Übungsbuch zur Investox Formelsprache

Einführung, praktische Anleitungen und Hintergrundwissen zur Investox Formelsprache

Stand 04.11.2013

Version 1.1

Inhalt

Hinweis: Neue Kapital in Version 1.1 sind mit **NEU** gekennzeichnet.

Einleitung

Wenn Sie mit Investox eigene Handelssysteme selbst programmieren und nicht nur die vorhandenen Vorlagen und Einflussfaktoren verwenden möchten, kommen Sie nicht umhin, sich mehr oder weniger in die Formelsprache von Investox einzuarbeiten. Zahlreiche Hilfestellungen dazu finden Sie im Handbuch bzw. der Online-Hilfe von Investox, in den mitgelieferten kommentierten Beispielen (Projekte und Einflussfaktoren), in den Beispielstrategien zum Download auf www.investox.de und nicht zuletzt auch in tausenden von Beiträgen im Anwenderforum von Investox (www.investoxforum.de).

Dieses Übungsbuch schließt nun die Lücke einer zusammenfassenden Einleitung, die den Einstieg in die Programmierung erleichtern soll. Sie werden hier also teilweise in neuer Anordnung und Darstellung Inhalte finden, die Ihnen vielleicht vom Handbuch schon bekannt sind. Diese werden aber auch ergänzt und insbesondere durch zahlreiche Screenshots lesbarer gestaltet. Als Einstieg in die Formelsprache empfiehlt es sich, die Kapitel der Reihe nach durchzuarbeiten. Es ist aber natürlich ebenso möglich, das Übungsbuch als Nachschlagewerk für einzelne Themen und Aufgabenstellungen zu verwenden.

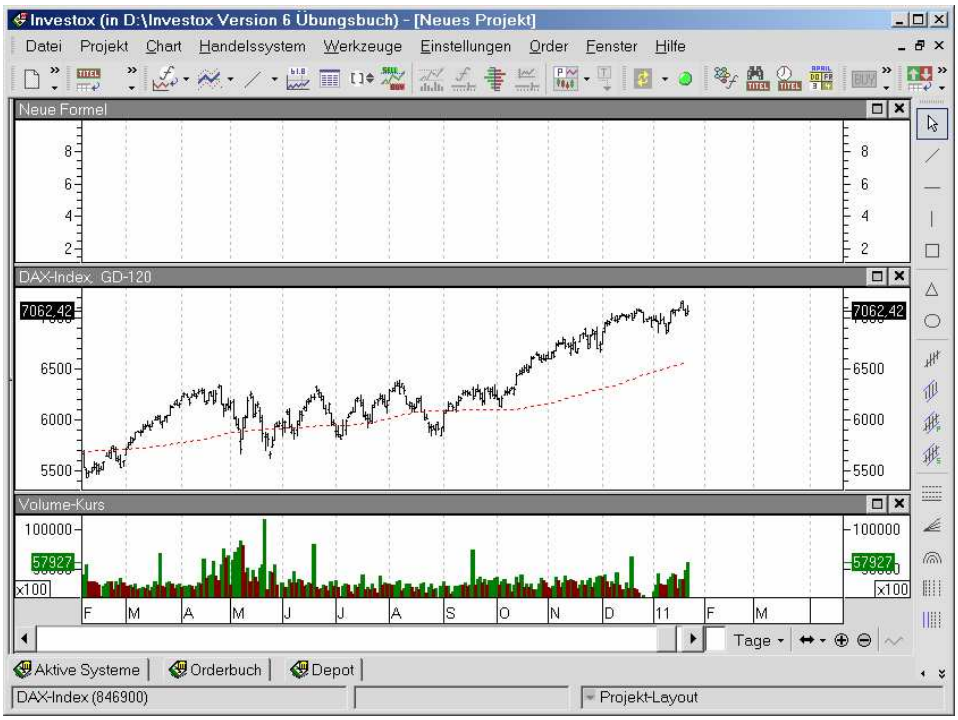
Dieses Übungsbuch führt nicht hin zur Programmierung profitabler Handelssysteme, sondern soll lediglich den Einstieg in die Umsetzung eigener Systeme erleichtern. Es beschäftigt sich auch nur in Einzelfällen näher mit der Arbeitsweise der eingebauten Indikatoren. Diese sind bereits in der Indikatoren-Referenz der Online-Hilfe dokumentiert: Sie erhalten die Erklärung eines Indikators im Formelfeld sehr rasch, indem Sie den Eingabecursor auf einen Indikatornamen setzen und F1 drücken.

Es ist geplant, das Übungsbuch im Laufe der Zeit durch weitere Inhalte zu ergänzen. Wenn Sie Vorschläge für Themen oder für Verbesserungen haben, oder wenn Ihnen Fehler auffallen, schreiben Sie uns bitte an info@investox.de.

Erste Schritte

Um die Beispiele im Programm nachvollziehen zu können, sollten Sie die grundlegenden Schritte der Bedienung von Investox bereits mit dem Handbuch studiert haben (zum Beispiel die Tutorien 1-3). Starten Sie dann Investox und legen Sie ein neues Projekt an. Wählen Sie als Titel für das Projekt am besten einen der mitgelieferten End-of-Day-Titel wie zum Beispiel den DAX-Index.

Zoomen Sie dann auf das letzte Jahr der Daten und fügen Sie mit dem Kontextmenü des Charts (rechte Maustaste auf eine freie Chartfläche klicken) mit dem Befehl „Neuer Teilchart“ einen Teilchart oben zu. In diesen Teilchart können wir später unsere Testformeln eingeben. Den vertikalen Trennbalken zwischen Titelliste und Chart verschieben wir ganz nach links, da uns hier vorerst nur der Chart interessiert. Investox mit dem Projekt im Standardlayout sollte nun ungefähr so aussehen:



Dann kann es aber auch schon losgehen.

Einführung in die Formelsprache

Zeitreihen-Formelsprache

Eine Zeitreihe (hier gleichbedeutend mit Datenreihe) ist eine zeitliche Abfolge von Daten. Ein Beispiel für eine Zeitreihe sind die Kursdaten, die im Chart etwa als Linie dargestellt werden.

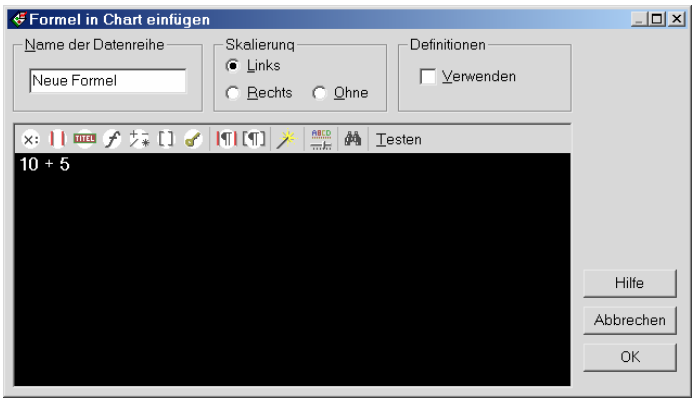
Die Investox Formelsprache ist speziell für die Aufgabenstellung einer Börsensoftware entwickelt und basiert auf einer Zeitreihen-Berechnung. Dies bedeutet, dass eine Berechnung stets auf eine gesamte Reihe von Daten angewendet wird und dass das Ergebnis wiederum eine ganze Zeitreihe ergibt – egal, ob die Berechnung nun konstante Werte, Kursdaten oder Berechnungen auf diese Daten beinhaltet.

Um dies zu verdeutlichen, fügen wir als erste Berechnung eine einfache Addition von zwei konstanten Werten in den Chart ein.

1. Klicken Sie mit der rechten Maustaste auf den oberen, leeren Teilchart und wählen Sie im Kontextmenü den Befehl **Formel einfügen**.
2. Den Namen „Neue Formel“ lassen wir einfach stehen und deaktivieren lediglich die Option **Definitionen / verwenden**. Klicken Sie dann in das Formelfeld.

