2, 255, 100)       ; "B2:CW256"
ref = XlBuildA1Ref( 65536, 256, 1, 1)     ; "IV65536"    (maximale Tabellengröße im XLS-Format)
ref = XlBuildA1Ref( 1048576, 16384, 1, 1) ; "XFD1048567" (maximale Tabellengröße im XLSX-Format)
ref = XlBuildA1Ref( 1, 1, 1, "A1")        ; "B2"
ref = XlBuildA1Ref( 0, -1, 1, 1, "B2")    ; "A2"
```

Eine neue Excel-Tabelle wird erzeugt und alle Kanäle einer Datengruppe 'MyGroup' fortlaufend spaltenweise übertragen. Anschließend wird das Ausgabeformat für alle gefüllten Zellen auf maximal 2 Nachkommastellen festgelegt (wobei davon ausgegangen wird, dass alle Kanäle die selbe Länge haben).

```
XlWbNew("")
ChanNum =  GrChanNum?(MyGroup)
FOR I = 1 TO ChanNum
   ref = XlBuildA1Ref( 1, I, 1, 1)
   XlSetValues( ref, 0, MyGroup:[I], 0)
END

count = leng?(MyGroup:[1])
range = XlBuildA1Ref( 1, 1, count, ChanNum)
XlSetCellFormat( range, "0,##")
XlWbSave( "c:\results\report", 0)
XlQuit()
```

In einer Tabelle wird eine Zeitspalte angelegt und entsprechend für die Ausgabe formatiert.

```
time = XlCreateTimeLine(channel1, 1, 0)
XlSetValues( "A1", 0, time, 1)
Count = Leng?(time)
Range = XlBuildA1Ref(1, 1, count, 1)
XlSetCellFormat(Range, "TT.MM.JJ hh:mm:ss,0")
```

Siehe auch:

XlSetText, XlSetValue, XlGetValues

XlCellMerge

Anwendungsbereich: Excel-Fernsteuerung

Verbindet die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlCellMerge ( TxBereich )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.

Beschreibung:

Die angegebenen Zellen im aktiven Tabellenblatt werden verbunden. Sind die Zellen bereits gefüllt, erzeugt EXCEL eine Meldung. Um eine Unterbrechung des Sequenzablaufs zu vermeiden, sollten die Zellen vor dem Verbinden leer sein.

Beispiele:

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Matrix für die Tabelle erzeugen
   Matrix = MatrixInit( 10, 10, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 3, I+2, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   ; Zellen verbinden
   XlCellMerge("A1:B14")
   XlCellMerge("C1:L2")
   XlCellMerge("C13:L14")
   XlCellMerge("M1:N14")
   ; rote Kreuze rechts und links in die verbundenen Zellen zeichnen
   XlSetBorderStyle("A1:B14", 48, "continous")
   XlSetBorderColor("A1:B14", 48, RGB(255,0,0))
   XlSetBorderStyle("M1:N14", 48, "continous")
   XlSetBorderColor("M1:N14", 48, RGB(255,0,0))
END
```

XlCreateTimeLine

Anwendungsbereich: Excel-Fernsteuerung

Konstruiert eine explizite Zeitspur für einen equidistanten Datensatz für die anschließende Übertragung nach Excel.

Deklaration:

```
XlCreateTimeLine ( Datensatz, Art, Reserviert ) -> Zeitspur
```

Parameter:

Datensatz	Datensatz, für den die Zeitspur erzeugt werden soll.
Art	
	0 : Relative Zeitspur
	1 : Absolute Zeitspur
Reserviert	Reservierter Parameter, immer 0
Zeitspur	Konstruierte Zeitspur

Beispiele:

Eine neue Excel-Datei mit einem Tabellenblatt wird erzeugt und ein Datensatz in diese Tabelle übertragen. Die erste Spalte wird als Zeitspalte mit absoluten Datum/Uhrzeit-Angaben und 1ms Auflösung konfiguriert, die zweite Spalte enthält die eigentlichen Werte des Datensatzes.

Die so erstellte Datei wird dann gespeichert.

```
XlWbNew("")
time = XlCreateTimeLine(channel1, 1, 0)
XlSetValues( "A1", 0, time, 1)
Count = Leng?(time)
Range = XlBuildA1Ref(1, 1, count, 1)
XlSetCellFormat(Range, "TT.MM.JJ hh:mm:ss,0")
XlSetValues( "B1", 0, channel1, 0)
XlWbSave( "c:\results\report", 0)
XlQuit()
```

Wie oben, die Zeitspalte wird hier jedoch relativ konfiguriert (Sekunden seit Triggerzeit). Die Ausgabe beginnt in der 2. Zeile, in die Zelle A1 wird die Triggerzeit des Datensatzes eingetragen.

```
XlSetValue("A1", Time?(Channel1), 1)
XlSetCellFormat("A1", "TT.MM.JJ hh:mm:ss,0")
time = XlCreateTimeLine(channel1, 0, 0)
XlSetValues( "A2", 0, time, 0)
Count = Leng?(time)
Range = XlBuildA1Ref(2, 1, count, 1)
XlSetCellFormat(Range, "0,000")
XlSetValues( "B2", 0, channel1, 0)
```

Siehe auch:

XlSetValues

XlFind

Anwendungsbereich: Excel-Fernsteuerung

Sucht im aktuellen Blatt nach Text

Deklaration:

```
XlFind ( TxSuche, EwWo, EwGanzeZelle, EwGroßKlein, EwRichtung ) -> Suchergebnis
```

Parameter:

TxSuche	Der Text, nach dem gesucht werden soll.
EwWo	Gibt an, in welchem Inhalt gesucht werden soll.
	0 : Werte
	1 : Formeln
	2 : Kommentare
EwGanzeZelle	Gibt an, ob der gesamte Zellinhalt verglichen werden soll
	0 : Suchtext muss im Zelleninhalt enthalten sein.
	1 : Suchtext muss mit Zeileninhalt genau übereinstimmen.
EwGroßKlein	Groß-/Kleinschreibung beachten
	0 : Groß-/Kleinschreibung NICHT beachten
	1 : Groß-/Kleinschreibung beachten
EwRichtung	Gibt die Suchrichtung an.
	0 : Spaltenweise
	1 : Zeilenweise
Suchergebnis	Textfeld mit allen Fundstellen. Die Treffer werden in Form einer 'A1'-Referenz geliefert.

Beschreibung:

Dieser Befehl bildet die Funktionalität des Excel-Dialogs "Suchen..." nach.

Wenn die aktuelle Selektion mehr als eine Zelle umfasst, wird die Suche auf den selektierten Bereich eingegrenzt. Ansonsten wird im gesamten Tabellenblatt gesucht.

Wenn der zu suchende Text nicht gefunden wird, wird ein leeres Textfeld zurückgeliefert.

Beispiele:

Eine Arbeitsmappe wird geladen und alle Tabellenblätter nach dem Text 'ENGINE_0124' durchsucht. Bei Erfolg wird der Wert in der Zelle rechts daneben ausgelesen.

```
IF XlWbOpen("z:\tmp\results.xlsx")
   sheetcount = XlSheetGetCount()
   FOR i = 1 TO sheetcount
      XlSheetActivate(i)
      hits = XlFind("ENGINE_0134", 0, 1, 0, 0)
      IF TxArrayGetSize(hits) = 1
         cell = XLBuildA1Ref(0, 1, 1, 1, hits[1])
         val = XlGetValue(cell, 0)
         BREAK
      END
   END
   XlQuit()
END
```

Eine Arbeitsmappe wird geladen und in der 2. Spalte des Tabellenblattes 'Table' alle Zellen gesucht, die den Text '_Engine_' enthalten. Der komplette Inhalt der gefundenen Zellen wird in ein neues Textarray eingetragen.

```
IF XlWbOpen("z:\tmp\results.xlsx")
   XlSheetActivate("Table2")
   XlSelectRange("B:B")
   hits = XlFind("_Engine_", 0, 0, 0, 0)
   foundNames = TxArrayCreate(0)
   FOR i = 1 TO TxArrayGetSize(hits)
      foundNames[i] = XlGetText(hits[i])
   END
```

```
   XlQuit()
END
```

Siehe auch:

XlGetText, XlSelectRange, XlBuildA1Ref

XlGetSelectedRange

Anwendungsbereich: Excel-Fernsteuerung

Excel: Selektierten Bereich abfragen

Deklaration:

```
XlGetSelectedRange ( ) -> TxBereich
```

Parameter:

TxBereich	Liefert den aktuell selektierten Zell-Bereich als 'A1'-Referenz. Leerer Text, wenn kein Zell-Bereich selektiert ist.

Beschreibung:

Die Funktion wirkt auf das aktive Blatt der aktiven Arbeitsmappe.

Beispiele für [TxBereich]-Angaben im 'A1'-Stil:

Selektion einer einzelnen Zelle	"C2"	Zelle C2 (3.Spalte, 2.Zeile)
Selektion eines zusammenhängenden Bereichs	"C2:E3"	Zelle C2 bis E3 (3. bis 5. Spalte, 2. bis 3. Zeile)
Selektion eines Spalte	"C:C"	Die 3. Spalte ist selektiert
Selektion mehrerer Spalten	"C:E"	Die 3., 4. und 5. Spalte ist selektiert
Selektion einer Zeile	"2:2"	Die 2. Zeile ist selektiert
Selektion mehrerer Zeilen	"2:4"	Die 2., 3. und 4. Zeile sind selektiert
Mehrfache Selektion	"A1:A3;C4"	Die Zellen A1,A2,A3 und C4 sind selektiert.

Wie im letzten Beispiel demonstriert, können auch mehrere Referenzen (getrennt durch Semikolon) geliefert werden, wenn ein nicht zusammenhängender Bereich selektiert ist.

Beispiele:

Die aktuell selektierte Zelle wird um eine Zeile nach unten verschoben:

```
sel = XlGetSelectedRange()
sel = XlBuildA1Ref(1, 0, 1, 1, sel)
XlSelectRange(sel)
```

Siehe auch:

XlSetSelectedRange, XlBuildA1Ref

XlGetText

Anwendungsbereich: Excel-Fernsteuerung

Ermittelt den Inhalt der angegebenen Zelle im aktiven Tabellenblatt als Text.

Deklaration:

```
XlGetText ( TxZelle ) -> TxInhalt
```

Parameter:

TxZelle	Zu lesende Zelle. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
TxInhalt	Inhalt der Tabellenzelle.

Beispiele:

Der Anwender wählt eine Excel-Datei zum Öffnen. Vom 2. Tabellenblatt der Datei werden aus der ersten Spalte die enthaltenen Werte ausgelesen, wobei in der ersten Zeile der Name des Kanals steht.

```
filename = DlgFileName("c:\results", "xlsx", "", 0)
IF XlWbOpen(filename)
   XlSheetActivate(2)
   TxName = XlGetText("A1")
   <TxName> = XlGetValues( "A2", 0, 0, 0)
END
```

Siehe auch:

XlSetText, XlGetValue, XlSetValue, XlGetValues

XlGetTextArray

Anwendungsbereich: Excel-Fernsteuerung

Ermittelt den Inhalt einer Spalte oder Zeile im aktiven Tabellenblatt.

Deklaration:

```
XlGetTextArray ( TxStartZelle, SvZeileOderSpalte, SvAnzahl ) -> Inhalt
```

Parameter:

TxStartZelle	An dieser Zelle beginnt das Lesen. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
SvZeileOderSpalte	Gibt an, ob in vertikaler Richtung (Spalte) oder horizontaler Richtung (Zeile) gelesen werden soll.
	0 : Vertikal (Spalte)
	1 : Horizontal (Zeile)
SvAnzahl	Gibt die maximale Anzahl zu lesender Zellen an. Das Lesen wird beendet, wenn die hier angegebene Anzahl erreicht ist.
Inhalt	Textarray mit dem gelesenen Inhalt der Zellen.

Beispiele:

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   XlSetText("A1","Text 1")
   XlSetText("A2","Text 2")
   XlSetText("B1","Text 3")
   XlSetText("B2","Text 4")
   aColumnA = XlGetTextArray("A1",0,2)
   aRow1    = XlGetTextArray("A1",1,2)
END
```

XlGetValue

Anwendungsbereich: Excel-Fernsteuerung

Ermittelt den Inhalt der angegebenen Zelle im aktiven Tabellenblatt und liefert diesen als Zahl.

Deklaration:

```
XlGetValue ( TxZelle, Format ) -> Inhalt
```

Parameter:

TxZelle	Zu lesende Zelle. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
Format	Datenformat der zu lesenden Zelle.
	0 : Der Wert der Zelle wird unkonvertiert gelesen.
	1 : Der gelesene Wert wird als Excel-Datum/Uhrzeit interpretiert (Anzahl der Tage seit 1.1.1900) und in das imc-Zeitformat (Anzahl der Sekunden ab 1.1.1980) konvertiert. Bei Daten < 1.1.1980 wird eine 0 zurückgegeben.
Inhalt	Inhalt der Tabellenzelle.

Beschreibung:

Im Fehlerfall, falls beispielsweise der Inhalt der Zelle nicht in eine Zahl konvertiert werden kann, wird eine 0 zurück geliefert.

Beispiele:

Ein Datensatz wird aus der ersten Spalte einer Excel-Datei gelesen. Die erste Zeile enthält den Namen, es folgen die Triggerzeit und die Abtastzeit. Die eigentlichen Daten beginnen in der 4. Zeile.

```
IF XlWbOpen("c:\results\report.xlsx")
   name = XlGetText("A1")
   time  = XlGetValue( "A2", 1)
   dx = XlGetValue( "A3", 0)
   <name> = XlGetValues("A4", 0, 0, 0)
   SetTime( <name>, time)
   XDELTA <name> dx
END
```

Siehe auch:

XlSetValue, XlSetText, XlGetText, XlGetValues

XlGetValues

Anwendungsbereich: Excel-Fernsteuerung

Ermittelt den (numerischen) Inhalt einer Spalte oder Zeile im aktiven Tabellenblatt.

Deklaration:

```
XlGetValues ( TxStartZelle, ZeileOderSpalte, Format, Anzahl ) -> Inhalt
```

Parameter:

TxStartZelle	An dieser Zelle beginnt das Lesen. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
ZeileOderSpalte	Gibt an, ob in vertikaler Richtung (Spalte) oder horizontaler Richtung (Zeile) gelesen werden soll.
	0 : Vertikal (Spalte)
	1 : Horizontal (Zeile)
Format	Datenformat des zu lesenden Bereichs.
	0 : Der Wert der Zellen wird unkonvertiert gelesen.
	1 : Die gelesenen Werte werden als Excel-Datum/Uhrzeit interpretiert (Anzahl der Tage seit 1.1.1900) und in das imc-Zeitformat (Anzahl der Sekunden ab 1.1.1980) konvertiert. Bei Daten < 1.1.1980 wird eine 0 zurückgegeben.
Anzahl	Gibt die maximale Anzahl zu lesender Zellen an. Das Lesen wird beendet, wenn die hier angegebene Anzahl erreicht ist oder bei der ersten Zelle, deren Inhalt nicht in eine Zahl konvertiert werden kann (z.B. leere Zelle). Wenn Sie eine 0 angeben, wird automatisch bis zur ersten nicht-numerischen Zelle gelesen.
Inhalt	Datensatz mit dem gelesenen Inhalt der Zellen.

Beispiele:

Ein Datensatz wird aus einer Excel-Datei gelesen. Die erste Spalte enthält die Zeitstempel der Werte, angegeben mit Datum/Uhrzeit. Die zweite Spalte enthält die entsprechenden numerischen Werte.

In FAMOS wird ein XY-Datensatz 'channel' angelegt, der als Triggerzeit die Zeit des ersten Samples zugewiesen bekommt. Die X-Spur enthält dann die relativen Differenzen zur Startzeit.

```
IF XlWbOpen("c:\results\report.xlsx")
   time = XlGetValues("A1", 0, 1, 0)
   data = XlGetValues("B1", 0, 0, 0)
   channel = XYof( time - time[1], data)
   SetTime( channel, time[1])
END
```

Siehe auch:

XlGetValues2, XlSetValues, XlGetValue, XlGetText

XlGetValues2

Anwendungsbereich: Excel-Fernsteuerung

Ermittelt den (numerischen) Inhalt einer Spalte oder Zeile im aktiven Tabellenblatt.

Deklaration:

```
XlGetValues2 ( TxStartZelle, ZeileOderSpalte, Format, Anzahl, Ersatzwert, Reserviert ) -> Inhalt
```

Parameter:

TxStartZelle	An dieser Zelle beginnt das Lesen. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
ZeileOderSpalte	Gibt an, ob in vertikaler Richtung (Spalte) oder horizontaler Richtung (Zeile) gelesen werden soll.
	0 : Vertikal (Spalte)
	1 : Horizontal (Zeile)
Format	Datenformat des zu lesenden Bereichs.
	0 : Der Wert der Zellen wird unkonvertiert gelesen.
	1 : Die gelesenen Werte werden als Excel-Datum/Uhrzeit interpretiert (Anzahl der Tage seit 1.1.1900) und in das imc-Zeitformat (Anzahl der Sekunden ab 1.1.1980) konvertiert. Bei Daten < 1.1.1980 wird eine 0 zurückgegeben.
Anzahl	Gibt die Anzahl zu lesender Zellen an.
Ersatzwert	Dieser Wert wird verwendet, falls ein Zelleninhalt nicht in einen numerischen Wert konvertiert werden kann.
Reserviert	Reservierter Parameter, auf 0 zu setzen.
Inhalt	Datensatz mit dem gelesenen Inhalt der Zellen.

Beschreibung:

Datenreihen in EXCEL-Tabellen enthalten manchmal Lücken, die ungültige Werte markieren sollen. Solche ungültigen Werte werden z.B. durch leere Zellen oder einen speziellen Text (z.B. 'NULL') gekennzeichnet.

Im Gegensatz zur Funktion XlGetValues(), die an solchen Zellen das Lesen beenden würde, überliest die Funktion XlGetValues2() solche nicht in eine Zahl konvertierbaren Zellen und trägt dafür in den erzeugten Datensatz einen wählbaren Ersatzwert ein.

Diese können dann in FAMOS bei Bedarf nachträglich aus dem Datensatz herausgeschnitten werden, dafür bietet sich z.B. die Funktion SearchLevel() an.

Beispiele:

Eine Excel-Datei enthält 2 Datensätze. Die erste Spalte enthält die gemeinsame Zeitspalte, angegeben mit Datum/Uhrzeit. Die zweite und dritte Spalte enthalten die entsprechenden numerischen Werte, wobei leere Zellen möglich sind, wenn bei einem Zeitstempel nur ein Kanal einen gültigen Wert besitzt.

Die 3. Spalte wird ausgelesen und in FAMOS ein XY-Datensatz 'channel' angelegt. Die ungültigen Werte werden anschließend aus dem Datensatz entfernt.