Hinweis: Falls die Zwischenablage beim Einfügen der Formel einen Text enthielt, wird dieser als Vorgabe im Formelfeld eingetragen. Löschen Sie in diesem Fall zunächst diese Vorgabe, so dass das Formelfeld leer ist.

3. Geben Sie nun im Formelfeld die Berechnung „10 + 5“ ein und klicken Sie **OK**.



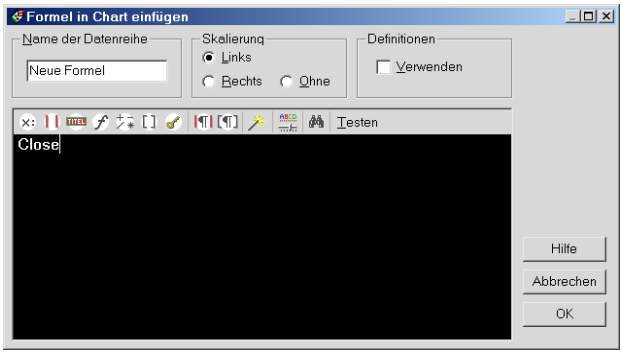
4. Den nachfolgenden Dialog „Neue Formel formatieren“ bestätigen Sie ebenfalls mit **OK**.

Im oberen Teilchart sehen Sie nun unsere erste Berechnung, nämlich eine horizontale Linie beim Wert 15 (das Ergebnis aus 10 + 5):

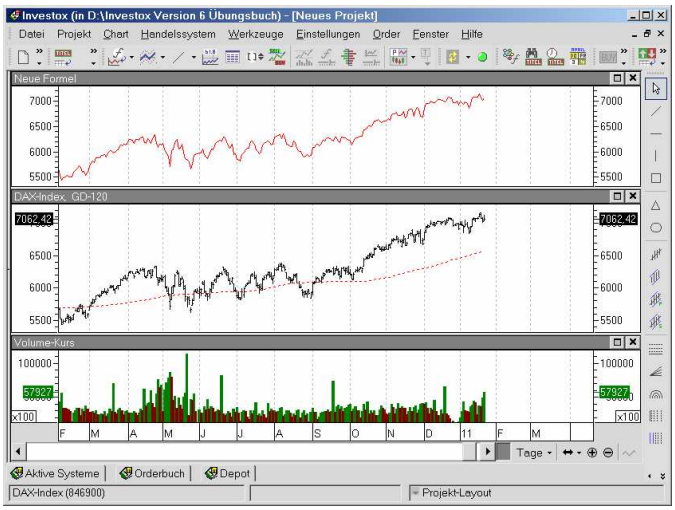
Diese Linie ist hier kein einzelner Wert, sondern eine Zeitreihe, die für jede Datenperiode den Wert 15 liefert.

Dies ist allerdings für sich betrachtet noch nicht so spannend, da wir für eine solche Darstellung im Chart ja auch eine horizontale Linie einzeichnen könnten. Interessanter wird es, wenn die Zeitreihe pro Periode unterschiedliche Werte liefert.

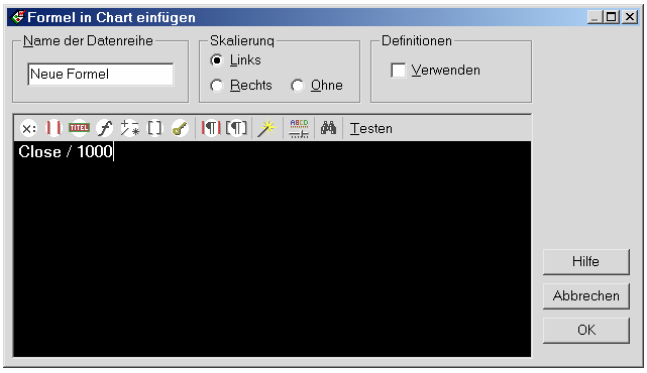
1. Klicken Sie mit rechten Maustaste auf die rote berechnete Linie, so dass sich das Kontextmenü für diese Linie öffnet. Wählen Sie im Kontextmenü den Befehl **Berechnung bearbeiten**.
2. Löschen Sie unsere Berechnung „10 +5“ aus dem Formelfeld und schreiben Sie statt dessen: „Close“. Klicken Sie **OK**.



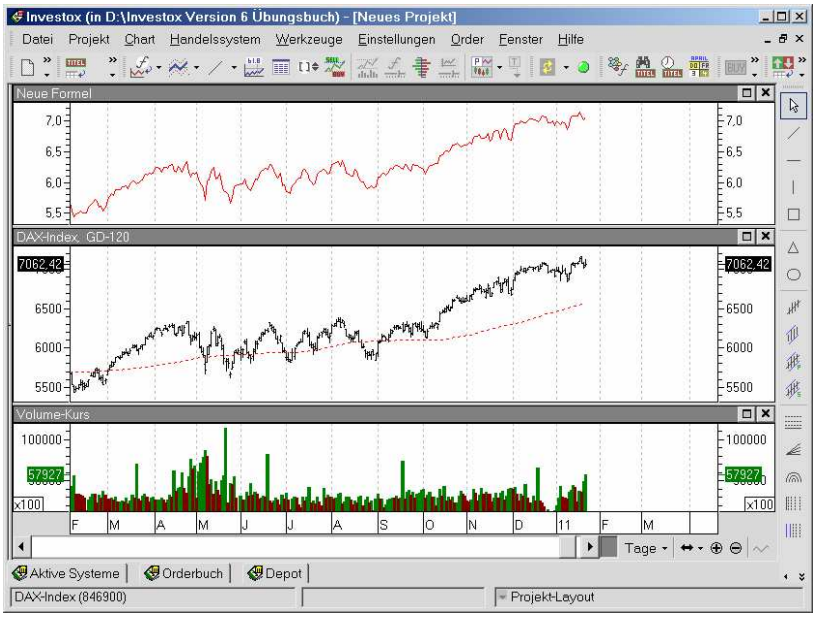
Da mit „Close“ die Schlußkurse der Datenreihe angesprochen werden, erhalten wir wie erwartet das folgende Bild:



Der obere Teilchart wiederholt also einfach den Kursverlauf des Titels. Auch dies ist natürlich noch nicht spannend. Das wird sich bald ändern. Gehen Sie wie oben beschrieben über das Kontextmenü der Formel im Chart in den Dialog „Berechnung bearbeiten“. Ergänzen Sie dort die Eingabe „Close“ mit dem Zusatz „ / 1000“, so dass sich „Close / 1000“ ergibt.



Klicken Sie **OK**. Zurück im Chart sehen wir unsere „richtige“ Berechnung, nämlich wiederum den Verlauf der Schlußkurse, nun aber schwankend im Bereich 5,5 bis 7 statt 5500 bis 7000 – da wir die Schlußkurse durch den Wert 1000 dividiert haben.



Wir sehen also, auch hier wurde die Berechnung in jeder Datenperiode durchgeführt: Jeder einzelne Schlußkurs der Zeitreihe wurde durch 1000 geteilt.

Exkurs: In einer Programmiersprache, die nicht auf Zeitreihen basiert, müsste man für die Berechnung jeder einzelnen Datenperiode eine entsprechende Anweisung programmieren, zum Beispiel in einer Schleife nach dem Muster:

```
For i = Erste Periode to LetztePeriode
   Close[i] = Close[i] / 1000
Next i
```

In der Investox Formelsprache müssen wir keine Schleifen programmieren, da sich jede Anweisung sofort auf die gesamte Datenreihe auswirkt. Dies hat auch den Vorteil, dass eine solche Berechnung einfacher zu notieren ist und vor allem wesentlich schneller ausgeführt werden kann als eine zeilenweise interpretierte Berechnung.

Für manche fortgeschrittenen Berechnungen, in denen mehr Zugriff auf die einzelnen Berechnungsschritte erforderlich ist, kann dies auch ein Nachteil sein. Daher gibt es in Investox als alternative Programmiersprache für Indikatoren auch VBScript.

Grundlegende Arten von Berechnungen

In diesem Kapitel lernen Sie die verschiedenen Bestandteile und Möglichkeiten der Formelsprache kennen. Wir verwenden für Beispiele wieder den Formeleditor für den Chart, ohne dabei auf die Werkzeuge des Formeleditors einzugehen. In der Dokumentation von Investox finden Sie eine genaue Beschreibung des Formeleditors und wie Sie dort die einzelnen Bestandteile wie Indikatoren, Operatoren etc. auch ohne Tippen bequem mit der Maus einfügen können.

Indikatoren

Investox enthält eine große Auswahl vorgefertigter Indikatoren, die direkt angewendet und analysiert werden können. Die Schreibweise von Indikatoren in Investox ist wie folgt:

```
IndikatorName(Parameterliste)
```

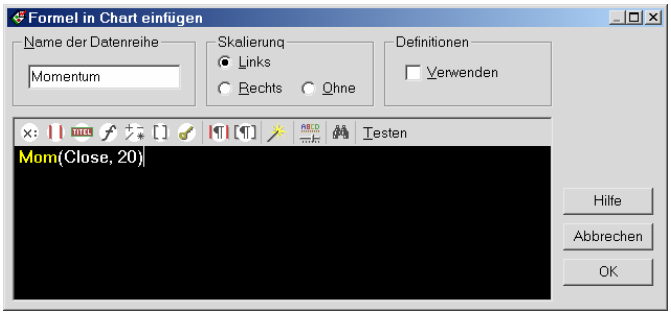
Dabei ist „IndikatorName“ (wie zu erwarten) der Name des Indikators. Dieser wird direkt, also ohne Leerstelle, gefolgt von einer öffnenden und einer schließenden Klammer. Zwischen den Klammern stehen eine oder mehrere Einstellungen (sogenannte „Parameter“), mit denen man festlegen kann, wie der Indikator genau arbeiten soll. Sind es mehrere Einstellungen, werden diese durch ein Komma getrennt.

Betrachten wir gleich ein konkretes Beispiel und fügen mit einer Formel einen Indikator in den Chart ein. Erstellen Sie zunächst wieder, wie unter „Erste Schritte“ beschrieben, ein neues Projekt mit dem Dax-Index und mit einem leeren Teilchart oben. Fügen Sie dann die folgende Formel in den oberen Teilchart ein:

```
Mom(Close, 20)
```

Der „IndikatorName“ ist hier „Mom“ und die Parameterliste enthält die beiden Einstellungen „Close“ und „20“. Berechnet wird entsprechend ein Momentum der Schlußkurse über 20 Perioden.

Beachten Sie die Klammersetzung: Die öffnende Klammer muss direkt hinter dem Indikatornamen folgen, so dass der Indikatorname im Formeleditor in gelber Farbe erscheint.



Nach **OK** und Bestätigen der Formatierung der Datenreihe wieder mit **OK** sollte der Chart mit dem Momentum im oberen Teilchart ungefähr so aussehen:

Der beliebte Momentum-Indikator verdeutlicht die Kursstärke, wobei Werte über 100 steigende und Wert unter 100 fallende Kurse anzeigen. Sein Verlauf korreliert meistens stark mit dem Kursverlauf im betrachteten Zeitraum.

Mit einer einzigen Anweisung „Mom(Close, 20)“ liefert Investox also eine ganze Zeitreihe mit Momentum-Werten – für jeden Close-Kurs einen Momentum-Wert.

Wenn Sie im Chart ganz nach links scrollen werden Sie feststellen, dass in den ersten 20 Perioden der Close-Kurse keine Momentum-Werte erscheinen. Dies liegt daran, dass die ersten 20 Perioden für die Berechnung des ersten Momentum-Wertes benötigt werden. In den Perioden davor kann mit anderen Worten noch kein Momentum über 20 Perioden berechnet werden, da hierfür noch nicht genügend Daten zur Verfügung stehen.

Einstellung des Zeitraums

Der betrachtete Zeitraum, also der Datenausschnitt, mit dem der Indikator berechnet wird, umfasst gemäß unserer Einstellung oben 20 Perioden (hier: Tage). Für die Berechnung des jeweiligen Datenpunktes des Momentums werden dabei jeweils die Close-Kurse der aktuellen Periode sowie der zurückliegenden 19 Perioden verwendet.

Leicht können wir den Berechnungszeitraum ändern. Schreiben Sie in der Formel:

```
Mom(Close, 60)
```

Der Chart müsste danach ungefähr so aussehen:

Mit 60 Tagen untersuchen wir hier schon eher den mittelfristigen Trend. Beliebt beim Momentum ist unter anderem die Analyse des Trends, indem man Trendlinien auf die Hoch- oder Tiefpunkte legt. Im folgenden Bild haben wir ein solche Trendlinie eingezeichet:

Das Momentum durchbricht am Ende diese Trendlinie. Dies kann auf ein vorläufiges Ende des vorigen Aufwärtstrends hindeuten. Doch die Analyse des Indikators ist hier nicht unser eigentliches Thema. Fahren wir fort mit der Formelsprache.

Welche Daten der Indikator verwendet

So wie wir den Indikator eingefügt haben, wird er auf die Schlußkurse („Close“) berechnet. Wir können ebenso jede andere Art von Daten statt „Close“ als Grundlage der Berechnung auswählen. Geben Sie als Beispiel an:

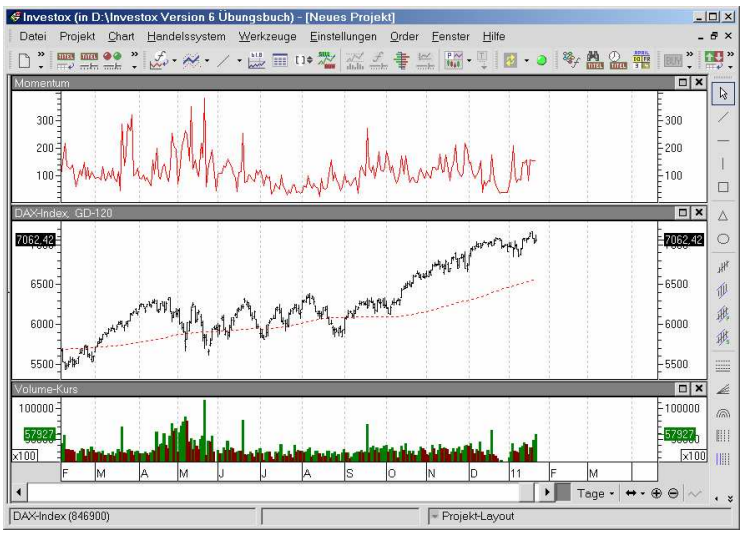
```
Mom(5, 60)
```

Im Chart sehen Sie nun lediglich eine horizontale Linie bei 100. Warum? Als Berechnungsgrundlage haben wir den festen Wert „5“ eingegeben, der niemals steigt oder fällt. Und ein Momentum von genau 100 zeigt eben an, dass keine „Kursänderung“ stattfindet.

Wir können aber auch etwas vielleicht Sinnvolleres eingeben wie etwa:

```
Mom(Volume, 60)
```

Dies sieht im Chart dann wie folgt aus:



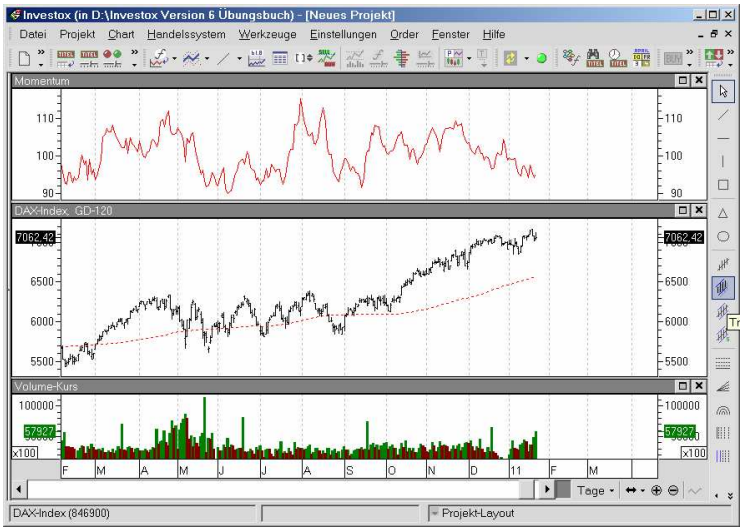
Anstatt einer Analyse des Verlaufs der Schlußkurse erhalten wir hier eine Analyse der Volumenänderung.