```
IF XlWbOpen("c:\results\report.xlsx")
   time  = XlGetValues("A1", 0, 1, 0)
   count = leng?(time)
   data = XlGetValues2("C1", 0, 0, count, -1e30, 0)
   channel = XYof( time - time[1], data)
   channel = SearchLevel(channel, 2, -1e29, 0, 0, 0, 0, 0)
   SetTime( channel, time[1])
END
```

Siehe auch:

XlGetValues, XlSetValues, XlGetValue, XlGetText

XlPaste

Anwendungsbereich: Excel-Fernsteuerung

Fügt den Inhalt der Zwischenablage in das aktive Blatt ein.

Deklaration:

```
XlPaste ( TxEinfügePosition ) -> Erfolg
```

Parameter:

TxEinfügePosition	Legt die Position fest, an der der Inhalt der Zwischenablage eingefügt werden soll. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle. Wenn leer, wird die aktuell selektierte Zelle verwendet.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Ein Kurvenfenster wird angezeigt und in die Zwischenablage kopiert. Ein neues Excel-Dokument wird erzeugt, das Kurvenbild an die Position 'B2' eingefügt und die Arbeitsmappe gespeichert.

```
CwLoadCCV(channel1, "layout.ccv")
CwAction("clipboard.copy")
XlWbNew("")
XlPaste("B2")
XlWbSave("z:\tmp\channel1.xlsx", 0)
XlQuit()
```

Siehe auch:

XlSetText, XlSetValue

XlQuit

Anwendungsbereich: Excel-Fernsteuerung

Beendet die verknüpfte Excel-Instanz.

Deklaration:

```
XlQuit ( )
```

Parameter:

Beschreibung:

Mit dieser Funktion wird die Excel-Instanz beendet, die explizit mit XlStart() oder implizit mit XlWbOpen()/XlWbNew() gestartet worden ist.

Eventuell noch offene Dateien werden geschlossen, eventuelle Änderungen gehen verloren.

Beispiele:

Eine Excel-Datei wird geöffnet und Daten eingefügt. Die aktualisierte Datei wird gedruckt, danach wird Excel wieder geschlossen.

```
IF XlWbOpen("c:\Templates\Template.xlsx")
   XlSetText("B1", "Channel1")
   XlSetValues("B2", 0, Channel1, 0)
   XlWbPrint()
   XlQuit()
END
```

Siehe auch:

XlStart, XlVisible

XlRunMacro

Anwendungsbereich: Excel-Fernsteuerung

Führt das angegebene Excel-Makro aus.

Deklaration:

```
XlRunMacro ( Makroname ) -> Erfolg
```

Parameter:

Makroname	Name des auszuführenden Makros.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Eine Excel-Datei wird geöffnet und Daten eingefügt. Anschließend wird ein Makro ausgeführt, welches in der Vorlage mit gespeichert ist. Die aktualisierte Datei wird unter neuem Namen gespeichert, danach wird Excel wieder geschlossen.

```
IF XlWbOpen("c:\Templates\Template.xlsx")
   XlSetText("B1", "Channel1")
   XlSetValues("B2", 0, Channel1, 0)
   XlRunMacro("Calculation"))
   XlWbSave("c:\results\report", 0)
   XlQuit()
END
```

Excel wird gestartet, sichtbar gemacht und ein Makro ausgeführt, welches in der Arbeitsmappe 'MyMacros.xlsm' definiert ist. Abschließend wird Excel wieder geschlossen.

```
IF XlStart()
   XlVisible(1)
    IF NOT( XlRunMacro("'c:\XLSTemplates\MyMacros.xlsm'!Macro2"))
      BoxMessage("Fehler", GetLastError(), "!1")
   END
   XlQuit()
END
```

Siehe auch:

XlWbOpen

XlSelectRange

Anwendungsbereich: Excel-Fernsteuerung

Selektiert eine Zelle oder einen Zellbereich.

Deklaration:

```
XlSelectRange ( TxBereich ) -> Erfolg
```

Parameter:

TxBereich	Zu selektierende Zelle bzw. Zellbereich. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle bzw. Zellbereichs.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die Funktion wirkt auf das aktive Blatt der aktiven Arbeitsmappe.

Beispiele für [TxBereich]-Angaben im 'A1'-Stil:

Selektion einer einzelnen Zelle	"C2"	Zelle C2 (3.Spalte, 2.Zeile)
Selektion eines zusammenhängenden Bereichs	"C2:E3"	Zelle C2 bis E3 (3. bis 5. Spalte, 2. bis 3. Zeile)
Selektion eines Spalte	"C:C"	Die 3. Spalte wird selektiert
Selektion mehrerer Spalten	"C:E"	Die 3., 4. und 5. Spalte wird selektiert
Selektion einer Zeile	"2:2"	Die 2. Zeile wird selektiert
Selektion mehrerer Zeilen	"2:4"	Die 2., 3. und 4. Zeile wird selektiert
Mehrfache Selektion	"A1:A3,C4"	Die Zellen A1,A2,A3 und C4 werden selektiert.

Wie im letzten Beispiel demonstriert, können Sie auch mehrere Referenzen (getrennt durch Semikolon) angeben, um nicht zusammenhängende Bereiche zu selektieren.

Beispiele:

Im aktuellen Blatt der aktuellen Arbeitsmappe werden alle Zellen, die den Text 'Overflow' enthalten, rot eingefärbt. Dazu wird ein Excel-Makro 'MakeCellRed' verwendet, welches in der Arbeitsmappe 'MyMacros.xlsm' definiert ist und auf die selektierte Zelle wirkt.

```
hits = XlFind("Overflow", 0, 0, 0, 0)
FOR i = 1 TO TxArrayGetSize(hits)
   XlSelectRange(hits[i])
   XlRunMacro("'c:\XLSTemplates\MyMacros.xlsm'!MakeCellRed")
END
```

VBA-Quelltext des Makros:

```
Sub MakeCellRed()
   With Selection.Interior
      .Pattern = xlSolid
      .PatternColorIndex = xlAutomatic
      .Color = 255
    End With
End Sub
```

Die aktuell selektierte Zelle wird um eine Zeile nach unten verschoben:

```
sel = XlGetSelectedRange()
sel = XlBuildA1Ref(1, 0, 1, 1, sel)
XlSelectRange(sel)
```

Siehe auch:

XlGetSelectedRange, XlBuildA1Ref

XlSetBorderColor

Anwendungsbereich: Excel-Fernsteuerung

Setzt die Farbe von Rahmenlinien für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetBorderColor ( TxBereich, SvRahmenLinie, SvLinienFarbe )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
SvRahmenLinie	Rahmenlinien der Zelle oder Gruppe.
	1 : Linie links
	2 : Linie rechts
	4 : Linie oben
	8 : Linie unten
	15 : Zellrahmen
	16 : Linie diagonal abwärts
	32 : Linie diagonal aufwärts
	48 : Kreuz
	64 : Linie Zellgruppe links
	128 : Linie Zellgruppe rechts
	256 : Linie Zellgruppe oben
	512 : Linie Zellgruppe unten
	960 : Zellgruppenrahmen
	1024 : Vertikale Linie innerhalb der Zellgruppe
	2048 : Horizontale Linie innerhalb der Zellgruppe
	3072 : Horizontale + vertikale Linie innerhalb der Zellgruppe
SvLinienFarbe	Farbe der Rahmenlinie

Beispiele:

Die Farben werden für die Rahmenlinien einer Zelle werden individuell erzeugt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Matrix für die Tabelle erzeugen
   Matrix = MatrixInit(5, 5, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues(ref, 0, Matrix[I], 0)
   END
   LOCAL colred   = 0
   LOCAL colgreen = 0
   FOR I = 1 TO count
      FOR J = 1 TO count
         LOCAL ref = XlBuildA1Ref( I, J, 1, 1)
         ; die Linienart und Stärke für alle 4 Seiten der Zelle definieren
         XlSetBorderStyle(ref,15,"continous")
         XlSetBorderThickness(ref,15,"thick")
         ; die Farben für linke und rechte Linie festlegen
         XlSetBorderColor(ref,1+2,RGB(colred,0,0))
         ; die Farben für obere und untere Linie festlegen
         XlSetBorderColor(ref,4+8,RGB(0,colgreen,0))
         ; Farbwerte erhöhen
         colgreen = colgreen + 8
         colred = colred + 8
      END
   END
```

```
END
```

XlSetBorderStyle

Anwendungsbereich: Excel-Fernsteuerung

Setzt den Stil von Rahmenlinien für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetBorderStyle ( TxBereich, SvRahmenLinie, TxLinenArt )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
SvRahmenLinie	Rahmenlinien der Zelle oder Gruppe.
	1 : Linie links
	2 : Linie rechts
	4 : Linie oben
	8 : Linie unten
	15 : Zellrahmen
	16 : Linie diagonal abwärts
	32 : Linie diagonal aufwärts
	48 : Kreuz
	64 : Linie Zellgruppe links
	128 : Linie Zellgruppe rechts
	256 : Linie Zellgruppe oben
	512 : Linie Zellgruppe unten
	960 : Zellgruppenrahmen
	1024 : Vertikale Linie innerhalb der Zellgruppe
	2048 : Horizontale Linie innerhalb der Zellgruppe
	3072 : Horizontale + vertikale Linie innerhalb der Zellgruppe
TxLinenArt	Art der Linie
	"noline" : keine Linie
	"continous" : Linie durchgezogen
	"dash" : Linie gestrichelt
	"dot" : Linie punktiert
	"dashdot" : Linie Strich/Punkt
	"dashdotdot" : Linie Strich/Doppelpunkt
	"double" : Linie doppelt
	"slanteddashdot" : Linie mit Schrägstrichen

Beschreibung:

Die Linientypen sind binär-codiert. Jede Linie besitzt einen eigenen Code. Es können die einzelnen Codes summiert werden, um verschiedene Rahmen zu realisieren.

Beispiele:

Es werden verschiedene Rahmenlinien und Linientypen in und um die Zellen gelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;alle Rahmenlinien entfernen
   XlSetBorderStyle("B1:J10",0x3FF,"noline")
   ;Rahmenlinie links, durchgezogen
   XlSetBorderStyle("B1:J1",1,"continous")
   ;Rahmenlinie oben, unten, durchgezogen
```

```
   XlSetBorderStyle("B3:J3",12,"continous")
   ;Rahmenlinie ganzer Rahmen gestrichelt
   XlSetBorderStyle("B5:J5",15,"dash")
   ;Rahmenlinie um die Zellgruppe, durchgezogen
   XlSetBorderStyle("B7:J10",960,"continous")
   ;Rahmenlinie diagonal abwärts, punktiert
   XlSetBorderStyle("B12:J12",16,"dot")
   ;Kreuz
   XlSetBorderStyle("B14:J14",48,"continous")
   ;Rahmenlinie diagonal aufwärts, punktiert
   XlSetBorderStyle("B16:J16",32,"dot")
END
```

XlSetBorderThickness

Anwendungsbereich: Excel-Fernsteuerung

Setzt die Rahmenlinien für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetBorderThickness ( TxBereich, SvRahmenLinie, TxRahmenStärke )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
SvRahmenLinie	Setzt die Rahmenlinien der entsprechenden Stärke.
	1 : Linie links
	2 : Linie rechts
	4 : Linie oben
	8 : Linie unten
	15 : Zellrahmen
	16 : Linie diagonal abwärts
	32 : Linie diagonal aufwärts
	48 : Kreuz
	64 : Linie Zellgruppe links
	128 : Linie Zellgruppe rechts
	256 : Linie Zellgruppe oben
	512 : Linie Zellgruppe unten
	960 : Zellgruppenrahmen
	1024 : Vertikale Linie innerhalb der Zellgruppe
	2048 : Horizontale Linie innerhalb der Zellgruppe
	3072 : Horizontale + vertikale Linie innerhalb der Zellgruppe
TxRahmenStärke	Setzt die Rahmenlinien der entsprechenden Stärke.
	"hairline" : Haarlinie
	"thin" : dünn
	"medium" : mittel
	"thick" : dick

Beschreibung:

Die Linientypen sind binär-codiert. Jede Linie besitzt einen eigenen Code. Es können die einzelnen Codes summiert werden, um verschiedene Rahmen zu realisieren.

Beispiele:

Es werden verschiedene Rahmenlinien verschiedener Stärke und Linientypen um die Zellen gelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;alle Rahmenlinien entfernen
   XlSetBorderStyle("B1:J10",0x3FF,"noline")
   ;Rahmenlinie um die Zellgruppe, durchgezogen
   XlSetBorderStyle("B2:J10",960,"continous")
   ;dicke Rahmenlinie um die Zellgruppe
   XlSetBorderThickness("B2:J10",960,"thick")
   ;Rahmenlinie um die Zellgruppe, gestrichelt
   XlSetBorderStyle("C4:I8",960,"dash")
   ;dünne Rahmenlinie um die Zellgruppe
   XlSetBorderThickness("C4:I8",960,"thin")
END
```

XlSetCellFormat

Anwendungsbereich: Excel-Fernsteuerung

Setzt das Ausgabeformat für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetCellFormat ( TxBereich, TxFormat )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
TxFormat	Formatcode

Beschreibung:

Mit dieser Funktion setzen Sie das Ausgabeformat für Zahlenwerte in Excel-Tabellen. Entscheidend ist dies besonders bei Datum/Uhrzeit - die Zahl '36526,5' entspricht beispielsweise im Excel-Zeitformat (Anzahl Tage seit 1.1.1900) dem '1.1.2000 12:00'. Damit diese Zeit auch entsprechend angezeigt und nicht einfach als reelle Zahl ausgegeben wird, muss das Ausgabeformat für die Zelle explizit auf ein Datums-Format gesetzt werden.

Diese Funktion entspricht dem Menübefehl 'Zellen formatieren/Zahlen' in Excel. Eine Übersicht über die möglichen Formatcodes finden Sie in diesem Dialog in der Kategorie 'Benutzerdefiniert' oder in der Excel-Hilfe unter dem Stichwort 'Zahlenformatcodes'. Zu beachten ist, dass die Formatcodes abhängig von der Sprache der verwendeten Excel-Version sind, dies betrifft insbesondere die häufig benötigten Codes für Datum/Uhrzeit-Komponenten und das Dezimaltrennzeichen bei Zahlen (Punkt vs. Komma).

Ausgewählte Formatcodes:

Anzeige	Deutscher Code	Englischer Code (falls verschieden)
Dezimalziffer (signifikant)	"#"	
Dezimalziffer (ggf. nichtsignifikante Null)	"0"	
Dezimaltrennzeichen	"," (Komma)	"." (Punkt)
Wissenschaftliche Notation	"E-", "e-", "e+", "e-"	
Monat	"M" (1-12) / "MM" (01-12)	"m" / "mm"
Tag	"T" (1-31) / "TT" (01-31)	"d" / "dd"
Jahr	"JJ" (00-99) / "JJJJ" (1900-9999)	"yy" / "yyyy"
Stunde	"h" (0-23) / "hh" (00-23)	
Minute	"m" (0-59) / "mm" (00-59)	
Sekunde	"s" (0-59)/ "ss" (00-59)	
Abgelaufene Zeit in Minuten	, z. B. 63:46: [mm]:ss	
Bruchteile einer Sekunde	h:mm:ss,00	h:mm:ss.00

Beispiele:

Zahl	Formatcode (deutsches Excel)	Anzeige
38710.5036	"0,00"	"38710,50"
38710.5036	"0,##"	"38710,5"
38710.5036	"0,000E+0"	"3,871E+4"
38710.5036	"TT.MM.JJ hh:mm"	"24.12.05 12:05"
38710.5036	"TT.MM.JJJJ hh:mm:ss,000"	"24.12.2005 12:05:11,040"

Beispiele:

Eine neue Excel-Datei mit einem Tabellenblatt wird erzeugt und ein Datensatz in diese Tabelle übertragen. Die erste Spalte wird als Zeitspalte mit absoluten Datum/Uhrzeit-Angaben konfiguriert, die zweite Spalte enthält die eigentlichen Werte des Datensatzes.

Das Ausgabeformat beider Spalten wird explizit festgelegt. Die so erstellte Datei wird dann gespeichert.

```
XlWbNew("")
time = XlCreateTimeLine(channel1, 1, 0)
XlSetValues( "A1", 0, time, 1)
Count = Leng?(time)
Range = XlBuildA1Ref(1, 1, count, 1)
XlSetCellFormat(Range, "TT.MM.JJ hh:mm:ss,0")
XlSetValues( "B1", 0, channel1, 0)
```

```
Range = XlBuildA1Ref(1, 2, count, 1)
XlSetCellFormat(Range, "0,00")
XlWbSave( "c:\results\report", 0)
XlQuit()
```

Wie oben, die Zeitspalte wird hier jedoch relativ konfiguriert (Sekunden seit Triggerzeit). Die Zeitspalte wird mit 1ms Auflösung ausgeben, die Datenspalte mit maximal 2 Nachkommastellen.

```
XlWbNew("")
time = XlCreateTimeLine(channel1, 0, 0)
XlSetValues( "A1", 0, time, 0)
Count = Leng?(time)
Range = XlBuildA1Ref(1, 1, count, 1)
XlSetCellFormat(Range, "0,000")
XlSetValues( "B1", 0, channel1, 0)
Range = XlBuildA1Ref(1, 2, count, 1)
XlSetCellFormat(Range, "0,##")
XlWbSave( "c:\results\report", 0)
XlQuit()
```

Siehe auch:

XlSetText, XlSetValue

XlSetColor

Anwendungsbereich: Excel-Fernsteuerung

Setzt die Farben für die Zelldarstellung der angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetColor ( TxBereich, TxZellEigenschaft, SvFarbe )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
TxZellEigenschaft	Eigenschaft für die angegebenen Zellen.
	"background" : Farbe des Hintergrunds der Zelle.
	"text" : Farbe des Texts in der Zelle.
SvFarbe	RGB-Wert

Beispiele:

Es wird die Text- und Hintergrundfarbe eines ausgewählten Bereiches festgelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Matrix für die Tabelle erzeugen
   Matrix = MatrixInit( 10, 10, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   ; Hintergrundfarbe setzen
   XlSetColor("A1:J10","background",RGB(250,250,150))
   ; Textfarbe setzen
   XlSetColor("A1:J10","text",RGB(0,0,250))
END
```

XlSetColumnWidth

Anwendungsbereich: Excel-Fernsteuerung

Definiert die Breite der Spalten als Vielfaches einer Basisbreite. Für proportionale Schriftarten ist die Basisbreite die Breite des Zeichens "0", anderenfalls die Breite eines Zeichens im Normalformat.

Deklaration:

```
XlSetColumnWidth ( TxBereich, SvSpaltenBreite )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
SvSpaltenBreite	Definiert die Spaltenbreite. Der Wert -1 setzt die Breite auf "automatisch".

Beispiele:

Es werden Höhe und Breite von Tabellenzellen festgelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ; Textausgabe
   XlSetText("A1", "example text 1")
   XlSetText("B1", "example text 2")
   XlSetText("C1", "example text 3")
   ; Zellbreite als Vielfaches der Breite des 1.Zeichens
   XlSetColumnWidth("A1:C1", 20)
   ; Zellhöhe in Points (Pixel) festlegen
   XlSetRowHeight("A1:C1", 50)
END
```

XlSetConditionColor

Anwendungsbereich: Excel-Fernsteuerung

Setzt eine Farbe für die Zelldrastellung unter einer Bedingung für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetConditionColor ( TxBereich, TxZellEigenschaft, TxConditionType, TxAusdruck, TxConditionOperator, SvParameter1, SvParameter2, SvFarbe )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
TxZellEigenschaft	Zelleigenschaft für die die Farbe bei erfüllter Bedingung verwendet wird.
	"background" : Farbe des Hintergrunds der Zelle.
	"text" : Farbe des Texts in der Zelle.
	"border" : Farbe des Rahmen der Zelle.
TxConditionType	Bezug der Bedingung
	"cellcontent" : Die Bedingung bezieht sich auf den Inhalt der Zelle.
	"expression" : Die Bedingung ist der ausgewertete Ausdruck.
	"reset" : Alle Bedingungen für die Zellen werden zurückgesetzt.
TxAusdruck	Ausdruck der für die Bedingung ausgewertet wird (z.B. A1>0).
TxConditionOperator	Operator für die Auswertung des Zellinhaltes
	"between" : Der Zellinhalt liegt zwischen den Parametern 1 und 2
	"notbetween" : Der Zellinhalt liegt nicht zwischen den Parametern 1 und 2
	"equal" : Der Zellinhalt ist gleich dem Parameter 1
	"notequal" : Der Zellinhalt ist nicht gleich dem Parameter 1
	"greater" : Der Zellinhalt ist größer als Parameter 1
	"less" : Der Zellinhalt ist kleiner als Parameter 1
	"greaterequal" : Der Zellinhalt ist größer oder gleich dem Parameter 1
	"lessequal" : Der Zellinhalt ist kleiner oder gleich dem Parameter 1
SvParameter1	Vergleichswert oder unterer Grenzwert
SvParameter2	Oberer Grenzwert
SvFarbe	Farbe die bei erfüllter Bedingung verwendet wird

Beispiele:

Es wird die Text-, Hintergrund- und Rahmenfarbe eines ausgewählten Bereiches unter einer Bedingung festgelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Matrix für die Tabelle erzeugen
   Matrix = MatrixInit( 10, 10, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   ; Hintergrundfarbe setzen
   XlSetConditionColor("A1:J10","background","cellcontent","","equal",1,0,RGB(250,250,0))
   ; Textfarbe setzen
   XlSetConditionColor("A1:J10","text","cellcontent","","equal",1,0,RGB(250,0,0))
   ; Rahmenfarbe setzen
   XlSetConditionColor("A1:J10","border","expression","=A1=1","",0,0,RGB(250,0,0))
END
```

XlSetFontSize

Anwendungsbereich: Excel-Fernsteuerung

Setzt die Schriftgröße für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetFontSize ( TxBereich, SvGröße )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
SvGröße	Schriftgröße der Zelle in Punkten (Pixel).

Beispiele:

Es wird eine schrittweise Vergrößerung des Textes vorgenommen.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Matrix für die Tabelle erzeugen
   Matrix = MatrixInit(5, 5, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues(ref, 0, Matrix[I], 0)
   END
   LOCAL TextSize = 8
   FOR I = 1 TO count
      FOR J = 1 TO count
         LOCAL ref = XlBuildA1Ref( I, J, 1, 1)
         XlSetFontSize(ref, TextSize)
         TextSize = TextSize + 1;
      END
   END
END
```

XlSetFontStyle

Anwendungsbereich: Excel-Fernsteuerung

Setzt den Stil der Schrift für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetFontStyle ( TxBereich, TxStil )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
TxStil	Stil der Schrift
	"reset" : Setzt auf den Ausgangszustand zurück
	"bold" : Text fett
	"italic" : Text 'italic'
	"underlined" : Text einfach unterstrichen
	"underlineddouble" : Text doppelt unterstrichen
	"underlineddoubleclosed" : Text doppelt unterstrichen, direkt unter dem Text
	"subscript" : subscript
	"superscript" : superscript
	"strikethrough" : Text durchgestrichen.

Beschreibung:

Die Funktion setzt einen Stil für den Text in einer Zelle. Bestimmte Stile können kombiniert werden. "Reset" setzt alle Stile zurück.

Beispiele:

Es werden Textgröße und der Stil des Textes festgelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Matrix für die Tabelle erzeugen
   Matrix = MatrixInit( 10, 10, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   ; Textgröße setzen
   XlSetFontSize("A1:J1",10)
   XlSetFontSize("A5:J5",20)
   ; Textstil setzen
   XlSetFontStyle("A2:J2","italic")
   XlSetFontStyle("A3:J3","bold")
   XlSetFontStyle("A4:J4","strikethrough")
   XlSetFontStyle("A5:J5","underlined")
END
```

XlSetRowHeight

Anwendungsbereich: Excel-Fernsteuerung

Definiert die Höhe der Zeilen in Pixel für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetRowHeight ( TxBereich, SvZeilenHöhe )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
SvZeilenHöhe	Definiert die Zeilenhöhe.

Beispiele:

Es werden Höhe und Breite von Tabellenzellen festgelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ; Textausgabe
   XlSetText("A1", "example text 1")
   XlSetText("B1", "example text 2")
   XlSetText("C1", "example text 3")
   ; Zellbreite als Vielfaches der Breite des 1.Zeichens
   XlSetColumnWidth("A1:C1", 20)
   ; Zellhöhe in Points (Pixel) festlegen
   XlSetRowHeight("A1:C1", 50)
END
```

XlSetText

Anwendungsbereich: Excel-Fernsteuerung

Belegt die angegebene Tabellenzelle im aktiven Tabellenblatt mit einem Text.

Deklaration:

```
XlSetText ( TxZelle, TxInhalt ) -> Erfolg
```

Parameter:

TxZelle	Zu beschreibende Zelle. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
TxInhalt	Neuer Inhalt
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Eine Excel-Datei wird geöffnet und Daten in 2 Spalten übertragen. In der ersten Zeile steht jeweils der Name des Kanals. Die aktualisierte Datei wird gedruckt und unter neuem Namen gespeichert.

```
IF NOT (XlWbOpen("c:\Templates\Template.xlsx"))
   BoxMessage("Fehler beim Öffnen", GetLastError(), "!1")
ELSE
   XlSetText("B1", "Channel1")
   XlSetValues("B2", 0, Channel1, 0)
   XlSetText("C1", "Channel2")
   XlSetValues("C2", 0, Channel2, 0)
   XlWbPrint()
   IF NOT(XlWbSave("c:\Results\Report.xlsx", 0))
      BoxMessage("Fehler beim Speichern", GetLastError(), "!1")
   END
   XlQuit()
END
```

Siehe auch:

XlGetText, XlGetValue, XlSetValue, XlGetValues

XlSetTextAlignment

Anwendungsbereich: Excel-Fernsteuerung

Setzt die Ausrichtung des Textes die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetTextAlignment ( TxBereich, TxAusrichtung )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
TxAusrichtung	Position des Textes innerhalb der Zelle.
	"left" : links
	"hcenter" : zentral horizontal
	"right" : rechts
	"top" : oben
	"vcenter" : zentral vertikal
	"bottom" : unten

Beispiele:

Es werden verschiedene Orientierungen und Ausrichtungen für die Zelltexte festgelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Matrix für die Tabelle erzeugen
   Matrix = 100*MatrixInit( 4, 4, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   ; alle Zelltexte horizontal und vertikal zentieren
   XlSetTextAlignment("A1:D4", "hcenter")
   XlSetTextAlignment("A1:D4", "vcenter")
   ; alle Zelltexte der Diagonalen drehen
   XlSetTextOrientation("A1", 45)
   XlSetTextOrientation("B2", 90)
   XlSetTextOrientation("C3", -90)
   XlSetTextOrientation("D4", -45)
END
```

XlSetTextArray

Anwendungsbereich: Excel-Fernsteuerung

Setzt den Inhalt einer Spalte oder Zeile im aktiven Tabellenblatt.

Deklaration:

```
XlSetTextArray ( TxStartZelle, SvZeileOderSpalte, TaTextArray ) -> Erfolg
```

Parameter:

TxStartZelle	An dieser Zelle beginnt das Schreiben. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
SvZeileOderSpalte	Gibt an, ob in vertikaler Richtung (Spalte) oder horizontaler Richtung (Zeile) geschrieben werden soll.
	0 : Vertikal (Spalte)
	1 : Horizontal (Zeile)
TaTextArray	Zu schreibende Textdaten
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Es werden Textarrays als Spalten- und Zeilenvektoren angelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   LOCAL headlines = TxArrayCreate(3)
   headlines[1]="1.Headline"
   headlines[2]="2.Headline"
   headlines[3]="3.Headline"
   LOCAL data = TxArrayCreate(3)
   data[1]="1.Dataset"
   data[2]="2.Dataset"
   data[3]="3.Dataset"
   XlSetTextArray("B1",1,headlines)
   XlSetTextArray("A2",0,data)
   ;Matrix für die Tabelle erzeugen
   LOCAL Matrix = MatrixInit( 3, 3, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   LOCAL I
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 2, I+1, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   XlSetColumnWidth("A1:D1",15)
   XlSetTextAlignment("B1:D4","hcenter")
END
```

XlSetTextOrientation

Anwendungsbereich: Excel-Fernsteuerung

Setzt die Orientierung des Textes die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetTextOrientation ( TxBereich, SvOrientierung )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
SvOrientierung	Orientierung des Textes innerhalb der Zelle.

Beispiele:

Es werden verschiedene Orientierungen und Ausrichtungen für die Zelltexte festgelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Matrix für die Tabelle erzeugen
   Matrix = 100*MatrixInit( 4, 4, "I")
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      LOCAL ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   ; alle Zelltexte horizontal und vertikal zentieren
   XlSetTextAlignment("A1:D4", "hcenter")
   XlSetTextAlignment("A1:D4", "vcenter")
   ; alle Zelltexte der Diagonalen drehen
   XlSetTextOrientation("A1", 45)
   XlSetTextOrientation("B2", 90)
   XlSetTextOrientation("C3", -90)
   XlSetTextOrientation("D4", -45)
END
```

XlSetTextShrinkToFit

Anwendungsbereich: Excel-Fernsteuerung

Aktiviert die Textverkleinerung zur Anpassung an die Zellgröße für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetTextShrinkToFit ( TxBereich, TxShrink )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
TxShrink	Anpassung der Größe des Textes an die Größe der Zelle.
	"noshrink" : Text nicht anpassen
	"shrink" : Text anpassen

Beispiele:

Es werden verschiedene Anpassungen für Anzeige eines Zelltextes definiert.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ; Textausgabe
   XlSetText("A1", "example text 1")
   XlSetText("B1", "example text 2")
   XlSetText("C1", "example text 3")
   ; Zellgröße als Vielfaches der Breite des 1.Zeichens
   XlSetColumnWidth("A1:C1", 8)
   ; einen Textumbruch definieren
   XlSetTextWrap("A1", "wrap")
   ; eine Textanpassung festlegen
   XlSetTextShrinkToFit("B1", "shrink")
END
```

XlSetTextWrap

Anwendungsbereich: Excel-Fernsteuerung

Aktiviert den Textumbruch-Modus für die angegebenen Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSetTextWrap ( TxBereich, TxUmbruch )
```

Parameter:

TxBereich	Zu formatierender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.
TxUmbruch	Definiert den Umbruch des Textes innerhalb der Zelle.
	"nowrap" : Text nicht umbrechen
	"wrap" : Text umbrechen

Beispiele:

Es werden verschiedene Anpassungen für Anzeige eines Zelltextes definiert.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ; Textausgabe
   XlSetText("A1", "example text 1")
   XlSetText("B1", "example text 2")
   XlSetText("C1", "example text 3")
   ; Zellgröße als Vielfaches der Breite des 1.Zeichens
   XlSetColumnWidth("A1:C1", 8)
   ; einen Textumbruch definieren
   XlSetTextWrap("A1", "wrap")
   ; eine Textanpassung festlegen
   XlSetTextShrinkToFit("B1", "shrink")
END
```

XlSetValue

Anwendungsbereich: Excel-Fernsteuerung

Belegt die angegebene Zelle im aktiven Tabellenblatt mit einer Zahl.

Deklaration:

```
XlSetValue ( TxZelle, Wert, Format ) -> Erfolg
```

Parameter:

TxZelle	Zu beschreibende Zelle. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
Wert	Zu übertragender Wert
Format	Datenformat des zu übertragenden Wertes.
	0 : Der Wert wird unkonvertiert geschrieben.
	1 : Der Wert wird als Datum/Uhrzeit im imc-Zeitformat interpretiert (Anzahl der Sekunden seit 1.1.1980) und in das Excel-Datum/Uhrzeit-Format konvertiert (Anzahl der Tage seit 1.1.1900).
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Bei Angabe der Option 1 (Datum/Uhrzeit) wird die aktuelle Zellenformatierung der Zielzelle geprüft. Falls diese auf 'Standard' steht, wird das Ausgabeformat auf Datum/Uhrzeit eingestellt.

Beispiele:

Der Datensatz 'Channel1' wird in die erste Spalte einer neuen Excel-Datei geschrieben. Die erste Zeile enthält den Namen, es folgen die Triggerzeit und die Abtastzeit. Die eigentlichen Daten beginnen in der 4. Zeile.

```
XlWbNew("")
XlSetText( "A1", "Channel1")
XlSetValue("A2", Time?(Channel1), 1)
XlSetCellFormat("A2", "TT.MM.JJ hh:mm:ss,000")
XlSetValue("A3", xdel?(Channel1), 0)
XlSetValues("A4", 0, Channel1, 0)
XlWbSave( "c:\results\report", 0)
XlQuit()
```

Siehe auch:

XlGetValue, XlGetText, XlSetText, XlGetValues

XlSetValues

Anwendungsbereich: Excel-Fernsteuerung

Setzt den Inhalt einer Spalte oder Zeile im aktiven Tabellenblatt.

Deklaration:

```
XlSetValues ( TxStartZelle, ZeileOderSpalte, Daten, Format ) -> Erfolg
```

Parameter:

TxStartZelle	An dieser Zelle beginnt das Schreiben. Entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle.
ZeileOderSpalte	Gibt an, ob in vertikaler Richtung (Spalte) oder horizontaler Richtung (Zeile) geschrieben werden soll.
	0 : Vertikal (Spalte)
	1 : Horizontal (Zeile)
Daten	Zu schreibende Daten
Format	Datenformat der zu übertragenden Daten.
	0 : Die Werte werden unkonvertiert geschrieben.
	1 : Die Werte werden als Datum/Uhrzeit im imc-Zeitformat interpretiert (Anzahl der Sekunden seit 1.1.1980) und in das Excel-Datum/Uhrzeit-Format konvertiert (Anzahl der Tage seit 1.1.1900).
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Der zu übertragende Datensatz muss equidistant abgetastet und unstrukturiert sein. Bei strukturierten Datensätzen (Komponenten (XY, komplex), Segmente, Events) müssen die einzelnen Elemente ggf. nacheinander übertragen werden.

Bei Angabe der Option 1 (Datum/Uhrzeit) wird die aktuelle Zellenformatierung des Zielbereichs geprüft. Falls diese auf 'Standard' steht, wird das Ausgabeformat auf Datum/Uhrzeit eingestellt.

Zum Schreiben von Excel-Dateien können Sie auch die Funktionen FileOpenXLS() und insbesondere FileOpenXLS2() verwenden. Diese sind im Allgemeinen deutlich leistungsfähiger und schneller, dafür aber weniger flexibel als die entsprechenden Funktionen in diesem Kit.

Beispiele:

Eine neue Excel-Datei mit einem Tabellenblatt wird erzeugt und ein Datensatz in diese Tabelle übertragen. Die zweite Spalte hat die Überschrift 'Data', gefolgt von den Werten des Datensatzes. Die erste Spalte hat die Überschrift 'Time', gefolgt von den korrespondierenden Datum/Uhrzeit-Angaben. Das Ausgabeformat beider Spalten wird explizit festgelegt. Die so erstellte Datei wird dann gespeichert.

```
XlWbNew("")
; Spalte mit Datum/Uhrzeit füllen
XlSetText("A1", "Time")
time = XlCreateTimeLine(channel1, 1, 0)
XlSetValues( "A2", 0, time, 1)
Count = Leng?(time)
Range = XlBuildA1Ref(2, 1, count, 1)
XlSetCellFormat(Range, "TT.MM.JJ hh:mm:ss,0")
; Datenspalte füllen
XlSetText("B1", "Data")
XlSetValues( "B2", 0, channel1, 0)
Range = XlBuildA1Ref(2, 2, count, 1)
XlSetCellFormat(Range, "0,00")
XlWbSave( "c:\results\report", 0)
XlQuit()
```

Die Events eines eventierten Datensatzes werden nacheinander in aufeinanderfolgende Spalten einer Excel-Tabelle übertragen.

```
XlWbNew("")
count = EventNum?(channel)
;      Falls es sich um einen segmentierten Datensatz handelt:
;      count = Leng?(Channel) / SegLen?(Channel)
FOR I = 1 TO count
   ref = XlBuildA1Ref( 1, I, 1, 1)
   XlSetValues( ref, 0, channel[I], 0)
END
XlWbSave("c:\results\report", 0)
XlQuit()
```

Siehe auch:

XlGetValues2, XlGetValue, XlGetText

XlSheetActivate

Anwendungsbereich: Excel-Fernsteuerung

Ein Blatt der aktuellen Arbeitsmappe wird aktiviert.

Deklaration:

```
XlSheetActivate ( TxTitelOderIndex ) -> Erfolg
```

Parameter:

TxTitelOderIndex	Titel oder Index des zu aktivierenden Blattes. Das erste Blatt hat den Index 1.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Viele Funktionen dieses Kits wirken auf das aktive Blatt der aktiven Arbeitsmappe. Das aktive Blatt kann sich ändern, wenn mit XlSheetAdd()/XlSheetDelete() ein Blatt erzeugt oder gelöscht wird, wenn mit XlSheetActivate() explizit ein neues Blatt aktiviert wird oder der Anwender manuell eine andere Seite aktiviert (z.B. Anklicken des entsprechenden Reiters der Arbeitsmappe).

Beispiele:

Eine Excel-Datei wird geöffnet und Daten in 2 Tabellenblätter übertragen. Die aktualisierte Datei wird gedruckt und unter neuem Namen gespeichert.