Ebenso können wir statt einem Kursfeld auch ganze Berechnungen als „Daten-Parameter“ verwenden. Dies wird genauer im Kapitel „Berechnungen verschachteln, Seite 20“, beschrieben. Als kleines Beispiel vorab schreiben wir:

Beachten Sie hierbei bitte die Klammersetzung. Die öffnende Klammer muss direkt hinter dem Indikatornamen folgen, so dass der Indikatorname im Formeleditor in gelber Farbe erscheint.

```
Mom( Mom(Close, 60) , 20)
```

Wir berechnen also ein 20-Tage-Momentum auf ein 60-Tage-Momentum, und analysieren auf diese Weise mit dem Momentum-Indikator die Stärke der Kursänderungen. Im Chart sieht dies so aus:



Am prinzipiellen Verlauf des Indikators ändert dies nichts, er schwankt auch hier um die 100er-Linie. Wenn Sie den Verlauf unserer Berechnung mit dem Kursverlauf vergleichen, werden Sie aber ein anderes „Phasen-Verhalten“ im Vergleich zum obigen Momentum beobachten (das Momentum kreuzt in anderen Kursphasen die 100er-Linie).

Auf einen anderen Titel berechnen

Was können wir nun noch einstellen, außer dem Berechnungszeitraum (Perioden-Einstellung) und der Berechnungsgrundlage („Daten-Parameter“)? Wir können den Indikator im selben Chart auf einen ganz anderen Titel berechnen lassen. Dazu setzen wir einfach den gewünschten Titel als zusätzlichen, ersten Parameter in den Indikator ein. Titel werden in Investox in Anführungszeichen angegeben.

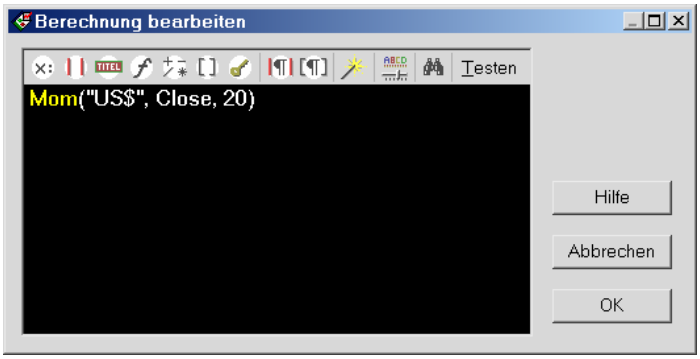
Wir möchten zum Beispiel in unserem DAX-Chart das 20-Tage-Momentum des Dollarkurses anzeigen lassen. Dazu schreiben Sie:

Wir gehen davon aus, dass der Titel „US$“ im Titelverzeichnis von Investox enthalten ist, so wie dies nach der Installation der Software der Fall ist.

```
MOM(„US$“, Close, 20)
```

Tip: Dialoggesteuert geht dies, indem Sie nur „Mom(Close, 20)“ schreiben, dann auf „Mom“ doppelklicken und den Titel „US$“ im Einstelldialog, Registerkarte „Titel“ aus dem Titelverzeichnis auswählen.

Im Formeleditor steht also:



Im Chart erhalten wir folgendes Resultat:

Da das Momentum nun auf eine ganz andere Datenreihe berechnet wird (hier US$), bekommen wir unter Umständen Momentum-Verläufe, die nicht mehr dem Kursverlauf der Basis des Charts ähneln (korrelieren). Solche weniger oder gar gegenläufig korrelierende Verläufe bilden eine wichtige Grundlage für die sogenannte „Intermarket-Analyse“, bei der der gegenseitige Einfluss verschiedener Märkte untersucht wird.

Rechenoperationen

Um eine weitere Stufe der Formelsprache zu erklimmen, müssen wir uns einer zunächst etwas trockeneren Materie zuwenden. Sobald Sie nicht nur einfach Indikatoren oder Kurse im Chart mit einer Berechnung anzeigen lassen, sondern irgend etwas damit berechnen möchten, werden Sie nicht um sogenannte Operatoren herumkommen.

Operatoren bilden das Gerüst für jede Berechnung. Sie eröffnen die Möglichkeit, die Indikatoren und Kurse auszuwerten und einzelne Berechnungsteile miteinander zu verknüpfen. Einen Operator haben wir im ersten Kapitel bereits kennengelernt: das Divisionszeichen „/“.

Es gibt aber natürlich noch mehr davon. Zu den Operatoren in der Investox Formelsprache gehören:

Die arithmetischen Operatoren (Rechenzeichen)	+, -, *, /, die Negation – und Modulus Mod
Die Vergleichsoperatoren	=, >, <, >=, <=, <>
Die logischen Operatoren	Not, And, Or, Xor
Klammern	( )

In der Regel werden Sie die Operatoren in der bekannten Weise als Verknüpfungs- oder Vergleichszeichen einsetzen. Wenn Sie es wünschen, können Sie aber auch die Indikatorschreibweise verwenden:

Operator-Schreibweise	**Indikator-Schreibweise**
Close / 1000	Div(Close, 1000)
Close + Open	Add(Close, Open)
-Close * 10	Mul(Neg(Close, 10))

Das Ergebnis von Operatoren

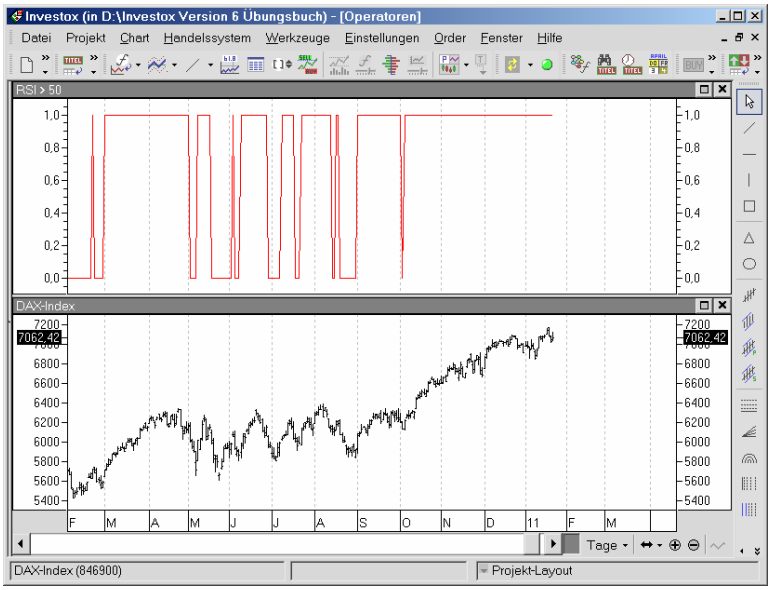
Auch wenn Sie einen Operator verwenden, ist das Ergebnis in der Zeitreihen-Berechnung von Investox immer eine Zeitreihe (wie wir im ersten Kapitel gesehen haben). Bei den **arithmetischen Operatoren** leuchtet das Ergebnis auch sofort ein. „1+1“ ergibt einfach die horizontale Linie „2“ und „Close / 1000“ ergibt die Werte der Close-Kurse geteilt durch 1000.

Bei den **Vergleichsoperatoren** und den **logischen Operatoren** ist das Ergebnis dagegen eigentlich ein Wahrheitswert, nämlich „Wahr“ oder „Falsch“. Nehmen wir an, ein Handelssystem würde die folgende Enter-Long-Regel verwenden.

```
RSI(Close, 30) > 50
```

Diese Regel wird „wirksam“, sobald die Bedingung zutrifft, dass der 30-Perioden-RSI über der 50er-Linie liegt. Dies ist gleichbedeutend damit, dass sie wirksam wird, wenn die Auswertung der Bedingung den Wert „Wahr“ ergibt.