```
IF NOT (XlWbOpen("c:\Templates\Template.xlsx"))
   BoxMessage("Fehler beim Öffnen", GetLastError(), "!1")
ELSE
   XlSheetActivate("Table1")
   XlSetText("B1", "Channel1")
   XlSetValues("B2", 0, Channel1, 0)
   XlSheetActivate("Table2")
   XlSetText("B1", "Channel2")
   XlSetValues("B2", 0, Channel2, 0)
   XlWbPrint()
   IF NOT(XlWbSave("c:\Results\Report.xlsx", 0))
      BoxMessage("Fehler beim Speichern", GetLastError(), "!1")
   END
   XlQuit()
END
```

Siehe auch:

XlSheetAdd, XlSheetGetActive

XlSheetAdd

Anwendungsbereich: Excel-Fernsteuerung

Ein Blatt wird in die aktive Arbeitsmappe eingefügt.

Deklaration:

```
XlSheetAdd ( TxTitel, EwPos, TxVorlage ) -> Erfolg
```

Parameter:

TxTitel	Titel des neuen Blattes.
EwPos	Gibt die Position an, an der das neue Blatt eingefügt wird. Geben Sie eine 0 an, wenn Sie das Blatt an letzter Position anhängen wollen. Mit einer 1 wird das neue Blatt das erste Blatt der Arbeitsmappe usw.
TxVorlage	Wenn Sie einen leeren Text angeben, wird ein leeres Standard-Tabellenblatt erzeugt. Sie können auch den kompletten Pfadnamen einer Excel-Datei angeben, die dann als Vorlage für die neue Seite verwendet wird.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Eine neue Excel-Datei wird erzeugt und mit den Daten einer FAMOS-Datengruppe 'MyGroup' befüllt. Das erste Blatt wird mit einigen allgemeinen Angaben gefüllt, dann wird für jeden Kanal der Gruppe ein neues Tabellenblatt angelegt und die erste Spalte mit den Werten dieses Kanals gefüllt.

Für Deck- und Datenblätter wird jeweils eine vorgefertigte Vorlage verwendet, die feste Texte und Zellenformatierungen enthält.

```
IF XlWbNew("c:\templates\firstpage.xlsx")
   XlVisible(1)
   XlSetText( "A3", "Name: Mike Smith")
   XlSetText( "A4", "Date: " + TimeToText( TimeSystem?(),0))
   FOR I = 1 TO GrChanNum?(MyGroup)
      XlSheetAdd( GrChanName?(MyGroup, I), 0, "c:\templates\datapage.xlsx")
      XlSetValues( "A1", 0, MyGroup:[I], 0)
   END
   XlWbSave( "c:\results\report", 0)
   XlQuit()
END
```

Siehe auch:

XlSheetDelete

XlSheetDelete

Anwendungsbereich: Excel-Fernsteuerung

Ein Blatt wird aus der aktiven Arbeitsmappe entfernt.

Deklaration:

```
XlSheetDelete ( TxTitelOderIndex ) -> Erfolg
```

Parameter:

TxTitelOderIndex	Titel oder Index des zu löschenden Blattes. Das erste Blatt hat den Index 1.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Eine Excel-Vorlage mit 2 Tabellenblättern wird geladen. Für die aktuelle Aufgabe wird allerdings nur das erste Tabellenblatt benötigt, die 2. Seite wird daher vor dem Speichern der aktualisierten Datei gelöscht.

```
XlWbOpen("c:\VorlageMit2Seiten.xlsx")
;...aktualisiere erstes Tabellenblatt...
XlSheetDelete(2)
; oder z.B.: XlSheetDelete("Table2")
XlWbSave("c:\results\report", 0)
```

Siehe auch:

XlSheetAdd

XlSheetExist

Anwendungsbereich: Excel-Fernsteuerung

Prüft, ob in der aktiven Arbeitsmappe ein Blatt mit gegebenem Titel vorhanden ist.

Deklaration:

```
XlSheetExist ( TxTitel ) -> Ergebnis
```

Parameter:

TxTitel	Titel des gesuchten Blattes.
Ergebnis	1 wenn ein solches Blatt existiert, 0 falls nicht. -1 im Fehlerfall, die Fehlerursache kann mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Eine Excel-Datei wird geladen und das Vorhandensein eines Arbeitsblattes mit gegebenem Titel geprüft. Falls nicht vorhanden, wird eine Fehlermeldung ausgegeben.

```
XlWbOpen("c:\results\report.xlsx")
ok = XlSheetExist("Data from 20.07.2018")
IF NOT ok
   BoxMessage( "Error", "Unexpected excel file", "!1")
END
```

Siehe auch:

XlSheetGetActive, XlSheetGetTitle, XlSheetGetCount, XlWbExist

XlSheetGetActive

Anwendungsbereich: Excel-Fernsteuerung

Das aktive Blatt der aktiven Arbeitsmappe wird ermittelt.

Deklaration:

```
XlSheetGetActive ( ) -> TxTitel
```

Parameter:

TxTitel	Titel des aktiven Blatts.

Beschreibung:

Viele Funktionen dieses Kits wirken auf das aktive Blatt der aktiven Arbeitsmappe. Das aktive Blatt kann sich ändern, wenn mit XlSheetAdd()/XlSheetDelete() ein Blatt erzeugt oder gelöscht wird, wenn mit XlSheetActivate() explizit ein neues Blatt aktiviert wird oder der Anwender manuell eine andere Seite aktiviert (z.B. Anklicken des entsprechenden Reiters der Arbeitsmappe).

Der hier ermittelte Titel entspricht genau dem angezeigten Namen auf dem zugehörigen Reiter der Arbeitsmappe.

Im Fehlerfall wird ein leerer Text zurückgegeben. Die Ursache kann dann mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Der Anwender wählt eine Excel-Datei zum Öffnen. Vom 2. Tabellenblatt der Datei werden aus der ersten Spalte die enthaltenen Werte ausgelesen. Der erzeugte Datensatz erhält den Namen des Tabellenblatts.

```
filename = DlgFileName("c:\results", "xlsx", "", 0)
IF XlWbOpen(filename)
   XlSheetActivate(2)
   SheetName = XlSheetGetActive()
   <SheetName> = XlGetValues( "A1", 0, 0, 0)
END
```

Siehe auch:

XlSheetAdd, XlSheetActivate

XlSheetGetCount

Anwendungsbereich: Excel-Fernsteuerung

Die Anzahl der Blätter in der aktiven Arbeitsmappen wird ermittelt.

Deklaration:

```
XlSheetGetCount ( ) -> Anzahl
```

Parameter:

Anzahl	Anzahl der Bätter. -1 im Fehlerfall, die Fehlerursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Beispiele:

Eine Excel-Datei wird geladen und die Titel aller Tabellenblätter in das Ausgabefenster ausgegeben.

```
IF XlWbOpen("z:\tmp\results.xlsx")
   sheetCount = XlSheetGetCount()
   FOR i = 1 TO sheetCount
      BoxOutput( XlSheetGetTitle(i), EMPTY, "", 1)
   END
END
```

Eine Arbeitsmappe wird geöffnet und eine Kopie der letzten Seite angehängt.

```
IF XlWbOpen("z:\tmp\results.xlsx")
   sheetCount = XlSheetGetCount()
   XlSheetInsertCopy("", sheetCount, 0, "Tabelle " + TForm(sheetCount+1, ""))
   ;...
END
```

Siehe auch:

XlSheetGetActive, XlSheetGetTitle, XlSheetExist

XlSheetGetTitle

Anwendungsbereich: Excel-Fernsteuerung

Der Titel eines Blattes in der aktiven Arbeitsmappe wird ermittelt.

Deklaration:

```
XlSheetGetTitle ( Index ) -> TxTitel
```

Parameter:

Index	Index des Blattes. Liegt zwischen 1 und der Zahl der Blätter in der aktiven Mappe.
TxTitel	Titel des Blattes. Leerer Text im Fehlerfall, die Fehlerursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die Anzahl der Blätter und somit der maximale Wert für [Index] kann mit der Funktion XlSheetGetCount() ermittelt werden.

Beispiele:

Eine Excel-Datei wird geladen und die Titel aller Tabellenblätter in das Ausgabefenster ausgegeben.

```
IF XlWbOpen("z:\tmp\results.xlsx")
   sheetCount = XlSheetGetCount()
   FOR i = 1 TO sheetCount
      BoxOutput( XlSheetGetTitle(i), EMPTY, "", 1)
   END
END
```

Siehe auch:

XlSheetGetActive, XlSheetGetCount, XlSheetExist

XlSheetInsertCopy

Anwendungsbereich: Excel-Fernsteuerung

Die Kopie eines Blattes wird in die aktive Arbeitsmappe eingefügt.

Deklaration:

```
XlSheetInsertCopy ( TxQuellArbeitsmappe, QuellBlattTitelOderIndex, EwPos [, TxBlattTitel] ) -> Erfolg
```

Parameter:

TxQuellArbeitsmappe	Name der Arbeitsmappe, die das zu kopierende Blatt enthält. Leerer String, wenn die aktuelle Arbeitsmappe verwendet werden soll.
QuellBlattTitelOderIndex	Titel oder Index des zu kopierenden Blattes. Das erste Blatt hat den Index 1.
EwPos	Gibt die Position an, an der das neue Blatt eingefügt wird. Geben Sie eine 0 an, wenn Sie das Blatt an letzter Position anhängen wollen. Mit einer 1 wird das neue Blatt das erste Blatt der Arbeitsmappe usw.
TxBlattTitel	Titel des neuen Blattes. Wenn leer, wird der von Excel automatisch vergebene Name verwendet. (optional , Standardwert: "")
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Wenn das einzufügende Blatt aus einer anderen Mappe stammt (Parameter [TxQuellArbeitsMappe] ist nicht leer), so muß auch diese Vorlagenmappe vorher mit XlWbOpen() geladen worden sein.

Beispiele:

Eine Arbeitsmappe wird geöffnet und eine Kopie der letzten Seite angehängt.

```
IF XlWbOpen("z:\tmp\results.xlsx")
   sheetCount = XlSheetGetCount()
   XlSheetInsertCopy("", sheetCount, 0, "Tabelle " + TForm(sheetCount+1, ""))
   ;...
END
```

Das erste Blatt einer Vorlagendatei "templates.xlsx" wird an die Arbeitsmappe 'results.xlsx' angehängt und die 2. Spalte mit den Werten eines vorher geladenen Datensatzes 'channel' gefüllt. Das neue Arbeitsblatt erhält als Titel das Trigger-Datum des Datensatzes. Anschließend wird die aktualisierte Mappe gespeichert.

```
IF XlWbOpen( "z:\tmp\templates.xlsx")
   IF XlWbOpen( "z:\tmp\results.xlsx")
      XlSheetInsertCopy( "templates.xlsx", 1, 0, TimeToText(Time?(channel), 1))
      XlSetValues("B2", 0, channel, 0)
      XlWbSave("", 0)
   END
   XlQuit()
END
```

Siehe auch:

XlSheetDelete, XlSheetAdd, XlSheetMove

XlSheetMove

Anwendungsbereich: Excel-Fernsteuerung

Die Position eines Blattes in der aktiven Arbeitsmappe wird geändert oder ein Blatt aus einer anderen Mappe in die aktive Mappe verschoben.

Deklaration:

```
XlSheetMove ( TxQuellArbeitsmappe, QuellBlattTitelOderIndex, EwPos [, TxBlattTitel] ) -> Erfolg
```

Parameter:

TxQuellArbeitsmappe	Name der Arbeitsmappe, die das zu verschiebende Blatt enthält. Leerer String, wenn die aktuelle Arbeitsmappe verwendet werden soll.
QuellBlattTitelOderIndex	Titel oder Index des zu verschiebenden Blattes. Das erste Blatt hat den Index 1.
EwPos	Gibt die neue Position des verschobenen Blattes an in der aktiven Arbeitsmappe an. Geben Sie eine 0 an, wenn Sie das Blatt an letzter Position anhängen wollen. Mit einer 1 wird das Blatt das erste Blatt der Arbeitsmappe usw.
TxBlattTitel	Titel des neuen Blattes. Wenn leer, wird der von Excel automatisch vergebene Name verwendet. (optional , Standardwert: "")
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Wenn das einzufügende Blatt aus einer anderen Mappe stammt (Parameter [TxQuellArbeitsMappe] ist nicht leer), so muß auch diese Mappe vorher mit XlWbOpen() geladen worden sein.

Beispiele:

Eine Arbeitsmappe wird geöffnet und die letzte Seite an die erste Position verschoben

```
IF XlWbOpen("z:\tmp\results.xlsx")
   sheetCount = XlSheetGetCount()
   XlSheetMove("", sheetCount, 1)
   ;...
END
```

Das erste Blatt einer Arbeitsmappe 'results1.xlsx' wird in die Arbeitsmappe 'results2.xlsx' an die letzte Position verschoben. Anschließend werden beide Mappen gespeichert.

```
IF XlWbOpen("z:\tmp\results1.xlsx")
   IF XlWbOpen("z:\tmp\results2.xlsx")
      XlSheetMove("results1.xlsx", 1, 0)
      XlWbSave("", 0)
      XlWbActivate("results1.xlsx")
      XlWbSave("", 0)
   END
   XlQuit()
END
```

Siehe auch:

XlSheetDelete, XlSheetAdd, XlSheetInsertCopy

XlSheetPrint

Anwendungsbereich: Excel-Fernsteuerung

Das aktive Blatt der aktiven Arbeitsmappe wird gedruckt.

Deklaration:

```
XlSheetPrint ( ) -> Erfolg
```

Parameter:

Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Eine Excel-Datei wird geladen und das 2. und 3. Blatt der Arbeitsmappe ausgedruckt.

```
filename = "c:\results\report.xlsx"
IF XlWbOpen(filename)
   XlSheetActivate(2)
   XlSheetPrint()
   XlSheetActivate(3)
   XlSheetPrint()
END
```

Siehe auch:

XlWbOpen, XlWbPrint

XlSheetRename

Anwendungsbereich: Excel-Fernsteuerung

Umbenennung eines Blattes in der aktiven Arbeitsmappe.

Deklaration:

```
XlSheetRename ( TxTitelOderIndex, TxNeuerTitel ) -> Erfolg
```

Parameter:

TxTitelOderIndex	Titel oder Index des zu umzubenennenden Blattes. Das erste Blatt hat den Index 1.
TxNeuerTitel	Neuer Titel
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Beispiele:

Eine Excel-Datei wird geladen und aktualisiert. Anschließend wird das erste Tabellenblatt mit dem aktuellen Datum benannt und die Datei unter neuem Namen gespeichert.

```
XlWbOpen("z:\tmp\results.xlsx")
; ... diverse Aktualisierungen
TxDate = TimeToText( TimeSystem?(), 1)
ok = XlSheetRename(1, TxDate)
XlWbSave("z:\tmp\results_"+ TxDate + ".xlsx", 0)
```

Siehe auch:

XlSheetGetTitle, XlSheetGetCount, XlSheetAdd

XlSheetSetColumnStandardWidth

Anwendungsbereich: Excel-Fernsteuerung

Definiert die Standardbreite für Tabellenspalten als ein Vielfaches der Breite eines Zeichens.

Deklaration:

```
XlSheetSetColumnStandardWidth ( SvStandardBreite )
```

Parameter:

SvStandardBreite	Breite einer Tabellenspalte

Beispiele:

Ein neues Excel-Dokument wird erzeugt. Die neue Spaltenbreite für alle Spalten wird festgelegt.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ;Spaltenbreite für alle Spalten festlegen
   XlSheetSetColumnStandardWidth(25)
END
```

XlSheetSetOption

Anwendungsbereich: Excel-Fernsteuerung

Setzt Anzeige- und Druck-Optionen für das aktive Blatt.

Deklaration:

```
XlSheetSetOption ( TxOptionsName, TxNeueEinstellung ) -> Erfolg
```

Parameter:

TxOptionsName	Bezeichnung der Option
	"GridLines.Show" : Gitternetz anzeigen
	"GridLines.Print" : Gitternetz drucken
	"Headings.Show" : Spalten-/Zeilenköpfe anzeigen
	"Headings.Print" : Spalten-/Zeilenköpfe drucken
	"Page.Orientation" : Definiert die Orientierung der Seite (portrait/landscape)
TxNeueEinstellung	[TxOptionsName] bestimmt die möglichen Vorgaben:
	"GridLines.Show" : Gitternetz anzeigen
	"on" ja **"off"** nein
	"GridLines.Print" : Gitternetz drucken
	"on" ja **"off"** nein
	"Headings.Show" : Spalten-/Zeilenköpfe anzeigen
	"on" ja **"off"** nein
	"Headings.Print" : Spalten-/Zeilenköpfe drucken
	"on" ja **"off"** nein
	"Page.Orientation" : Print column-/row headings
	"portrait" Hochformat **"landscape"** Querformat
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Beispiele:

Ein neues Excel-Dokument wird erzeugt, die erste Spalte mit den Werten eines vorher berechneten Datensatzes 'channel1' gefüllt und ausgedruckt. Im Ausdruck werden die Gitternetzlinien und Spalten-/Zeilenköpfe angezeigt.

```
channel1 = Ramp(0,1,100)
XlWbNew("")
XlSetValues("A1", 0, channel1, 0)
XlSheetSetOption("GridLines.Print", "on")
XlSheetSetOption("Headings.Print", "on")
XlWbPrint()
```

Siehe auch:

XlWbPrint

XlSheetSetPicture

Anwendungsbereich: Excel-Fernsteuerung

Fügt ein Bild aus einer Datei ein.

Deklaration:

```
XlSheetSetPicture ( TxDateiName, SvOptionen, TxBereich, SvPosX, SvPosY, SvBreite, SvHöhe )
```

Parameter:

TxDateiName	Name der Bilddatei
SvOptionen	Einfügeoptionen
	0 : Der Zellbereich ist gültig, das Bild überdeckt die angegebenen Zellen und wird angepasst.
	1 : Nur die linke obere Ecke des Zellbereichs ist gültig, das Bild behält seine originale Größe.
	2 : Alle Positionsangaben sind gültig, das Bild überdeckt den angegebenen Bereich und wird angepasst.
	3 : Nur die Position ist gültig, das Bild behält seine originale Größe.
TxBereich	Zellbereich des Bildes, entweder eine Zell-Referenz im 'A1'-Stil oder der Name einer benannten Zelle bzw. Zellbereichs.
SvPosX	Position links
SvPosY	Position oben
SvBreite	Breite des Bildes
SvHöhe	Höhe des Bildes

Beispiele:

Ein neues Excel-Dokument wird erzeugt, ein Bild an verschiedene Positionen geladen und angepasst.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   ; Das Bild überdeckt die Zellen im angegebenen Zellbereich.
   XlSheetSetPicture("C:\imc\Projects\ExcelTestProjekt\Flugzeug.jpg",0,"D1:F9",0,0,0,0)
   ; Das Bild überdeckt den Pixelbereich 150x100 ab der Position 25,25.
   XlSheetSetPicture("C:\imc\Projects\ExcelTestProjekt\Flugzeug.jpg",2,"",25,25,150,100)
   ; Das Bild überdeckt den Pixelbereich entsprechend der Größe des Bildes beginnend an der linken oberen Ecke des Zellbereiches.
   XlSheetSetPicture("C:\imc\Projects\ExcelTestProjekt\Flugzeug.jpg",1,"A150",0,0,0,0)
   ; Das Bild überdeckt den Pixelbereich entsprechend der Größe des Bildes beginnend an der angegebenen Position.
   XlSheetSetPicture("C:\imc\Projects\ExcelTestProjekt\Flugzeug.jpg",3,"",100,200,0,0)
END
```

XlSheetSetPrintArea

Anwendungsbereich: Excel-Fernsteuerung

Definiert den Bereich für die zu druckenden Zellen im aktiven Tabellenblatt.

Deklaration:

```
XlSheetSetPrintArea ( TxBereich )
```

Parameter:

TxBereich	Zu druckender Bereich. Entweder eine Bereichs-Referenz im 'A1:B2'-Stil oder der Name eines benannten Bereichs.

Beispiele:

Ein neues Excel-Dokument wird erzeugt, mit einer Matrix gefüllt. Es wird der Druckbereich über die gesamte Matrix definiert und der Druckbereich ist auf 1 Seite anzupassen.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   Matrix = MatrixInit( 20, 20, "I") ; Matrix für die Tabelle erzeugen
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   ;Druckbereich definieren
   XlSheetSetPrintArea("A1:T20")
   ;Druckbereich auf 1 Seite festlegen
   XlSheetSetPrintFitTo(1,1)
END
```

XlSheetSetPrintFitTo

Anwendungsbereich: Excel-Fernsteuerung

Passt den zu druckenden Zellenbereich auf die angegebene Anzahl von neben- bzw. untereinander liegenden Seiten an.

Deklaration:

```
XlSheetSetPrintFitTo ( SvAnzahlHorizontal, SvAnzahlVertical )
```

Parameter:

SvAnzahlHorizontal	Anzahl der nebeneinander liegenden Seiten
SvAnzahlVertical	Anzahl der untereinander liegenden Seiten

Beispiele:

Ein neues Excel-Dokument wird erzeugt, mit einer Matrix gefüllt. Es wird der Druckbereich über die gesamte Matrix definiert und der Druckbereich ist auf 1 Seite anzupassen.

```
IF XlStart() ; EXCEL start
   XlVisible(1) ; EXCEL show
   XlWbNew(""); workbook create
   Matrix = MatrixInit( 20, 20, "I") ; Matrix für die Tabelle erzeugen
   LOCAL count = Leng?(Matrix) / SegLen?(Matrix)
   FOR I = 1 TO count
      ref = XlBuildA1Ref( 1, I, 1, 1)
      XlSetValues( ref, 0, Matrix[I], 0)
   END
   ;Druckbereich definieren
   XlSheetSetPrintArea("A1:T20")
   ;Druckbereich auf 1 Seite festlegen
   XlSheetSetPrintFitTo(1,1)
END
```

XlStart

Anwendungsbereich: Excel-Fernsteuerung

Startet eine Excel-Instanz.

Deklaration:

```
XlStart ( ) -> Erfolg
```

Parameter:

Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die Funktion prüft, ob bereits eine aktive Excel-Instanz existiert, die durch Funktionen dieses Kits erzeugt wurde. Falls nicht, wird eine neue Excel-Instanz gestartet, die das Ziel aller nachfolgenden Funktionsaufrufe aus diesem Kit ist.

Excel wird versteckt gestartet. Verwenden Sie ggf. die Funktion XlVisible(), um das Excel-Hauptfenster sichtbar zu machen.

Sie müssen diese Funktion nicht aufrufen, falls Sie anschließend gleich eine Datei laden (XlWbOpen()) oder eine neue Arbeitsmappe erzeugen (XlWbNew()) wollen. Beide Funktionen prüfen, ob eine entsprechende Excel-Instanz bereits existiert und starten Excel, falls nötig.

Am Ende der Excel-Verwendung sollten Sie unbedingt XlQuit() aufrufen, um die (eventuell nicht sichtbare) Excel-Instanz zu beenden und somit nicht mehr benötigte Ressourcen freizugeben.

Multithreading: Jeder Ausführungs-Thread verwendet eine eigene EXCEL-Instanz. Wenn also beispielsweise in einer parallel ausgeführten Sequenzfunktion (BEGIN_PARALLEL) mittels XlWbOpen() EXCEL gestartet und ein Dokument geladen wird, sind weitere Zugriffe auf dieses Dokument nur innerhalb der selben Sequenzfunktion erlaubt. Wenn die EXCEL-Instanz nicht mit XlQuit() explizit geschlossen wurde, erfolgt das Schließen automatisch am Ende der Sequenzfunktion.

Beispiele:

Excel wird gestartet, sichtbar gemacht und ein Makro ausgeführt, welches in der Arbeitsmappe 'MyMacros.xlsm' definiert ist. Abschließend wird Excel wieder geschlossen.

```
IF XlStart()
   XlVisible(1)
    IF NOT( XlRunMacro("'c:\XLSTemplates\MyMacros.xlsm'!Macro2"))
      BoxMessage("Fehler", GetLastError(), "!1")
   END
   XlQuit()
END
```

Siehe auch:

XlQuit, XlVisible

XlVisible

Anwendungsbereich: Excel-Fernsteuerung

Mit dieser Funktion können Sie die Sichtbarkeit der Arbeitsfenster steuern, die zu der verknüpften Excel-Instanz gehören.

Deklaration:

```
XlVisible ( Option )
```

Parameter:

Option	Sichtbarkeit
	0 : Excel ist nicht sichtbar.
	1 : Excel ist sichtbar (und bedienbar).

Beispiele:

Eine Excel-Datei wird geöffnet und das Tabellenblatt mit dem Titel 'Table2' aktiviert und angezeigt.

Nach Bestätigung durch den Anwender werden die Daten in der 2. Spalte (beginnend in Zeile 3) gelesen und nach FAMOS übertragen.

```
IF XlWbOpen("c:\results\report.xlsx")
   XlVisible(1)
   XlSheetActivate("Table2")
   IF BoxMessage( "Prüfung", "Diese Daten übertragen?", "?4") = 1
      DataColumn2 = XlGetValues("B3", 0, 0, 0)
   END
   XlQuit()
END
```

Siehe auch:

XlStart, XlQuit

XlWbActivate

Anwendungsbereich: Excel-Fernsteuerung

Die Arbeitsmappe mit dem angegebenen Namen wird aktiviert.

Deklaration:

```
XlWbActivate ( Name ) -> Erfolg
```

Parameter:

Name	Name der zu aktivierenden Arbeitsmappe.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die Funktion ist mitunter notwendig, wenn Sie mehrere Arbeitsmappen gleichzeitig geöffnet haben. Die meisten Funktionen dieses Kits wirken auf die hier festgelegte aktive Arbeitsmappe.

Der hier anzugebende Name entspricht genau dem angezeigten Namen in der Titelzeile des zugehörigen Excel-Fensters, also bei bereits gespeicherten Arbeitsmappen inklusive der Dateinamenserweiterung (z.B. '.xls' oder '.xlxs')

Beispiele:

Zu Beginn einer Auswertung wird eine neue Excel-Datei erzeugt. Der (von Excel automatisch vergebene) Name wird gemerkt. Am Ende der Auswertung soll die neu erstellte Arbeitsmappe gespeichert werden.

```
XlWbNew("")
XlVisible(1)
NewBook = XlWbGetActive()
;...
IF XlWbActivate(NewBook)
   XlWbSave( "c:\results\report.xlsx", 0)
END
```

Siehe auch:

XlWbOpen, XlWbGetActive

XlWbClose

Anwendungsbereich: Excel-Fernsteuerung

Die aktuelle Arbeitsmappe wird geschlossen.

Deklaration:

```
XlWbClose ( )
```

Parameter:

Beschreibung:

Eventuell noch nicht gesicherte Änderungen an der Arbeitsmappe gehen verloren.

Beispiele:

In einem gegebenen Verzeichnis werden alle vorhandenen Excel-Dateien nacheinander geöffnet, angezeigt und nach Bestätigung durch den Anwender ausgedruckt.

```
FileListID = FsFileListNew("c:\results", "*.xlsx", 0, 0, 1)
count = FsFileListGetCount(FileListID)
FOR I = 1 TO count
   filename = FsFileListGetName(FileListID, I)
   IF XlWbOpen( filename)
      XlVisible(1)
      IF BoxMessage( "Prüfung", "Diese Datei drucken?", "?4") = 1
         XlWbPrint()
      END
      XlWbClose()
   END
END
XlQuit()
```

Siehe auch:

XlWbOpen, XlWbNew

XlWbExist

Anwendungsbereich: Excel-Fernsteuerung

Prüft, ob eine Arbeitsmappe mit gegebenen Namen gerade offen ist.

Deklaration:

```
XlWbExist ( Name ) -> Ergebnis
```

Parameter:

Name	Name der gesuchten Arbeitsmappe.
Ergebnis	1 wenn eine solche Mappe existiert, 0 falls nicht. -1 im Fehlerfall, die Fehlerursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Der hier anzugebene Name entspricht genau dem angezeigten Namen in der Titelzeile des zugehörigen Excel-Fensters, also bei bereits gespeicherten Dokumenten inklusive der Dateinamenserweiterung (z.B. '.xls' oder '.xlxs').

Es werden nur Arbeitsmappen berücksichtigt, die durch die von FAMOS erzeugte Excel-Instanz geöffnet wurden.

Neben den per Fernsteuerung durch XlWbOpen()/XlWbNew() erzeugten Mappen zählen hier auch Dateien, die möglicherweise manuell durch den Anwender an der durch XlVisible() sichtbar gemachten Bedienoberfläche geladen wurden.

Beispiele:

Am Beginn einer längeren Auswertung wird eine Excel-Datei geladen und angezeigt, die am Ende der Auswertung nach diversen Aktualisierungen gedruckt werden soll. Da der Anwender in der Zwischenzeit die Datei bereits manuell geschlossen haben könnte, wird vor dem Druck geprüft, ob die gewünschte Datei noch geöffnet ist.

```
XlWbOpen("c:\results\report.xlsx")
XlVisible(1)
;...
ok = XlWbExist("report.xlsx")
IF ok
   XlWbActivate("report.xlsx")
   XlWbPrint()
END
XlQuit()
```

Siehe auch:

XlWbGetActive, XlWbGetName, XlWbGetCount

XlWbGetActive

Anwendungsbereich: Excel-Fernsteuerung

Der Name der gerade aktiven Arbeitsmappe wird ermittelt.

Deklaration:

```
XlWbGetActive ( ) -> Name
```

Parameter:

Name	Name der aktiven Arbeitsmappe. Leerer Text im Fehlerfall, die Fehlerursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die Funktion ist mitunter notwendig, wenn Sie mehrere Arbeitsmappen gleichzeitig geöffnet haben. Die meisten Funktionen dieses Kits wirken auf die aktive Arbeitsmappe. Die aktive Arbeitsmappe kann sich ändern, wenn mit XlWbOpen() oder XlWbNew() eine neue Mappe erzeugt wird, wenn mit XlWbActivate() explizit eine neue Mappe aktiviert wird oder der Anwender manuell eine andere Mappe aktiviert(z.B. Anklicken des entsprechenden Excel-Fensters).

Der hier ermittelte Name entspricht genau dem angezeigten Namen in der Titelzeile des zugehörigen Excel-Fensters, also bei bereits gespeicherten Dokumenten inklusive der Dateinamenserweiterung (z.B. '.xls' oder '.xlxs').

Beispiele:

Zu Beginn einer Auswertung wird eine neue Excel-Datei erzeugt. Der (von Excel automatisch vergebene) Name wird für die spätere Verwendung gemerkt.

```
XlWbNew("")
NewBookName = XlWbGetActive()
```

Am Beginn einer längeren Auswertung wird eine Excel-Datei geladen und angezeigt, die am Ende der Auswertung nach diversen Aktualisierungen gedruckt werden soll. Da der Anwender in der Zwischenzeit die Datei bereits manuell geschlossen oder eine andere Datei geladen haben könnte, wird vor dem Druck geprüft, ob tatsächlich die gewünschte Datei noch aktiv ist.

```
XlWbOpen("c:\results\report.xlsx")
XlVisible(1)
;...
tx = XlWbGetActive()
IF tx = "report.xlsx"
   XlWbPrint()
END
XlQuit()
```

Siehe auch:

XlWbOpen, XlWbNew

XlWbGetCount

Anwendungsbereich: Excel-Fernsteuerung

Die Anzahl der gerade offenen Arbeitsmappen wird ermittelt.

Deklaration:

```
XlWbGetCount ( ) -> Anzahl
```

Parameter:

Anzahl	Anzahl der offenen Arbeitsmappen. -1 im Fehlerfall, die Fehlerursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Die hier ermittelte Anzahl bezieht sich nur auf Arbeitsmappen, die durch die von FAMOS erzeugte Excel-Instanz geöffnet wurden.

Neben den per Fernsteuerung durch XlWbOpen()/XlWbNew() erzeugten Mappen zählen hier auch Dateien, die möglicherweise manuell durch den Anwender an der durch XlVisible() sichtbar gemachten Bedienoberfläche geladen wurden.

Beispiele:

Am Ende einer längeren Auswertung werden alle geöffneten Mappen aufgezählt. Wenn der Name mit "Report_" beginnt, wird die Mappe gedruckt.

```
n = XlWbGetCount()
FOR i = 1 to n
   name = XlWbGetName(i)
   IF TLike(name, "report_*", 0)
      XlWbPrint()
   END
END
XlQuit()
```

Siehe auch:

XlWbGetActive, XlWbGetName, XlWbExist

XlWbGetName

Anwendungsbereich: Excel-Fernsteuerung

Der Name einer geöffneten Arbeitsmappe wird ermittelt.

Deklaration:

```
XlWbGetName ( Index ) -> Name
```

Parameter:

Index	Index der Arbeitsmappe. Liegt zwischen 1 und der Zahl der zur Zeit offenen Mappen.
Name	Name der Arbeitsmappe. Leerer Text im Fehlerfall, die Fehlerursache kann mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Der hier ermittelte Name entspricht genau dem angezeigten Namen in der Titelzeile des zugehörigen Excel-Fensters, also bei bereits gespeicherten Dokumenten inklusive der Dateinamenserweiterung (z.B. '.xls' oder '.xlxs').

Es werden nur Arbeitsmappen berücksichtigt, die durch die von FAMOS erzeugte Excel-Instanz geöffnet wurden.

Neben den per Fernsteuerung durch XlWbOpen()/XlWbNew() erzeugten Mappen zählen hier auch Dateien, die möglicherweise manuell durch den Anwender an der durch XlVisible() sichtbar gemachten Bedienoberfläche geladen wurden.

Die Anzahl der offenen Mappen und somit der maximale Wert für [Index] kann mit der Funktion XlWbGetCount() ermittelt werden.

Beispiele:

Am Ende einer längeren Auswertung werden alle geöffneten Mappen aufgezählt. Wenn der Name mit "Report_" beginnt, wird die Mappe gedruckt.

```
n = XlWbGetCount()
FOR i = 1 to n
   name = XlWbGetName(i)
   IF TLike(name, "report_*", 0)
      XlWbPrint()
   END
END
XlQuit()
```

Siehe auch:

XlWbGetActive, XlWbGetCount, XlWbExist

XlWbNew

Anwendungsbereich: Excel-Fernsteuerung

Eine neue Arbeitsmappe wird erstellt.

Deklaration:

```
XlWbNew ( TxVorlage ) -> Erfolg
```

Parameter:

TxVorlage	Wenn Sie einen leeren Text angeben, wird eine neue Arbeitsmappe mit genau einem Tabellenblatt erzeugt. Sie können auch den kompletten Pfadnamen einer Excel-Datei angeben, die dann als Vorlage für das neue Dokument verwendet wird.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Falls noch nicht durch einen vorherigen Aufruf von XlStart()/ XlWbOpen() /XlWbNew() geschehen, wird zunächst eine versteckte Excel-Instanz gestartet, auf die alle nachfolgenden Befehle dieses Kits wirken. Verwenden Sie ggf. die Funktion XlVisible(), um das Excel-Hauptfenster sichtbar zu machen.

Am Ende der Excel-Verwendung sollten Sie unbedingt XlQuit() aufrufen, um die (eventuell nicht sichtbare) Excel-Instanz zu beenden und somit nicht mehr benötigte Ressourcen freizugeben.

Zum Schreiben von Excel-Dateien können Sie auch die Funktionen FileOpenXLS() und FileOpenXLS2() verwenden. Diese sind im Allgemeinen deutlich leistungsfähiger und schneller, dafür aber weniger flexibel als die entsprechenden Funktionen in diesem Kit.

Multithreading: Jeder Ausführungs-Thread verwendet eine eigene EXCEL-Instanz. Wenn also beispielsweise in einer parallel ausgeführten Sequenzfunktion (BEGIN_PARALLEL) mittels XlWbOpen() EXCEL gestartet und ein Dokument geladen wird, sind weitere Zugriffe auf dieses Dokument nur innerhalb der selben Sequenzfunktion erlaubt. Wenn die EXCEL-Instanz nicht mit XlQuit() explizit geschlossen wurde, erfolgt das Schließen automatisch am Ende der Sequenzfunktion.