Nun können unsere Berechnungen nicht mit den Worten „Wahr“ und „Falsch“ hantieren. Schauen wir also, wie das Ergebnis der obigen Berechnung aussieht. Fügen Sie die obige RSI-Formel in einen eigenen Teilchart in den Chart ein. Das Ergebnis sieht wie folgt aus:



Das Ergebnis der Berechnung ist also immer entweder 0 oder 1. Eine 1 steht hier für die Bereiche, in denen der 30er-RSI über der 50-Linie liegt, eine 0 für die übrigen Bereiche (in denen der RSI demnach <= 50 beträgt).

Praktischer Weise sind die Wahrheitswerte also wie folgt definiert:

```
Falsch  = 0
Wahr    = 1
```

Diese Darstellung mit Zahlenwerten ermöglicht es auch, dass wir mehrere Teilbedingungen miteinander verknüpfen können, so zum Beispiel:

```
RSI(Close, 30) > 50
  AND
MACD(Close) > 0
```

Diese Bedingung besteht aus den beiden Teilbedingungen für den RSI und den MACD. Wenn die Bedingung ausgewertet wird, werden zunächst die Wahrheitswerte der beiden Teilbedingungen berechnet. Trifft zum Beispiel sowohl die RSI- wie auch die MACD-Bedingung zu, ergibt sich

```
1
  AND
1
```

Aufgrund des logischen AND-Operators gilt, dass die Gesamtbedingung nur dann ‘Wahr’ ergibt, wenn beide Teilbedingungen ‘Wahr’ sind. Das Endergebnis ist also hier? Genau: „1“, also ‘Wahr’.

Mit dem Formel-Assistenten des Formeleditors lassen sich Teilbedingungen bequem unter verschiedenen Gesichtspunkten verknüpfen.

Hier eine Aufstellung, welches Ergebnis die einzelnen logischen Operatoren liefern:

Not	Logische Negation	Kehrt den Wahrheitswert um. Aus „Wahr“ (=1) wird „Falsch“ (=0) und umgekehrt.
And	Logisches Und	Ergibt „Wahr“, wenn beide verknüpften Teilbedingungen „Wahr“ sind, sonst „Falsch“.
Or	Logisches Oder	Ergibt „Wahr“, wenn eine der beiden verknüpften Teilbedingungen „Wahr“ ist, sonst „Falsch“.
Xor	Logisches ausschließliches Oder	Ergibt „Wahr“, wenn eine und nur eine der beiden verknüpften Teilbedingungen „Wahr“ ist. Sind beide oder keine der beiden Teilbedingungen „Wahr“, ist das Ergebnis „Falsch“.

Beispiele zum Üben (Ergebnis in rechter Spalte):

Berechnung	**Ergibt**
1 AND 0	0
1 OR 0	1
0 OR 0	0
1 AND Not 0	1

Rechnen mit Wahrheitswerten

Da das Ergebnis von Rechenoperationen auch bei logischen Operatoren und Vergleichsoperatoren eine Zahl (0 oder 1) ergibt, kann man damit, wenn man will, auch rechnen:

```
Mom(Close, 10) *  (ROC(Close, 1, $) > 0)
```

Hier wird das 10-Perioden-Momentum multipliziert mit einem Test des 1-Perioden-ROC auf >0. Wie sieht das Ergebnis dieser Berechnung aus? Es liefert in Perioden, in denen der Close-Kurs steigt, den Wert des 10-Perioden-Momentums, in den anderen Perioden den Wert 0.

Mom(Close, 10) * **1** Perioden, in denen der Schlußkurs steigt, also der ROC>0 ist

Mom(Close, 10) * **0** Perioden, in denen der Schlußkurs nicht steigt

Wahr ist alles, was nicht falsch ist!

Wie wir oben sagten, entspricht dem logischen „Falsch“ der Wert 0, dem logischen „Wahr“ der Wert 1. Nun gibt es aber in Berechnungen oft auch andere Werte als 0 und 1. Wichtig ist daher die Regel:

```
Grundsätzlich entspricht für Investox jeder Wert <> 0 dem Wahrheitswert „Wahr“
```

Vorsicht bei der Verknüpfung mehrerer Operatoren!

Wenn Sie Indikatoren mit logischen Operatoren wie ‚AND’ oder ‚OR’ verknüpfen, verwenden Sie immer auch Vergleichsoperatoren (>, <) für die Indikatoren. Also zum Beispiel:

```
RSI(Close, 10) > 50 AND RSI(Close, 20) < 50
```

Der Grund dafür ist, dass logische Operatoren ihre Teile bitweise vergleichen. Wenn Sie möchten, können Sie dies genauer nachvollziehen (wenn nicht, gehen Sie direkt zum nächsten Abschnitt):

Angenommen, Sie lassen die Vergleichsoperatoren weg und verwenden statt der obigen Formel (versehentlich) folgende Formel:

```
RSI(Close, 10) AND RSI(Close, 20)
```

Der 10er-RSI und der 20er-RSI werden hier miteinander verknüpft, ohne dass die RSI-Werte zunächst jeweils mit einem Vergleichswert ausgewertet werden. Auch hierfür berechnet Investox ein Ergebnis, indem es alle einzelnen Bits der beiden RSI-Werte jeweils mit AND verknüpft:

10er-RSI	20er-RSI	Binärer Wert 10er-RSI (wird gerundet)	Binärer Wert 20er-RSI (wird gerundet)	Ergebnis (bitweise verknüpft)
44,5	63,5	45 = 32+8+4+1 = 0101101	64 = 1000000	0101101 AND 1000000 = 0000000 **Dezimal 0**
54,3	63,4	54 = 32+16+4+2 = 0110110	63 = 32+16+8+4+2+1 = 0111111	0110110 AND 0111111 = 0110110 **Dezimal 54**

Das Ergebnis ändert sich also von 0 auf 54, was intuitiv sicherlich nicht mit Blick auf die Änderung der beiden RSI-Werte verständlich wird.

Wenn die Teilbedingungen, also die beiden RSI, zunächst mit Vergleichsoperatoren >/< ausgewertet werden, hat es der logische Operator dagegen nur mit den Wert 0 und 1 zu tun – und hier entspricht die bitweise Auswertung dem, was wir erwarten.

Rangfolge der Operatoren

Wenn mehrere Operatoren verknüpft werden, ist auch die Reihenfolge der Berechnung nicht ganz unwichtig.

Bei den Operatoren gibt es eine Rangordnung. Es werden also nicht einfach alle Operatoren einer Berechnung von links nach rechts berechnet, sondern es wird die Priorität der einzelnen Operatoren beachtet.

Die folgende Übersicht listet die Rangfolge der Operatoren auf, wobei die erste Zeile die höchste und die letzte Zeile die niedrigste Priorität anzeigt. Die Klammerzeichen werden also immer zuerst und die logischen Operatoren zuletzt ausgewertet. Operatoren derselben Prioritätsebene werden von links nach rechts ausgewertet.

Ebene	Operator	Funktion
1	( )	Klammerung
2	-	Negation
3	* /	Multiplikation Division
4	MOD	Modulus
5	+ -	Addition Subtraktion
6	= < > <= >= <>	Ist gleich Kleiner als Größer als Kleiner gleich Größer gleich Ungleich
7	Not	Logische Negation
8	And	Logisches Und
9	Or	Logisches Oder
10	Xor	Logisches ausschließliches Oder

Merksätze zur Rangfolge der Berechnung

1) Rechenzeichen und Vergleichsoperatoren werden immer vor den logischen Operatoren berechnet. Wenn also mehrere Teilberechnungen mit logischen Operatoren (AND / OR) verknüpft werden, werden die Teilberechnungen immer zuerst ausgeführt.

```
RSI(Close, 10) / 10 > 5  AND GD(Volume, 5, s) – 100 <= 50000
```

Zuerst wird jeweils berechnet was links bzw. rechts von AND steht, dann erst werden die beiden Teilbedingungen mit AND verknüpft.

2) AND wird vor OR berechnet.

```
x AND y AND z OR a AND b
```

Zuerst werden die Teilbedingungen links bzw. rechts von OR berechnet, dann erst die beiden Teilergebnisse mit OR verknüpft.