Beispiele:

Eine neue Excel-Datei mit einem Tabellenblatt wird erzeugt und ein Datensatz in diese Tabelle übertragen. Die zweite Spalte hat die Überschrift 'Data', gefolgt von den Werten des Datensatzes. Die erste Spalte hat die Überschrift 'Time', gefolgt von den korrespondierenden Datum/Uhrzeit-Angaben. Das Ausgabeformat beider Spalten wird explizit festgelegt. Die so erstellte Datei wird dann gespeichert.

```
XlWbNew("")
; Spalte mit Datum/Uhrzeit füllen
XlSetText("A1", "Time")
time = XlCreateTimeLine(channel1, 1, 0)
XlSetValues( "A2", 0, time, 1)
Count = Leng?(time)
Range = XlBuildA1Ref(2, 1, count, 1)
XlSetCellFormat(Range, "TT.MM.JJ hh:mm:ss,0")
; Datenspalte füllen
XlSetText("B1", "Data")
XlSetValues( "B2", 0, channel1, 0)
Range = XlBuildA1Ref(2, 2, count, 1)
XlSetCellFormat(Range, "0,00")
XlWbSave( "c:\results\report", 0)
XlQuit()
```

Eine neue Excel-Datei wird erzeugt und mit den Daten einer FAMOS-Datengruppe 'MyGroup' befüllt. Das erste Blatt wird mit einigen allgemeinen Angaben gefüllt, dann wird für jeden Kanal der Gruppe ein neues Tabellenblatt angelegt und die erste Spalte mit den Werten dieses Kanals gefüllt.

Für Deck- und Datenblätter wird jeweils eine vorgefertigte Vorlage verwendet, die feste Texte und Zellenformatierungen enthält.

```
IF XlWbNew("c:\templates\firstpage.xlsx")
   XlVisible(1)
   XlSetText( "A3", "Name: Mike Smith")
   XlSetText( "A4", "Date: " + TimeToText( TimeSystem?(),0))
   FOR I = 1 TO GrChanNum?(MyGroup)
      XlSheetAdd( GrChanName?(MyGroup, I), 0, "c:\templates\datapage.xlsx")
      XlSetValues( "A1", 0, MyGroup:[I], 0)
   END
   XlWbSave( "c:\results\report", 0)
   XlQuit()
END
```

Siehe auch:

XlWbOpen, XlWbSave

XlWbOpen

Anwendungsbereich: Excel-Fernsteuerung

Das angegebene Datei im EXCEL-Format wird geöffnet.

Deklaration:

```
XlWbOpen ( Dateiname [, Kennwort] ) -> Erfolg
```

Parameter:

Dateiname	Kompletter Pfadname der zu öffnenden Datei.
Kennwort	Falls die Arbeitsmappe geschützt ist, das erforderliche Kennwort. (optional , Standardwert: "")
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beschreibung:

Falls noch nicht durch einen vorherigen Aufruf von XlStart()/ XlWbOpen() /XlWbNew() geschehen, wird zunächst eine versteckte Excel-Instanz gestartet, auf die alle nachfolgenden Befehle dieses Kits wirken. Verwenden Sie ggf. die Funktion XlVisible(), um das Excel-Hauptfenster sichtbar zu machen.

Am Ende der Excel-Verwendung sollten Sie unbedingt XlQuit() aufrufen, um die (eventuell nicht sichtbare) Excel-Instanz zu beenden und somit nicht mehr benötigte Ressourcen freizugeben.

Zum Lesen und Schreiben von Excel-Dateien können Sie auch die Funktionen FileOpenXLS() und FileOpenXLS2() verwenden. Diese sind im Allgemeinen deutlich leistungsfähiger und schneller, dafür aber weniger flexibel als die entsprechenden Funktionen in diesem Kit.

Multithreading: Jeder Ausführungs-Thread verwendet eine eigene EXCEL-Instanz. Wenn also beispielsweise in einer parallel ausgeführten Sequenzfunktion (BEGIN_PARALLEL) mittels XlWbOpen() EXCEL gestartet und ein Dokument geladen wird, sind weitere Zugriffe auf dieses Dokument nur innerhalb der selben Sequenzfunktion erlaubt. Wenn die EXCEL-Instanz nicht mit XlQuit() explizit geschlossen wurde, erfolgt das Schließen automatisch am Ende der Sequenzfunktion.

Beispiele:

Eine Excel-Datei wird geöffnet und Daten in 2 Spalten übertragen. Die aktualisierte Datei wird gedruckt und unter neuem Namen gespeichert.

```
IF NOT (XlWbOpen("c:\Templates\Template.xlsx"))
   BoxMessage("Fehler beim Öffnen", GetLastError(), "!1")
ELSE
   XlSetText("B1", "Channel1")
   XlSetValues("B2", 0, Channel1, 0)
   XlSetText("C1", "Channel2")
   XlSetValues("C2", 0, Channel2, 0)
   XlWbPrint()
   IF NOT(XlWbSave("c:\Results\Report.xlsx", 0))
      BoxMessage("Fehler beim Speichern", GetLastError(), "!1")
   END
   XlQuit()
END
```

Eine Excel-Datei wird geöffnet und das Tabellenblatt mit dem Titel 'Table2' aktiviert und angezeigt.

Nach Bestätigung durch den Anwender werden die Daten in der 2. Spalte (beginnend in Zeile 3) gelesen und nach FAMOS übertragen.

```
IF XlWbOpen("c:\results\report.xlsx")
   XlVisible(1)
   XlSheetActivate("Table2")
 IF BoxMessage( "Prüfung", "Diese Daten übertragen?", "?4") = 1
      DataColumn2 = XlGetValues("B3", 0, 0, 0)
   END
   XlQuit()
END
```

Siehe auch:

XlWbSave, XlWbNew

XlWbPrint

Anwendungsbereich: Excel-Fernsteuerung

Die aktuelle Arbeitsmappe wird gedruckt.

Deklaration:

```
XlWbPrint ( ) -> Erfolg
```

Parameter:

Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Eine Excel-Datei wird geöffnet und Daten in 2 Spalten übertragen. Die aktualisierte Datei wird gedruckt.

```
IF NOT (XlWbOpen("c:\Templates\Template.xlsx"))
   BoxMessage("Fehler beim Öffnen", GetLastError(), "!1")
ELSE
   XlSetText("B1", "Channel1")
   XlSetValues("B2", 0, Channel1, 0)
   XlSetText("C1", "Channel2")
   XlSetValues("C2", 0, Channel2, 0)
   XlWbPrint()
   XlQuit()
END
```

Siehe auch:

XlWbOpen, XlSheetPrint

XlWbSave

Anwendungsbereich: Excel-Fernsteuerung

Die aktuelle Arbeitsmappe wird gespeichert.

Deklaration:

```
XlWbSave ( TxDateiName, Reserviert ) -> Erfolg
```

Parameter:

TxDateiName	Dateiname, unter dem die Arbeitsmappe gespeichert werden soll. Wenn Sie hier einen leeren Text angeben, wird die Datei unter ihrem aktuellen Namen gespeichert.
Reserviert	Reservierter Parameter, auf 0 zu setzen.
Erfolg	Erfolg der Funktion: 1, wenn die Funktion erfolgreich ausgeführt werden konnte; 0 im Fehlerfall. Im Fehlerfall kann die Ursache mit der Funktion GetLastError() ermittelt werden.

Beispiele:

Eine Excel-Datei wird geöffnet und Daten in 2 Spalten übertragen. Die aktualisierte Datei wird gedruckt und unter neuem Namen gespeichert.

```
IF NOT (XlWbOpen("c:\Templates\Template.xlsx"))
   BoxMessage("Fehler beim Öffnen", GetLastError(), "!1")
ELSE
   XlSetText("B1", "Channel1")
   XlSetValues("B2", 0, Channel1, 0)
   XlSetText("C1", "Channel2")
   XlSetValues("C2", 0, Channel2, 0)
   XlWbPrint()
   IF NOT(XlWbSave("c:\Results\Report.xlsx", 0))
      BoxMessage("Fehler beim Speichern", GetLastError(), "!1")
   END
   XlQuit()
END
```

Eine neue Excel-Datei wird erzeugt und mit den Daten einer FAMOS-Datengruppe 'MyGroup' befüllt. Das erste Blatt wird mit einigen allgemeinen Angaben gefüllt, dann wird für jeden Kanal der Gruppe ein neues Tabellenblatt angelegt und die erste Spalte mit den Werten dieses Kanals gefüllt.

Für Deck- und Datenblätter wird jeweils eine vorgefertigte Vorlage verwendet, die feste Texte und Zellenformatierungen enthält.

```
IF XlWbNew("c:\templates\firstpage.xlsx")
   XlVisible(1)
   XlSetText( "A3", "Name: Mike Smith")
   XlSetText( "A4", "Date: " + TimeToText( TimeSystem?(),0))
   FOR I = 1 TO GrChanNum?(MyGroup)
      XlSheetAdd( GrChanName?(MyGroup, I), 0, "c:\templates\datapage.xlsx")
      XlSetValues( "A1", 0, MyGroup:[I], 0)
   END
   XlWbSave( "c:\results\report", 0)
   XlQuit()
END
```

Siehe auch:

XlWbOpen, XlWbNew

xMax

Liefert die x-Position aller relativen Maxima, die über einer bestimmten Schwelle liegen.
Alternativer Name: **xMaxi**

Deklaration:

```
xMax ( Daten, EwSchwelle ) -> XMaxima
```

Parameter:

Daten	Zu untersuchender Datensatz. Erlaubte Typen: [ND],[XY].
EwSchwelle	Schwelle
XMaxima	Die ermittelten X-Koordinaten der relativen Maxima oberhalb [EwSchwelle].

Beschreibung:

Die x-Koordinaten aller relativen Maxima, die oberhalb einer bestimmten Schwelle liegen, bilden das Ergebnis.

Falls der Parameter ein XY-Datensatz ist, muss dieser eine monotone Zeit- bzw. x-Spur aufweisen.

- Mit der Funktion Value() können Sie die zugehörigen y-Koordinaten ermitteln.
- Setzen Sie die Schwelle auf -1e100, um alle relativen Maxima zu bestimmen.
- Wird in einem Datensatz der Wert eines relativen Maximums über mehr als einen Datenpunkt gehalten, wird nur die erste x-Koordinate berücksichtigt.

Beispiele:

```
NDxmaxi = xMax(Smo5(NDdata), 10 'A')
```

Das Ergebnis sind die x-Koordinaten aller relativen Maxima des Datensatzes NDdata, die größer als 10A sind. Der Datensatz wird zuvor etwas geglättet, so dass unbedeutende relative Maxima unterdrückt werden.

```
NDymaxi = Value(NDdata, xMax(NDdata, -1e100))
```

Das Ergebnis sind die y-Koordinaten aller relativen Maxima des Datensatzes NDdata, die größer als -1e100 sind. In der Praxis werden damit alle relativen Maxima ermittelt.

```
xMini = xMax( -NDdata, 0)
xyMini = xyOf( xMini, Value( NDdata, xMini))
```

Das Ergebnis ist ein XY-Datensatz, der alle relativen Minima des Datensatzes NDdata mit negativem y-Wert enthält.

Siehe auch:

Value2, All0, Top

XOff

Der Offset eines Datensatzes in x-Richtung wird gesetzt.

Deklaration:

```
XOff ( Daten, EwXOffset ) -> Ergebnis
```

Parameter:

Daten	Datensatz, dessen x-Offset gesetzt werden soll.
EwXOffset	Neuer x-Offset.
Ergebnis	Datensatz-Kopie mit neuem Offset.

Beschreibung:

Es wird eine Kopie der Eingangsdaten erzeugt und der angegebene x-Offset eingetragen. Alle anderen Zahlen- und Kennwerte bleiben unberührt.

Der x-Offset ist die x-Koordinate des ersten Punktes des Datensatzes. Bei über der Zeit aufgenommenen Messdaten wird dieser Kennwert auch oft als Pretrigger bezeichnet.

- Die Einheit des neuen x-Offsets sollte der x-Einheit des übergebenen Datensatzes entsprechen.
- Der x-Offset sollte in seiner Größenordnung nicht um zu viele Zehnerpotenzen größer als die Abtastzeit sein. Ansonsten können Unterschiede in den x-Koordinaten der Punkte des Datensatzes nicht (ausreichend) aufgelöst werden.

Beispiele:

Die x-Koordinate des Wertes eines Histogramms, das in Form von ASCII-Daten ohne Zeitbasisinformation gelesen wurde, wird auf -128 gesetzt.

```
NDcorrect = XOff(NDhisto, -128)
```

Siehe auch:

XOff?, XDel, Leng, XOFFSET

XOff?

Der Offset eines Datensatzes in x-Richtung wird ermittelt.

Deklaration:

```
XOff? ( Daten ) -> EwXOffset
```

Parameter:

Daten	Datensatz, dessen x-Offset ermittelt werden soll.
EwXOffset	x-Offset

Beschreibung:

Der x-Offset ist die x-Koordinate des ersten Punktes des Datensatzes. Bei über der Zeit aufgenommenen Messdaten wird dieser Kennwert auch oft als Pretrigger bezeichnet.

- Das Ergebnis hat die x-Einheit des übergebenen Datensatzes.

Beispiele:

Die x-Koordinate des 2. Punktes des Datensatzes wird ermittelt:

```
xOf2ndSample = XOff?(NDdata) + XDel?(NDdata)
```

Siehe auch:

XOff, XDel?, Leng?

XOFFSET

X-Offset setzen

Deklaration:

```
XOFFSET Variablenname EwXOffset
```

Parameter:

Variablenname	Name der Variablen, deren X-Offset neu gesetzt werden soll.
EwXOffset	Abstand des ersten Datenpunktes vom Nullpunkt in X-Einheiten.

Beschreibung:

Das Kommando XOFFSET ist veraltet, statt dessen sollte in neu zu erstellenden Sequenzen die Funktion XOff() verwendet werden.

Der X-Offset des Parameters wird auf einen neuen Wert gesetzt.

Beispiele:

```
XUNIT Temp s
XOFFSET Temp 10
```

Der Variablen "Temp" wird zuerst die X-Einheit "s" zugewiesen. Der folgende Befehl bewirkt, dass der erste Datenpunkt der Variablen "Temp" bei "10 s" liegt

Siehe auch:

XOff, XOff?, XDel

XOR

Logischer "Exklusiv-ODER"-Operator

Deklaration:

```
Operand1 XOR Operand2 -> NullOderEins
```

Parameter:

Operand1	Erster zu vergleichender Einzelwert bzw. Datensatz.
Operand2	Zweiter zu vergleichender Einzelwert bzw. Datensatz.
NullOderEins	Ergebnis, 0 oder 1 (bzw. aus 0 und 1 bestehender Datensatz).

Beschreibung:

"Exklusiv-ODER"-Verknüpfung zweier Zahlen. Das Ergebnis ist 1, wenn **genau** einer der Operanden gleich 0 ist. Andernfalls ist das Ergebnis 0.

Der Operator kann auf Einzelwerte und auf Datensätze angewendet werden. Bei Datensätzen erfolgt eine punktweise Verknüpfung.

Wenn ein Parameter vom Typ XY ist, muss der andere Parameter ein Einzelwert sein.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), der jeweils andere Parameter muss dann aber entweder exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen) oder ein Einzelwert sein.

Beispiele:

Zwei digitale Datensätze werden verknüpft. Der Ergebnisdatensatz ist überall dort 1, wo beide Operandendatensätze verschieden sind.

```
Result = (DigChannel1 XOR DigChannel2)
```

Siehe auch:

AND, NOT, OR

XUNIT

X-Einheit setzen
Alternativer Name: **XEINHEIT**

Deklaration:

```
XUNIT Variablenname NeueEinheit
```

Parameter:

Variablenname	Name der Variablen, der eine neue x-Einheit zugewiesen werden soll.
NeueEinheit	Neue x-Einheit

Beschreibung:

Das Kommando XUNIT ist veraltet, statt dessen sollte in neu zu erstellenden Sequenzen die Funktion SetUnit() verwendet werden.

Die Einheit der x-Achse eines Datensatzes wird neu definiert, d. h. dem Datensatz wird eine neue x-Einheit zugewiesen.

Der Datensatz ist der erste Parameter, die Einheit der zweite.

Die Einheit wird ohne einschließende Hochkommata angegeben. Existiert kein zweiter Parameter, wird die Einheit gelöscht.

Bei komplexen Datensätzen wird die Einheit des Parameters verändert. Bei XY-Datensätzen wird die Einheit der X-Komponente gesetzt.

Beispiele:

```
ASCII
FileLoad("test.dat", "", 0)
h = Histo(test)
XUNIT h V
```

Die Einheit der x-Achse eines Histogramms wird auf "V" gesetzt. In der ASCII-Datei war diese Information nicht vorhanden.

```
XUNIT data
```

Nach diesem Befehl hat die Variable keine Einheit an der x-Achse.

Siehe auch:

SetUnit, Unit?, YUNIT, XDELTA

XY

Ein durch seine (X,Y)-Koordinaten gegebener Datensatz wird mit fester Abtastzeit nachabgetastet.

Deklaration:

```
XY ( XDaten, YDaten ) -> Ergebnis
```

Parameter:

XDaten	Die Zeit- bzw. x-Werte für den nachabzutastenden Datensatz. Typ: [ND]
YDaten	Die y-Werte für den nachabzutastenden Datensatz. Typ: [ND]
Ergebnis	Der ermittelte äquidistant abgetastete Datensatz.

Beschreibung:

Diese Funktion erzeugt aus den gegebenen X- und Y-Koordinaten eines Datensatzes durch lineare Nachabtastung mit einer konstanten Abtastzeit bzw. Delta-X einen äquidistant abgetasteten Datensatz. Das Delta-X für die Nachabtastung wird aus der X-Komponente automatisch bestimmt.

Im Allgemeinen sind die Funktion XYdt() bzw. XYdt2() besser geeignet. Sie können hier die Abtastungsbreite und die Interpolationsart wählen.

Die Funktion sollte nur auf streng monotone (Teile von) XDaten angewendet werden. Die Funktion XY setzt voraus, dass bei beiden Komponenten die Anzahl der Datenpunkte, die Abtastzeit und der x-Offset gleich sind.

- Sind die Abtastzeiten oder die x-Offsets der beiden Komponenten unterschiedlich, benutzen Sie die Funktion RSamp() oder RSampEx() zum Angleichen.
- Haben beide Komponenten eine unterschiedliche Länge, wird die geringere für beide Komponenten angenommen.
- Ist die x-Komponente nicht streng monoton, kann die Funktion XY nicht mehr richtig arbeiten. Das Ergebnis kann in imc FAMOS stets nur eine eindeutige Kurve sein.
- Die x-Komponente darf nicht konstant sein, da der erzeugte Datensatz stets eine zeitliche Ausdehnung haben muss.
- Die Auflösung des erzeugten Datensatzes richtet sich nach dem kleinsten Abstand benachbarter x-Koordinaten. Es werden maximal 10e6 Ergebnispunkte erzeugt.
- Die Funktion XY-Darstellung der Kurvenfenster arbeitet teilweise anders (zeitechte und unverfälschte Überlagerung der Komponenten).