Beachte: Wenn OR und AND gemeinsam eingesetzt werden, empfiehlt sich eine Klammerung der Teilausdrücke, da sich sonst leicht Fehler einschleichen!

Klammerung

Klammern haben die höchste Priorität bei der Berechnung und werden daher dazu verwendet, die Reihenfolge der Berechnung zu bestimmen.

Nicht immer werden Klammern benötigt. Wenn Sie mehrere Teilrechnungen, die arithmetische Rechenzeichen und Vergleichsoperatoren verwenden, mit logischen Operatoren verknüpfen, ist auch ohne Klammern sichergestellt, dass zuerst alle Teilrechnungen ausgeführt werden:

```
Close > Ref(Close, -1) AND Low > Ref(Low, -1) AND High > Ref(High, -1)
```

Wird genauso berechnet wie:

```
(Close > Ref(Close, -1)) AND (Low > Ref(Low, -1)) AND (High > Ref(High, -1))
```

Man kann aber mit Klammern die „normale“ Reihenfolge der Berechnung der Operatoren verändern. Die folgenden Beispiele zeigen, wie das Ergebnis einer Formel durch Klammerung verändert werden kann:

Formel	Ergebnis	Erklärung
1 + 2 * 37	75	Das Multiplikationszeichen hat eine höhere Priorität als das Additionszeichen und wird daher zuerst berechnet.
(1 + 2) * 37	111	Durch Klammerung der Addition erzwingen Sie, dass diese zuerst berechnet wird.

Kommentare

Kommentare können zur Erläuterung einer Berechnung mitten in die Berechnung hineingeschrieben werden, müssen dazu aber als solche gekennzeichet werden. Die Kennzeichnung für Kommentare ist ein Paar von geschweiften Klammern. Kommentare werden im Formeleditor standardmäßig grün dargestellt.

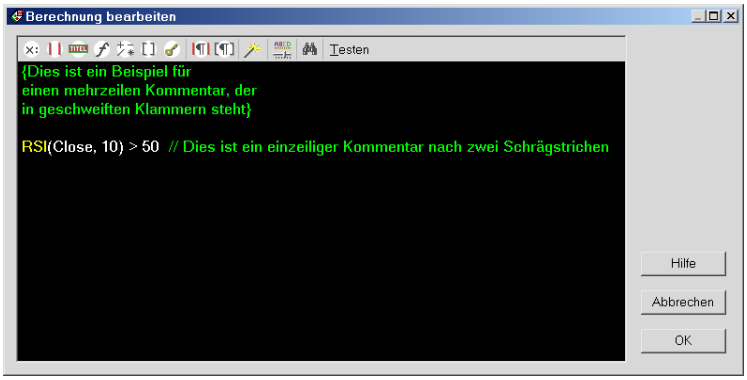
```
Berechnung.. { Kommentar... } Berechnung...
```

Steht in der Berechnung eine linke geschweifte Klammer ohne eine zugehörige rechte Klammer, so wird der ganze nach der Klammer folgende Text als Kommentar aufgefasst.

Eine weitere Möglichkeit bietet der Zeilenkommentar, bei dem nur der Rest der Zeile auskommentiert wird. Der Zeilenkommentar wird durch einen doppelten Schrägstrich „//“ gekennzeichnet.

```
// Was hier steht, ist ein Kommentar
```

Die verschiedenen Kommentar-Arten sehen Sie im folgenden Bild:



Eingeklappte Einflussfaktoren können übrigens mit Zeilenkommentaren nicht auskommentiert werden.

Effiziente Berechnung – mit Variablen rechnen

Je effizienter Sie eine Berechnung programmieren, desto schneller kann die Berechnung von Investox ausgeführt werden. Dies wirkt sich vor allem bei komplexeren Berechnungen aus.

Ein wichtiger Gesichtspunkt hierbei ist, dass Sie solche Teilberechnungen oder Indikatoren zusammenfassen und zwischenspeichern, die in der Gesamtberechnung mehrfach in gleicher Form auftreten.

Nehmen wir als Beispiel den Indikator „Phase“, dessen Berechnung aufwendig ist und daher einige Zeit dauert. Fügen Sie im Chart die folgende Formel ein:

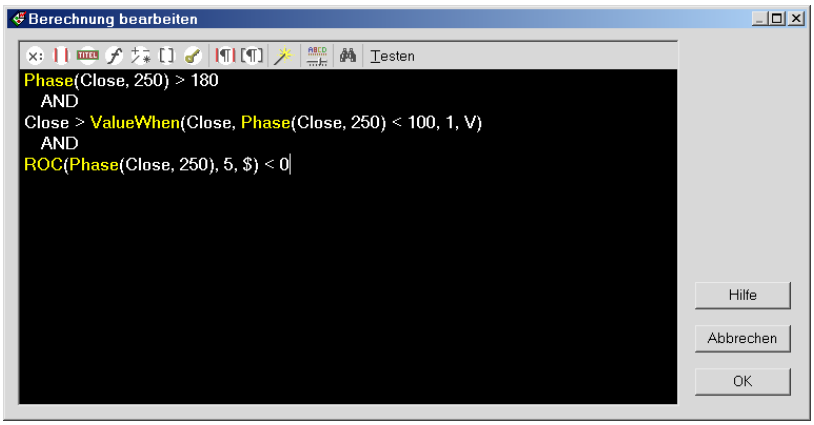
```
Phase(Close, 250) > 180
AND
Close > ValueWhen(Close, Phase(Close, 250) < 100, 1, V)
AND
ROC(Phase(Close, 250), 5, $) < 0
```

„Phase(Close, 250)“ berechnet hier, in welcher Phase der Close-Kurs in Bezug auf den 250-Perioden-Zyklus steht. Diese Phase wird in drei Teilberechnungen jeweils berechnet und verwendet, die wiederum mit AND verknüpft werden. Es wird geprüft ob

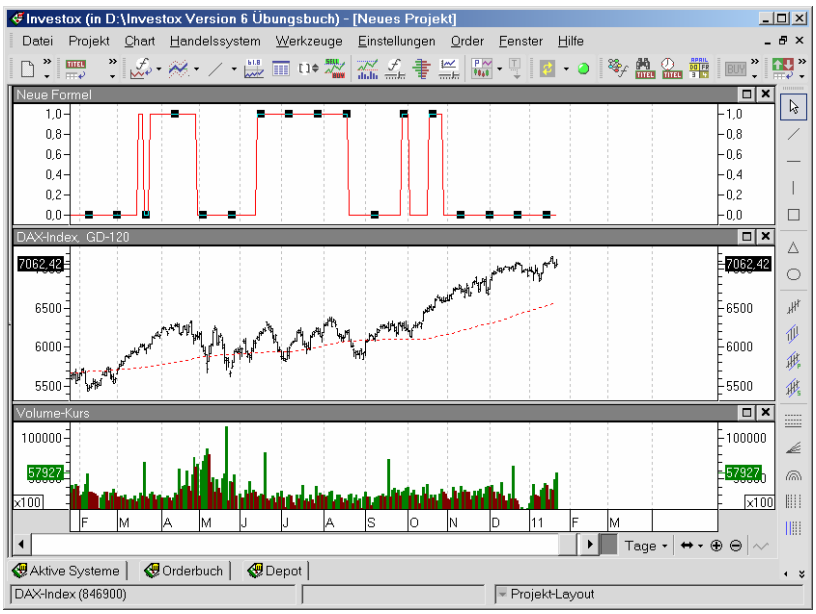
- Der Wert der Phase größer als 180 ist
- Ob der Closekurs aktuell über dem Closekurs liegt zum Zeitpunkt, als die Phase zuletzt unter dem Wert 100 lag
- Ob der Wert der Phase in den letzten 5 Perioden gesunken ist

Wenn alle drei Teilbedingungen zutreffen, wird eine 1 (=“Wahr“), an allen anderen Stellen eine 0 (=“Falsch“) ausgegeben.

Im Formeleditor sieht dies so aus:



Wenn Sie **OK** klicken, dauert es – je nach Rechnergeschwindigkeit – einen kleinen Moment, bis die Formel im Chart dargestellt wird.



Wir können die Berechnungszeit nun wesentlich verkürzen, indem wir dafür sorgen, dass der „Phase“-Indikator nicht dreimal, sondern nur einmal berechnet werden muss. Dies erreichen wir, indem wir die Berechnung des Indikators in einer Variablen zwischenspeichern und dann diese Variable an den drei betreffenden Stellen einsetzen. Dazu dient nun das Schlüsselwort „Calc“:

```
calc Phase250: Phase(Close, 250);
```

Mit dieser Zeile wird die Berechnung des Phasen-Indikators mit Hilfe des „Calc“-Schlüsselwortes einer Variablen zugewiesen, die wir hier willkürlich „Phase250“ nennen. Die Calc-Definition besteht also aus

dem Schlüsselwort „Calc“,

gefolgt von einem Leerzeichen und dem selbst festgelegten Variablennamen,

gefolgt von einem Doppelpunkt,

gefolgt von der Berechnung, deren Ergebnis (eine Zeitreihe) der Variablen zugewiesen wird,

gefolgt zum Abschluss von einem Strichpunkt.

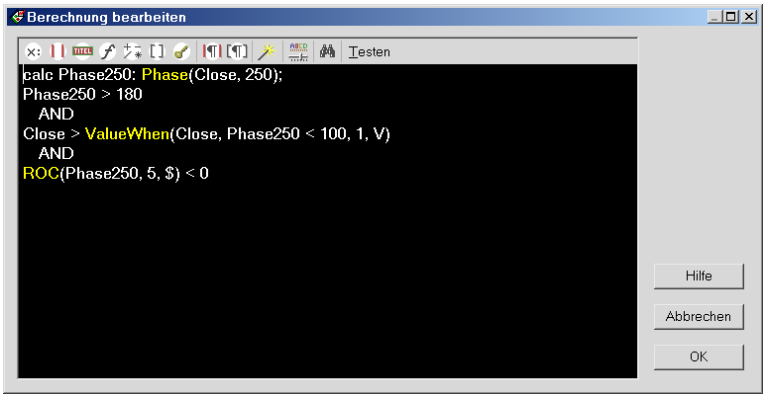
Die Variablendefinition hat also formal folgenden Aufbau, wobei „Variable“ und „Berechnung“ von Ihnen festgelegt werden:

```
Calc Variable: Berechnung ;
```

In unserer Berechnung können wir nun an allen drei Stellen statt „Phase(Close, 250)“ einfach die Variable „Phase250“ einsetzen. Gehen Sie also wieder in den Formeleditor und ändern Sie die Formel wie folgt:

```
calc Phase250: Phase(Close, 250);
Phase250 > 180
AND
Close > ValueWhen(Close, Phase250 < 100, 1, V)
AND
ROC(Phase250, 5, $) < 0
```

Das sieht im Formeleditor so aus:



Klicken Sie **OK**. Die Berechnung sollte nun spürbar schneller ausgeführt werden, das Ergebnis im Chart aber genauso aussehen wie zuvor.

Ebenso können wir konstante Werte, wie zum Beispiel Perioden-Einstellungen von Indikatoren, einer Variablen zuweisen und diese Variable dann einsetzen. Dazu dient das Schlüsselwort „Const“:

```
Const Perioden: 250;                    // den konstanten Wert definieren
Calc Phase250: Phase(Close, Perioden);  // und dann einsetzen
```

Die Definition einer Variablen für eine Konstante entspricht also der einer Berechnung, nur dass statt „calc“ das Schlüsselwort „const“ verwendet wird:

```
Const Variable: KonstanterWert ;
```

Wie wir im Kapitel „Zeitreihen-Formelsprache“ (Seite 5) gelernt haben, stellt Investox auch konstante Werte als Ergebnis einer Berechnung als Datenreihe dar. Im Chart eingefügt zeigen also die beiden folgenden Formeln dasselbe Ergebnis:

```
Const Perioden: 250;
Perioden
```

```
Calc Perioden: 250;
Perioden
```

Wozu wird dann überhaupt zwischen „Const“ und „Calc“ unterschieden? Der Grund dafür ist, dass mit „Const“ definierte Variablen als Einstellung für Werte-Parameter von Indikatoren verwendet werden können, Calc-Variablen dagegen nicht:

Korrekt:

```
Const Perioden: 250;
Phase(Close, Perioden)
```

Falsch:

```
Calc Perioden: 250;
Phase(Close, Perioden)
```

Vorteile Const / Calc:

- Der Formelcode ist besser lesbar, da die Einstellwerte einen Namen haben und gleiche Werte erkennbar sind.
- Wird eine Einstellung mehrfach in einer Berechnung verwendet, muss man bei einer Veränderung den Wert nur einmal ändern.
- Bei Calc kommt die mögliche schnellere Berechnung hinzu, falls eine Teilberechnung nur einmal statt mehrmals berechnet werden muss (wie oben gezeigt).

Globale Variablen

Die Gültigkeit von mit „Calc“ bzw. „Const“ definierten Variablen ist auf die jeweilige Formel begrenzt, in der sie definiert werden. Es besteht aber auch die Möglichkeit, Variablen als „global“ gültig zu definieren. Eine Variable wird dadurch „global“, dass Sie vor die normale Definition das Schlüsselwort „Global“ schreiben:

```
Global Const Name: Berechnung;
Global Calc Name: Berechnung;
```

Allerdings können globale Variablen ausschließlich in den Handelssystem-Regeln unter „Definitionen“ definiert werden, an keinem anderen Ort sonst. Dort definiert, können die Variablen dann in den verschiedenen Bereichen eines Handelssystems eingesetzt werden:

- In allen Handelsregeln
- In den Einstellungen der Testbedingung (zum Beispiel Enter-/Exit-Basis)
- In den Stop-Einstellungen

Der Vorteil der globalen Variablen ist, dass sie nur einmal pro Aktualisierung berechnet werden müssen und dann in all diesen Bereichen zur Verfügung stehen (Effizienz). Zudem müssen globale Variablen, wenn ihre Berechnung oder Einstellung geändert werden soll, nur an einer Stelle, nämlich in den Definitionen des Handelssystems bearbeitet werden (Übersichtlichkeit).

Hinweis: Wenn Sie eine Variable als global definieren, empfiehlt es sich, auch alle im Formeltext nachfolgenden Variablen als global zu definieren, um sicherzustellen, dass alle abhängigen Daten in allen Bereichen zur Verfügung stehen.

Berechnungen verschachteln

Bis hierher haben wir nur einzelne Indikatoren oder einfache Verknüpfungen mit Operatoren kennen gelernt.

Ein wichtiges Merkmal der Formelsprache von Investox ist es aber, dass die einzelnen Elemente der Berechnung auch ineinander eingesetzt, das heißt „verschachtelt“ werden können. Dies bildet die Voraussetzung dafür, dass auch individuelle und komplexe Auswertungen und Analysen möglich sind. Daher verarbeitet Investox im Datenreihen-Parameter eines Indikators jede beliebige gültige Unterberechnung.

Betrachten wir zunächst den Indikator „ROC“ (Rate of Change), der eine Wertänderung berechnet.