Beispiele:

NDx und NDy sind Beispieldatensätze. NDx steigt streng monoton, und NDy ist die erste Halbwelle der Sinus-Funktion. Die XY-Überlagerung beider Komponenten liefert einen Halbkreis:

```
NDx = -cos(Ramp(0, PI / 100, 100))
NDy = sin(Ramp(0, PI / 100, 100))
NDhalfCircle = XY(NDx, NDy)
```

Siehe auch:

XYdt2, XYdt, RSampEx, XYof

XYdt

Ein durch seine (X,Y)-Koordinaten gegebener Datensatz wird mit fester Abtastzeit nachabgetastet.

Deklaration:

```
XYdt ( XDaten, YDaten, Ewdt ) -> Ergebnis
```

Parameter:

XDaten	Die Zeit- bzw. x-Werte für den nachabzutastenden Datensatz. Typ: [ND]
YDaten	Die y-Werte für den nachabzutastenden Datensatz. Typ: [ND]
Ewdt	Resultierende Abtastzeit bzw. Delta-x des zu berechnenden (äquidistant abgetasteten) Ergebnisses.
Ergebnis	Der ermittelte äquidistant abgetastete Datensatz.

Beschreibung:

Diese Funktion erzeugt aus den gegebenen X- und Y-Koordinaten eines Datensatzes durch Nachabtastung mit einer konstanten Abtastzeit bzw. Delta-X einen äquidistant abgetasteten Datensatz.

Datensätze, die mit den mathematischen Funktionen berechnet werden sollen, müssen im Allgemeinen die gleiche Abtastzeit haben und äquidistant sein. Äquidistant bedeutet: der Abstand zwischen zwei Messpunkten ist für alle Werte des Datensatzes gleich. Bei Langzeitmessungen, vernetzten (Bus-)Systemen oder bei reduzierten Datensätzen ist dies nicht immer der Fall. Häufig liegen die Messwerte in Form von Koordinatenpaaren für Zeit und Messwert vor.

Um auch solche Datensätze weiterverarbeiten zu können, müssen sie zuerst in einen äquidistanten Datensatz umgerechnet werden, was mit dieser Funktion möglich ist. Die Abtastzeit des entstehenden Datensatzes kann dabei vorgegeben werden.

Der neue Datensatz wird aus X- und Y-Koordinaten durch lineare Interpolation berechnet. Das geschieht so, als wenn man ein festes Zeitraster mit der neuen Abtastzeit über die Kurve legt und dann alle Punkte der Kurve, die sich mit dem Raster schneiden, markiert und in die neue Kurve übernimmt.

Wenn Sie den Datensatz mit konstanter Interpolation nachabtasten wollen, können Sie die Funktion XYdt2() verwenden.

- Der X-Datensatz muss monoton wachsend sein.

Durch das Nachabtasten eines Datensatzes gehen Informationen verloren. Wenn Sie mit großer Abtastzeit nachabtasten, so können Minima und Maxima der Kurve verfälscht werden.

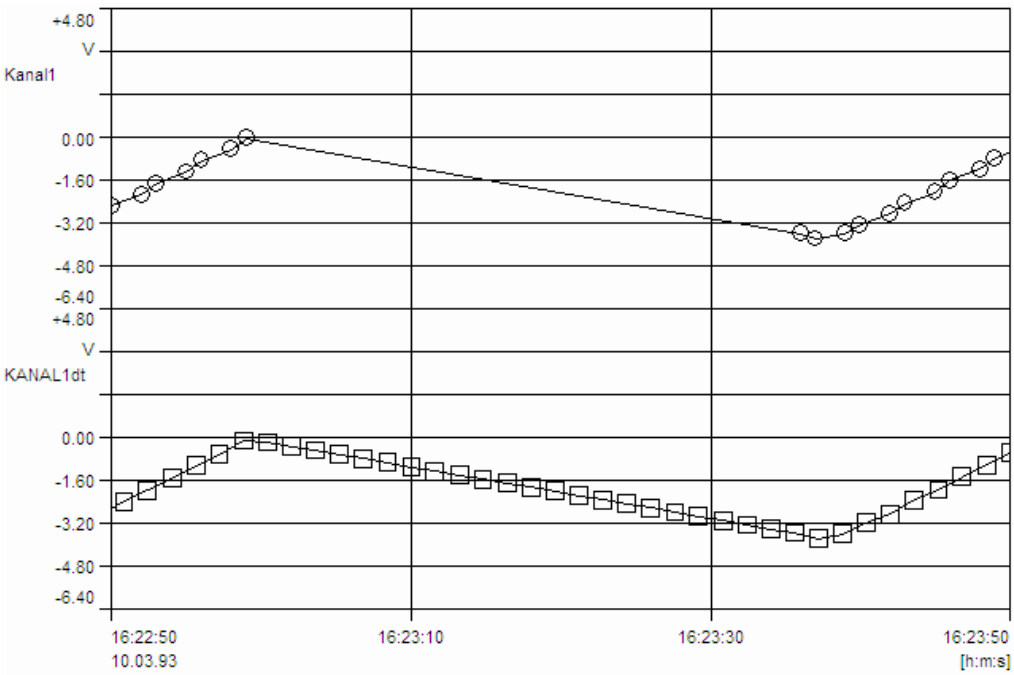
Beispiele:

Wenn der abzutastende Datensatz ein XY-Datensatz mit 2 Komponenten ist, können Sie diesen Datensatz mit Angabe der Komponentenkennungen (.X, .Y) benutzen:

```
NormalerSatz = XYdt(XYSatz.X, XYSatz.Y, 0.1)
```

Im Beispiel wird aus den Datensätzen "Kanal1" und "Zeit" der äquidistante Datensatz "Kanal1dt" mit einer Abtastzeit von 1.6 Sekunden berechnet.

```
Kanal1dt = XYdt(Zeit, Kanal1, 1.6)
```



Das Bild zeigt in der oberen Kurve die Original-Datensätze. Jeder Messwert, der aufgezeichnet wurde, ist mit einem Kringel markiert. In der

unteren Kurve ist das Ergebnis der Nachabtastung zu sehen. Auch hier ist jeder Wert mit einem Quadrat markiert.

Siehe auch:

XYdt2, RSampEx, Value2, XYof

XYdt2

Ein durch seine (X,Y)-Koordinaten gegebener Datensatz wird mit fester Abtastzeit nachabgetastet.

Deklaration:

```
XYdt2 ( XDaten, YDaten, Ewdt, EwInterpolation ) -> Ergebnis
```

Parameter:

XDaten	Die Zeit- bzw. x-Werte für den nachabzutastenden Datensatz. Typ: [ND]
YDaten	Die y-Werte für den nachabzutastenden Datensatz. Typ: [ND]
Ewdt	Resultierende Abtastzeit bzw. Delta-x des zu berechnenden (äquidistant abgetasteten) Ergebnisses.
EwInterpolation	Trifft eine X-Koordinate des Ergebnisses nicht genau eine X-Koordinate des Datensatzes, wird das Ergebnis wie folgt interpoliert:
	0 : Linear. Der Eingangsdatensatz wird linear interpoliert.
	1 : Konstant, davorliegender Wert. Der Eingangsdatensatz wird konstant interpoliert, d.h. jeder Wert wird konstant gehalten, bis der nächste Wert gültig wird. Als Ergebniswert wird also derjenige Wert des Eingangsdatensatzes verwendet, dessen x-Koordinate unmittelbar VOR der gesuchten x-Koordinate liegt.
	2 : Konstant, nächstliegender Wert. Als Ergebniswert wird derjenige Wert des Eingangsdatensatzes verwendet, dessen x-Koordinate am NÄCHSTEN zur gesuchten x-Koordinate liegt.
Ergebnis	Der ermittelte äquidistant abgetastete Datensatz.

Beschreibung:

Diese Funktion erzeugt aus den gegebenen X- und Y-Koordinaten eines Datensatzes durch Nachabtastung mit einer konstanten Abtastzeit bzw. Delta-X einen äquidistant abgetasteten Datensatz.

- Der X-Datensatz sollte monoton wachsend sein.
- Die Länge der Parameter-Datensätze für X und Y sollte gleich sein. Ansonsten werden 'überzählige' Werte im längeren Datensatz ignoriert.
- **Lineare** Interpolation wird im Allgemeinen bei kontinuierlichen Eingangssignalen verwendet. Das Verhalten der Funktion ist hier identisch zu XYdt().
- Die **konstanten** Interpolationsarten werden dann sinnvoll verwendet, wenn das Eingangssignal aus vordefinierten diskreten Werten gebildet wird. Dies ist z.B. bei digitalen Daten der Fall oder auch bei Messdaten, die Ihrer Bedeutung nach nur ganzzahlig sein können (z.B. die aktuelle Getriebestufe bei Antrieben). Der Ergebnisdatensatz besteht nur aus Werten, die auch im Eingangsdatensatz vorhanden sind, und besitzt das selbe Datenformat.
- Lineare Interpolation bei digitalen Eingangsdaten ist nicht möglich, der Interpolationsparameter wird dann ggf. automatisch auf den Wert 1 (konstante Interpolation) korrigiert.
- Bei den konstanten Interpolationsarten hat das Ergebnis das selbe Datenformat wie die übergebenen Y-Daten.
- Durch das Nachabtasten eines Datensatzes gehen Informationen verloren. Wenn Sie mit großer Abtastzeit nachabtasten, so können Minima und Maxima der Kurve verfälscht werden.

Beispiele:

Ein XY-Datensatz wird mit 0.1s nachabgetastet. Es wird ggf. linear interpoliert.

```
Signal01 = XYdt2(Signal.X, Signal.Y, 0.1, 0)
```

An einem Fahrzeug wird die aktuelle Getriebestufe protokolliert. Es wird bei jedem Schaltvorgang der eingelegte Gang (<gear>, enthält nur die Werte 1 - 5) und die Zeit <t> (seit Messungsbeginn) gespeichert. Um anschließende Berechnungen zu vereinfachen, wird aus diesen beiden Eingangsdaten ein mit 0.5 s äquidistant abgetasteter Datensatz erzeugt.

```
gear_resampled = XYdt2(t, gear, 0.5, 1)
```

Siehe auch:

XYdt, RSampEx, Value2, XYof

XYof

Aus X- und Y-Komponente wird ein XY-Datensatz gebildet.
Alternativer Name: **XYvon**

Deklaration:

```
XYof ( KomponenteX, KomponenteY ) -> XYDaten
```

Parameter:

KomponenteX	Datensatz, aus dem die X-Komponente gebildet wird. [ND]
KomponenteY	Datensatz, aus dem die Y-Komponente gebildet wird. [ND]
XYDaten	Resultierender XY-Datensatz

Beschreibung:

Fasst zwei normale reelle Datensätze zu einem zweikomponentigen XY-Datensatz zusammen.

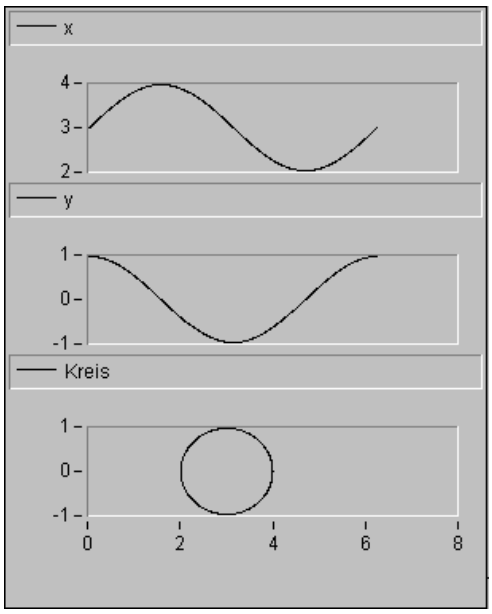
Bei unterschiedlichen Längen der Parametersätze wird die Y-Spur entsprechend gekürzt bzw. mit Nullen aufgefüllt (bei skalierbaren ganzzahligen Datenformaten wird der unskalierte Zahlenwert auf 0 gesetzt), da X- und Y-Komponente die gleiche Länge aufweisen müssen.

Beide Parameter dürfen strukturiert sein (Events/ Segmente), müssen dann aber exakt die gleiche Struktur aufweisen (gleiche Segmentlänge, Event-Anzahl und -Längen).

Beispiele:

Erzeugung eines XY-Datensatzes, dessen Wertepaare einen Kreis bilden:

```
x = sin(Ramp(0, 2*PI/ 360, 360)) + 3
y = cos(Ramp(0, 2*PI/ 360, 360))
circle = XYof(x, y)
```



Siehe auch:

CmpX, CmpY, XYdt, XYdt2

YUNIT

Y-Einheit setzen
Alternativer Name: **YEINHEIT**

Deklaration:

```
YUNIT Variablenname NeueEinheit
```

Parameter:

Variablenname	Name der Variablen, der eine neue y-Einheit zugewiesen werden soll.
NeueEinheit	Neue y-Einheit

Beschreibung:

Das Kommando YUNIT ist veraltet, statt dessen sollte in neu zu erstellenden Sequenzen die Funktion SetUnit() verwendet werden.

Die Einheit der y-Achse eines Datensatzes wird neu definiert, d.h. dem Datensatz wird eine neue y-Einheit zugewiesen.

Der Datensatz ist der erste Parameter, die Einheit der zweite.

Die Einheit wird ohne einschließende Hochkommata angegeben. Existiert kein zweiter Parameter, wird die Einheit gelöscht.

Bei komplexen Datensätzen wird nur die Einheit der ersten Komponente verändert. Bei XY-Datensätzen wird die Einheit der Y-Komponente gesetzt.

Beispiele:

```
YUNIT Daten A
```

Die Einheit der y-Achse des Datensatzes "Daten" wird auf "A" gesetzt.

```
YUNIT Daten
```

Jetzt hat die Variable "Daten" keine Einheit an der y-Achse mehr.

Siehe auch:

SetUnit, Unit?, XUNIT

ZDel?

Das Inkrement in z-Richtung (Delta-Z) wird ermittelt.

Deklaration:

```
ZDel? ( Daten ) -> EwZDelta
```

Parameter:

Daten	Datensatz, dessen z-Inkrement ermittelt werden soll.
EwZDelta	Delta-Z

Beschreibung:

Das Inkrement eines Datensatzes in z-Richtung wird ermittelt. Dieser Wert wird u.a. für die Skalierung der z-Achse bei 3D-Darstellungen von segmentierten Daten verwendet.

Siehe auch:

SetZDel, ZOff?, Leng?

ZOff?

Der Offset in z-Richtung wird ermittelt.

Deklaration:

```
ZOff? ( Daten ) -> EwZOffset
```

Parameter:

Daten	Datensatz, dessen z-Offset ermittelt werden soll.
EwZOffset	z-Offset

Beschreibung:

Der Startwert in z-Richtung wird ermittelt. Dieser Wert wird u.a. für die Skalierung der z-Achse bei 3D-Darstellungen von segmentierten Daten verwendet.

Siehe auch:

SetZOff, ZDel?, SetZDel

ZoomSpectrumChirpZ

Verfügbar ab: Professional Edition (SpectrumAnalysis-Kit)

Die Chirp-z-Transformation wird auf das Zeitsignal angewendet. Dabei wird das Effektivwert-Spektrum des Zeitsignals in einem gewünschten Frequenzbereich bestimmt. Die Länge des Zeitsignals muss keine 2er-Potenz sein. Das Spektrum kann mit beliebiger Auflösung von 0 Hz bis zur halben Abtastfrequenz bestimmt werden.

Deklaration:

```
ZoomSpectrumChirpZ ( Zeitsignal, FreqMin, FreqMax, FreqDelta, Fenstertyp ) -> Ergebnis
```

Parameter:

Zeitsignal	Zeitlicher Verlauf des Signals, von dem das Spektrum berechnet werden soll.
FreqMin	Unteres Ende des Frequenzbereichs, >=0
FreqMax	Oberes Ende des Frequenzbereichs, <= halbe Abtastfrequenz
FreqDelta	Frequenzlinienabstand, >= 0
Fenstertyp	Fensterfunktion für die benutzte FFT
	0 : Rechteck
	1 : Hamming
	2 : Hanning
	3 : Blackman
	4 : Blackman / Harris
	5 : Flat Top
Ergebnis	Das ermittelte Spektrum ist ein komplexer Datensatz mit Betrag und Phase. Der Betrag der einzelnen Frequenzlinien ist als Effektivwert angegeben.

Beschreibung:

Die Anzahl der ermittelten Frequenzlinien beträgt:

- Anzahl = 1 + (FreqMax - FreqMin) / FreqDelta

Dabei wird aufgerundet. FreqMin wird stets als Untergrenze eingehalten.

Bei folgenden Parametern wird die allgemeine DFT (diskrete Fouriertransformation) der Zeitdaten bestimmt, allerdings mit einem schnellen Algorithmus. Siehe auch DFTSpectrum():

- FreqMin = 0
- FreqMax = Abtastfrequenz / 2
- FreqDelta = FreqMax / Punkte_des_Zeitsignals / 2

Soll eine einzige Frequenzlinie berechnet werden, ist folgendes zu wählen:

- FreqDelta = 0.0
- FreqMin = FreqMax

Ansonsten gilt: FreqDelta > 0.0.

Beispiele:

Von einem Zeitsignal t soll das Spektrum in einem kleinen Bereich um 50Hz bestimmt werden:

```
Spektrum = ZoomSpectrumChirpZ ( t, 48, 52, 0.01, 0 )
```

Von einem Zeitsignal t soll die DFT bestimmt werden:

```
fmax = 0.5 / xdel?(t) ; höchste Frequenz
fdelta = fmax / leng?(t) / 2 ; Frequenzlinienabstand
Spektrum = ZoomSpectrumChirpZ ( t, 0, fmax, fdelta, 0 )
```

Siehe auch:

FFT, DFTSpectrum

PowerPoint-Kit (Überblick)

Verfügbar ab: Professional Edition

Übersicht

Diese Kit stellt Funktionen zur Steuerung von Microsoft PowerPoint zur Verfügung.

Sie können eine Präsentation erstellen.

Folien können aus einer anderen Präsentation in die aktuelle Präsentation eingefügt werden. Folien können dupliziert, verschoben und gelöscht werden.

In der Präsentation können Textfelder, Tabelleninhalte oder Bilder durch Inhalte aus FAMOS ersetzen werden.

Das PowerPoint-Kit benötigt imc FAMOS 7.3. oder eine höhere Version.

Voraussetzung für die Verwendung ist, dass auf demselben Rechner eine unterstützte PowerPoint-Version installiert ist.

Unterstützt werden zur Zeit: PowerPoint 2010, 2013 und 2016

Systemvoraussetzungen und Installation

Das Powerpoint-Kit ist ab der 'Professional Edition' von imc FAMOS enthalten.

Um das Powerpoint-Kit in FAMOS verwenden zu können, muss es zur Benutzung angemeldet sein. Bei einer normal ablaufenden Installation wird diese Anmeldung automatisch ausgeführt und die Funktionen des Kits erscheinen in der Funktionsliste von FAMOS unter "Präsentation / Powerpoint-Kit".

Sollten die Funktionen nicht verfügbar sein, prüfen Sie bitte mit dem Menü-Befehl "Extra / Optionen / Erweiterungen / Kits" die angemeldeten Kits. In der Liste sollte auch eine Zeile "PowerPoint- Kit [imcPowerPointKit.dll]" erscheinen. Ist dieser Eintrag nicht zu sehen, kontrollieren Sie bitte, ob sich die Datei "imcPowerPointKit.dll" im gleichen Verzeichnis wie die Datei "Famos.exe" befindet und wenden sich ggf. an die Hotline.

Präsentation für die Benutzung durch das imc PowerPoint Kit vorbereiten

Für die Benutzung einer PowerPoint Datei durch das imc PowerPoint Kit sind zusätzliche Vorbereitungen zu treffen. Das Kit kann Inhalte von Textfeldern, Tabellen und Bildern ersetzen. Dazu müssen diese Objekte in PowerPoint gekennzeichnet sein.

Diese Kennzeichnung erfolgt über den Alternativtext der Form. Jedem Objekt einer Folie kann ein Alternativtext zugewiesen werden.

Die Kit- Funktionen durchsuchen alle Formen einer Folie nach dem Alternativtext. Wurde eine Form gefunden, so wird der Text, der Zelleninhalt der Tabelle oder das Bild ersetzt.

In PowerPoint können Sie den Alternativtext wie folgt eingeben.

1. Selektieren Sie auf der Folie das Form-Objekt
2. Klicken Sie mit der rechten Maustaste in das Element und wählen Sie "Form formatieren" bzw. "Grafik formatieren" aus.
3. Klicken Sie auf "Größe und Eigenschaften" und dann auf "Alternativtext"
4. Geben Sie im Feld "Beschreibung" einen Text für die Kennzeichnung ein, z.B. FAMOS_Text1. Dieser Kennzeichnungstext wird in den Kit-Funktionen zum Suchen des Formobjektes benutzt.
5. **Wichtig!** Geben Sie den Kennzeichnungstext **nicht** im Feld "Titel" ein.
6. Wollen Sie den Alternativtext für seine ursprüngliche Funktion benutzen (barrierefreies PowerPoint), so geben Sie im Feld "Beschreibung" zuerst den Text für die Kennzeichnung ein, gefolgt von einem Semikolon (;). Dann kann der Text folgen, der für ein barrierefreies PowerPoint gedacht ist. Das Semikolon ist nicht Teil des Kennzeichnungstextes.

Multithreading

Jeder Ausführungs-Thread verwendet eine eigene Powerpoint-Instanz. Wenn also beispielsweise in einer parallel ausgeführten Sequenzfunktion (BEGIN_PARALLEL) mittels PptOpenPresentation() Powerpoint gestartet und ein Dokument geladen wird, sind weitere Zugriffe auf dieses Dokument nur innerhalb der selben Sequenzfunktion erlaubt. Wenn die Powerpoint-Instanz nicht explizit mit PptClosePresentation() geschlossen wurde, erfolgt das Schließen automatisch am Ende der Sequenzfunktion.

Copyright

Microsoft PowerPoint ist eingetragenes Warenzeichen der Microsoft Corporation, USA.

Für den Zugriff auf PowerPoint benutzt das Kit die Assemblies PowerPointApi.dll, OfficeApi.dll, VBIDEApi.dll und NetOffice.dll.aus dem Paket NetOffice 1.7.3. Das Paket wurde unter der MIT Licence. Copyright © 2012 Sebastian Lange veröffentlicht.

R-Kit (Überblick)

Verfügbar ab: Professional Edition

Dieses Kit stellt Funktionen zur Anbindung des R-Systems an FAMOS zur Verfügung.

Die Funktionen realisieren eine Brücke zum R-System.

R ist ein Programmiersystem für statistische Analysen. R enthält eine sehr große Bibliothek von Funktionen, die für statistische Analysen benutzt werden können. R ist frei verfügbar.

Das R-Kit enthält Funktionen zum Setzen und Lesen von R- Variablen und zum Ausführen von R- Skripten.

Voraussetzungen

Das R- Kit ist ab der **'Professional Edition'** von imc FAMOS 7.3 oder höher enthalten.

Das R-System muss auf dem PC installiert sein. Es wird mindestens die Version 3.3.2 benötigt. Es kann aus dem Internet unter http://www.r-project.org heruntergeladen und installiert werden. Bei der Installation können alle vorgeschlagenen Einstellungen beibehalten werden.

Um das R-System effektiv zu benutzen, sind Programmierkenntnisse in R notwendig.

Anmeldung in FAMOS

Um das R-Kit in FAMOS verwenden zu können, muss es zur Benutzung angemeldet sein. Bei einer normal ablaufenden Installation wird diese Anmeldung automatisch ausgeführt und die Funktionen des Kits erscheinen in der Funktionsliste von FAMOS unter "Analyse / Statistik / R-Kit".

Sollten die Funktionen nicht verfügbar sein, prüfen Sie bitte mit dem Menü-Befehl "Extra / Optionen / Erweiterungen / Kits" die angemeldeten Kits. In der Liste sollte auch eine Zeile "R- Kit [imcRKit.dll]" erscheinen. Ist dieser Eintrag nicht zu sehen, kontrollieren Sie bitte, ob sich die Datei "imcRKit.dll" im gleichen Verzeichnis wie die Datei "Famos.exe" befindet und wenden sich ggf. an die Hotline.

Allgemeines

In R muss bei den Funktionsnamen und Variablennamen die Groß- und Kleinschreibung beachtet werden.

Die fehlerhafte Ausführung einer R-Kit- Funktion führt zum Abbruch der FAMOS-Sequenz.

Multithreading

Alle Funktionen des R-Kits dürfen in beliebigen Ausführungs-Threads (BEGIN_PARALLEL) aufgerufen werden, wirken aber Thread-übergreifend und global. Alle Aufrufe werden intern zunächst in den FAMOS-Haupt-Thread verschoben und von dort ausgeführt, da die R-Laufzeitumgebung keine parallelen Aufrufe aus unterschiedlichen Threads unterstützt.

Copyright

R ist unter der GNU General Public License (GPL), Version 2 veröffentlicht.

Die Quelldateien von R werden auf https://cran.r-project.org/sources.html zur Verfügung gestellt.

Für den Zugriff auf das R-System benutzt das R-Kit die Assemblies:

R.Net und RDotNet.NativeLibrary : Copyright (c) 2010, RecycleBin

DynamicInterop : Copyright (c) 2015 Jean-Michel Perraud; Copyright (c) 2014 Daniel Collins, CSIRO;Copyright (c) 2013 Kosei, evolvedmicrobe

Die Beispiele des t-Tests wurden von Friedrich Leisch : Einführung in die induktive Statistik http://groll.userweb.mwn.de/StatistikII_SS09/VL_Folien_3.pdf übernommen.

Die Daten für das Würfelbeispiel im Chi-Quadrat- Anpassungstest wurden von Dipl.-Math. Uwe Gorbracht, veröffentlicht unter http://rechen-fuchs.de/chi-quadrat-anpassungstest-wuerfel-beispiel/, übernommen.

Die Daten für das Beispiel zur Absatzverteilung wurden von einer Internetseite der Uni St.Gallen, Uni Basel, FhbB, 2007, http://www.mri.imh.unisg.ch/Analysemethoden/Datenanalyse/Induktiv/univariat/chi2anpassungstest.html entnommen.

Für den Unabhängigkeitstest zwischen Augen- und Haarfarbe wurden Daten von Annette Bieniusa: "Programmieren in Anwendungen" Technische Universität Kaiserslautern, https://softech.informatik.uni-kl.de/homepage/teaching/PIA_SS14/7_Hypotheses.pdf verwendet.

Für den Unabhängigkeitstest zur Erwerbstätigkeit wurden die Daten von http://wikis.fu-berlin.de/pages/viewpage.action?pageId=712409813 entnommen.

Python-Kit (Überblick)

Verfügbar ab: FAMOS 2022, Professional Edition

Das Python-Kit stellt Funktionen zur Verfügung, die eine Brücke zur Programmiersprache **Python** realisieren.

Python ist eine universelle, objektorientierte Programmiersprache, die wegen ihrer Einsteigerfreundlichkeit, Plattform-Unabhängigkeit, Erweiterbarkeit und freien Verfügbarkeit (Open Source) besonders in Lehre und Wissenschaft/Technik weit verbreitet ist. Eine große Auswahl von Bibliotheken steht zur Verfügung, z.B. für numerische Berechnungen, visuelle Aufbereitung von Daten, Bildanalyse bis hin zu maschinellem Lernen. Bekannte Erweiterungsbibliotheken für den technisch-wissenschaftlichen Sektor sind beispielsweise **NumPy**, **SciPy** und **TensorFlow**.

FAMOS erzeugt eine eingebettete Instanz der Python-Laufzeitumgebung, die einen Interpreter für die Python-Programmiersprache zur Verfügung stellt.

Es stehen Funktionen zum Lesen und Schreiben von Python-Variablen sowie zur Ausführung von Python-Funktionen, -Codezeilen oder ganzen Programmen zur Verfügung.

System-Voraussetzungen:

FAMOS:

- Das Python-Kit benötigt **imc FAMOS 2022**, Professional Edition, oder eine höhere Version.

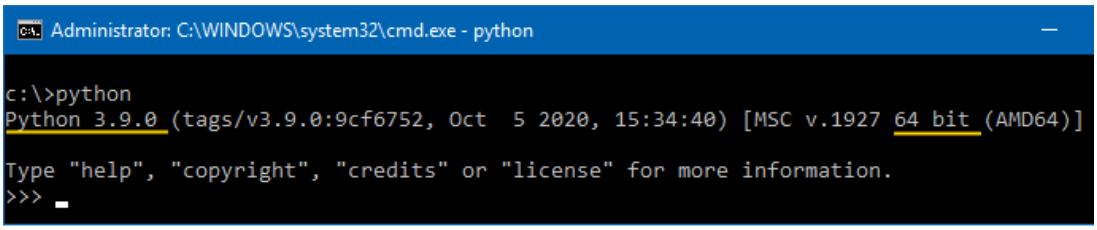
Python:

- Auf dem PC **muss eine unterstützte Python-Version installiert sein.**
- Unterstützt wird ausschließlich die Python-Referenzimplementierung der "Python Software Foundation" (CPython) in einer der nachfolgend gelisteten Versionen, die unter https://www.python.org heruntergeladen und installiert werden können.
- Kompatible **CPython-Versionen: 3.8 (64Bit), 3.9 (64Bit), 3.10 (64Bit), 3.11 (64Bit)**
- Der Pfad auf das Python-Installationsverzeichnis sollte in der **PATH**-Umgebungsvariable eingetragen sein.

NumPy (optional):

- Die FAMOS-Python-Brücke bietet (optional) besondere Unterstützung für Datentypen, die in der Erweiterungsbibliothek **"NumPy"** definiert sind (https://numpy.org).
- Kompatible **NumPy-Versionen: 1.19 (64Bit) ... 1.23 (64Bit)**

Tipp: Für eine schnelle Überprüfung, ob eine passende Python-Version installiert ist, können Sie einfach in der Windows-Eingabeaufforderung "python" eingeben. Wenn Python installiert ist, wird der Interpreter gestartet und gibt die Version aus.



Multithreading

Alle Funktionen des Python-Kits dürfen in beliebigen Ausführungs-Threads (BEGIN_PARALLEL) aufgerufen werden, wirken aber Thread-übergreifend und global. Alle Aufrufe werden intern zunächst in den FAMOS-Haupt-Thread verschoben und von dort ausgeführt, da die Python-Laufzeitumgebung keine parallelen Aufrufe aus unterschiedlichen Threads unterstützt.

ASAM-ODS-Kit (Überblick)

Verfügbar ab: Enterprise Edition

Übersicht

Das ASAM-ODS-Plugin für FAMOS besteht aus 2 Teilen, dem eigentlichen Browser-Plugin mit Bedienoberfläche für das manuelle Arbeiten mit ODS-konformen Datenablagen, sowie dem ODS-Kit. Dabei handelt es sich um eine Funktionensammlung, die Grundfunktionen für den automatisierten Zugriff auf ODS-Datenablagen für den Einsatz in FAMOS-Sequenzen enthält.

In diesem Abschnitt finden Sie die Beschreibung der Funktionen des ODS-Kits. Einführende Bemerkungen zum ASAM-ODS-Standard, zu den unterstützten Servertypen und zur Bedienung der Plugin-Oberfläche finden Sie im Kapitel "Plugin".

Systemvoraussetzungen und Installation

Um das ASAM-ODS-Kit in FAMOS verwenden zu können, muss es zur Benutzung angemeldet sein. Bei einer normal ablaufenden Installation wird diese Anmeldung automatisch ausgeführt und die Funktionen des Kits erscheinen in der Funktionsliste von FAMOS unter "Datenbanken / ASAM-ODS-Kit".

Sollten die Funktionen nicht verfügbar sein, prüfen Sie bitte mit dem Menü-Befehl "Extra / Optionen / Erweiterungen / Kits" die angemeldeten Kits. In der Liste sollte auch eine Zeile "ASAM-ODS-Kit [ImcOds02.dll]" erscheinen. Ist dieser Eintrag nicht zu sehen, kontrollieren Sie bitte, ob sich die Datei "imcOds02.dll" im gleichen Verzeichnis wie die Datei "Famos.exe" befindet und wenden sich ggf. an die Hotline.

Multithreading

Alle Funktionen des ASAM-ODS-Kits dürfen nur im Standard-Ausführungs-Thread aufgerufen werden. Ein Aufruf innerhalb eines BEGIN_PARALLEL-Blocks (also innerhalb von Sequenzfunktionen, die in einem eigenen Thread ausgeführt werden) ist nicht erlaubt.

VideoPlayer-Kit (Überblick)

Verfügbar ab: Professional Edition

Das Videoplayer-Kit ist Bestandteil des Videoplayer-Plugins zum Abspielen von Videodateien. Es kann sowohl zur Fernsteuerung des Plugins als auch zur Steuerung von Videoplayer-Elementen in Panels des FAMOS-Datenbrowsers verwendet werden. Mit den Funktionen des Kits können Abläufe automatisiert werden, es stehen alle notwendigen Funktion zum Laden von Videodateien, zur Steuerung der Wiedergabe und zum Setzen und Abfragen relevanter Parameter zur Verfügung.

Systemvoraussetzungen und Installation

Das Video-Kit ist ab der 'Professional Edition' von imc FAMOS enthalten und wird zusammen mit dem FAMOS-Videoplayer-Plugin installiert.

Um das Video-Kit in FAMOS verwenden zu können, muss es zur Benutzung angemeldet sein. Bei einer normal ablaufenden Installation wird diese Anmeldung automatisch ausgeführt und die Funktionen des Kits erscheinen in der Funktionsliste von FAMOS unter "Präsentation / Video".

Sollten die Funktionen nicht verfügbar sein, prüfen Sie bitte mit dem Menü-Befehl "Extra / Optionen / Erweiterungen / Kits" die angemeldeten Kits. In der Liste sollte auch eine Zeile "Video-Player [ImcVpl02.dll]" erscheinen. Ist dieser Eintrag nicht zu sehen, kontrollieren Sie bitte, ob sich die Datei "imcVpl02.dll" im gleichen Verzeichnis wie die Datei "Famos.exe" befindet und wenden sich ggf. an die Hotline.

Multithreading

Die Funktionen des Video-Kits dürfen in jedem Ausführungs-Thread aufgerufen werden und wirken global.

Anwendungsbeispiel

In der folgenden Sequenz wird die Beispieldatei Crash.avi im Pluginfenster geöffnet und mit ihrem Datensatz Crash.dat verbunden. Anschließend werden die Parameter der Videodatei so angepasst, dass der Lichtblitz im Bildhintergrund zeitlich mit dem Maximum der Variablen CrashTest:Licht zusammenfällt und das Brechen des Frontscheibenglases mit dem Maximum der Variablen CrashTest:Beschleunigung.

```
;Demo-Sequenz: VideoKit\Adjust.seq
;Zeigt das Anpassen der Video-Parameter
;Lade Meßdaten
LADEN "c:\Famos\VideoKit\Crash.dat"

;Öffne Kurvenfenster
KvKonfig(CrashTest:Licht,"c:\Famos4\VideoKit\Crash.ccv")

;Lade Videodatei
err=VpVideoLoad("c:\Famos\VideoKit\Crash.avi",0)
;Verbinde Kurvenfenster und Video
err=VpSetLink(CrashTest:Licht,0)

;Passe Parameter für synchrone Anzeige von Video und Kurve an
err=VpSetXOffset(0.185,0)
err=VpSetRecordRate(32,0)

;Spiele das Video langsam ab
err=VpSetPlayRate(1.5,0)
err=VpSetPosFrames(15,0)
err=VpPlay(0)

;Entferne überflüssige Variable
ENTFERNEN err
```

Eine ähnliche Aufgabe ist in Datenbrowser-Panels noch einfacher zu lösen. Ein Videoplayer-Element kann zusammen mit einem Kurvenfenster auf einer Panelseite angeordnet werden und beide Elemente bereits zur Designzeit für eine synchrone Widergabe verknüft werden. Die Konfiguration des Kurvenfenster kann evenfalls bereits komplett beim Design der Seite erfolgen. Die Aufrufe von VpSetLink() und KvKonfig() sind damit dann zur Laufzeit nicht mehr notwendig. Im folgenden Beispiel werden noch einige Abspiel-Parameter gesetzt, aber auch diese könnten bereits entsprechend beim Design der Seite am Videoplayer-Element konfiguriert werden.

```
;Lade Meßdaten
LADEN "c:\Famos\VideoKit\Crash.dat"

;Lade Panel mit Kurvenfenster und Videoplayer
DbLoadPanel("C:\VIDEO\VIDEO.PANEL", 0)

;Lade Videodatei (falls nicht schon beim Design fest eingetragen)
err=VpVideoLoad("c:\Famos\VideoKit\Crash.avi",1)

;Passe Parameter für synchrone Anzeige von Video und Kurve an
err=VpSetXOffset(0.185,1)
err=VpSetRecordRate(32,1)

;Spiele das Video langsam ab
err=VpSetPlayRate(1.5,1)
err=VpSetPosFrames(15,1)
```

```
err=VpPlay(1)

;Entferne überflüssige Variable
ENTFERNEN err
```