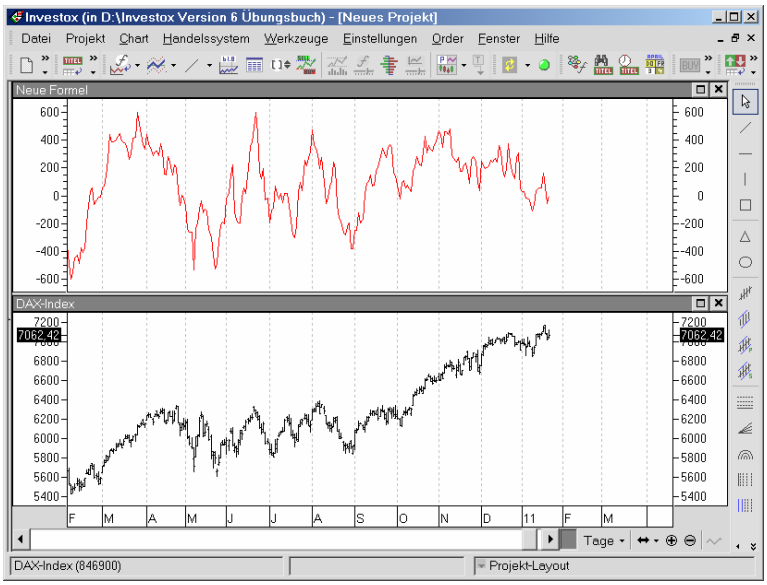
```
ROC(Daten, Perioden, $)
```

Der Parameter „Daten“ legt beim ROC-Indikator fest, für welche Daten eine Wertänderung berechnet werden soll. Der Parameter „Perioden“ legt fest, wieviele Perioden zwischen den beiden Werten liegen, deren Änderung berechnet werden soll. Mit dem letzten Parameter kann schließlich eingestellt werden, ob die absolute ($) oder prozentuale Wertänderung gefragt ist.

Betrachten wir nun den „Daten“-Parameter. Im einfachsten Fall können Sie für „Daten“ ein Preisfeld angeben, zum Beispiel die Schlusskurse. Dies sehen wir uns an. Öffnen Sie einen EoD-Chart mit dem DAX-Index und fügen Sie die folgende Formel in einen leeren Teilchart oben ein:

```
ROC(Close, 20, $)
```

Dies berechnet die absolute Wertänderung der Schlusskurse über 20 Perioden und sieht ungefähr so aus:



Der Indikator schwankt um die 0-Linie. Ein Wert > 0 zeigt an, dass der Schlusskurs gegenüber dem Kurs vor 20 Perioden gestiegen ist, ein Wert < 0 umgekehrt, dass der Schlußkurs in diesem Zeitraum gefallen ist (der Indikator zeigt sogar genau, um wie viel).

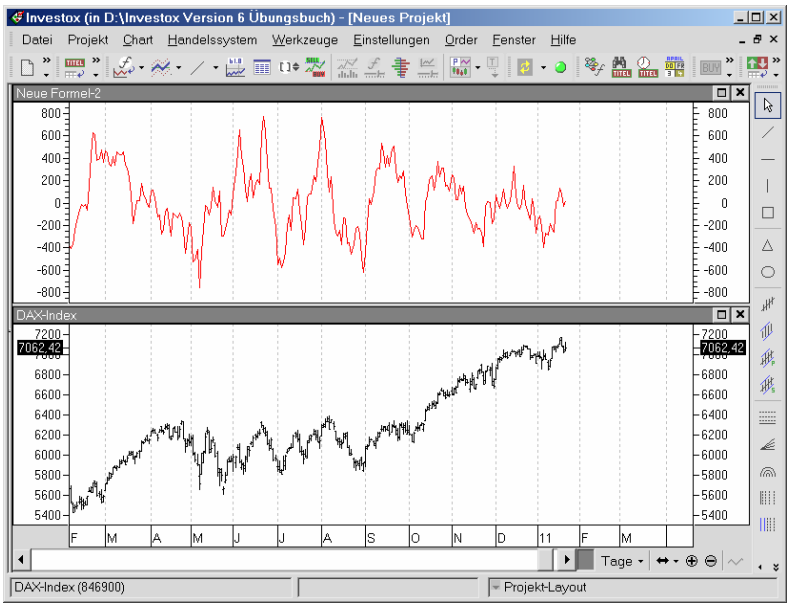
Nun kann es sein, dass wir nicht nur wissen möchten, ob die Kurse in den letzten 20 Perioden gefallen oder gestiegen sind, sondern auch, ob die Stärke der Kursänderung größer oder kleiner wurde, und zwar auf Sicht von 10 Perioden. Wir möchten mit anderen Worten sehen, ob die Kursänderung in den letzten 10 Perioden gefallen oder gestiegen ist. Dazu verwenden wir wieder den ROC-Indikator:

```
ROC( Daten            , 10, $)
```

diesmal aber nicht auf Schlusskurse, sondern auf die ROC-20 von oben berechnet. Wir setzen in den Daten-Parameter also die ROC-20-Berechnung ein:

```
ROC( ROC(Close, 20, $), 10, $)
```

Wenn Sie diese Formel in den Chart einfügen, sieht dies so aus:



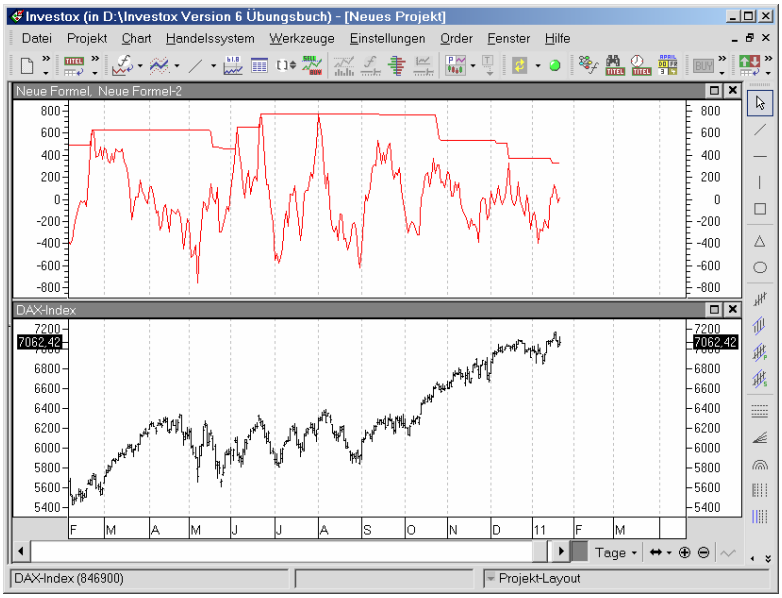
Werte des Indikators über 0 zeigen nun nicht an, dass der Kurs gestiegen ist, sondern dass die Stärke der 20-Perioden-Kursänderung in den letzten 10 Perioden zugenommen hat.

Hinweis: Beachten Sie, dass die Verschachtelung nicht von „links nach rechts“, sondern von „innen nach außen“ verläuft: Der Indikator, der zuerst berechnet werden muss, steht „innerhalb“ des Indikators, der danach berechnet wird. Im Beispiel oben wird zuerst die ROC-20 auf Close, danach die ROC-10 auf die ROC-20 berechnet.

Die Verschachtelung lässt sich natürlich viel weiter treiben. So könnte Sie zum Beispiel jetzt interessieren, welches jeweils in den letzten 60 Perioden die größte Zunahme der Kursänderungsstärke gewesen ist. Zu diesem Zweck setzen wir die vorige Berechnung in den Höchstwerte-Indikator „HHV“ ein:

```
HHV(Daten                    , 60)
HHV(ROC(ROC(Close, 20, $), 10, $), 60)
```

Diese Formel fügen wir nun zusätzlich in den oberen Teilchart ein, damit wir gut erkennen, wie der HHV-Indikator den Höchstwert der Berechnung in den jeweils letzten 60 Perioden festhält:

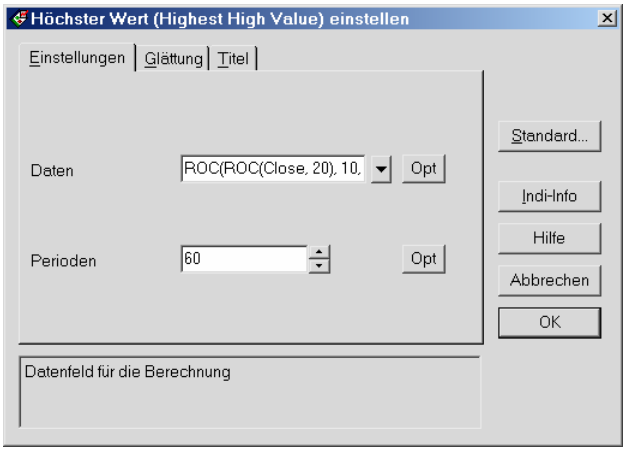


Entschlüsseln der Verschachtelung mit der Indikator-Einstellbox

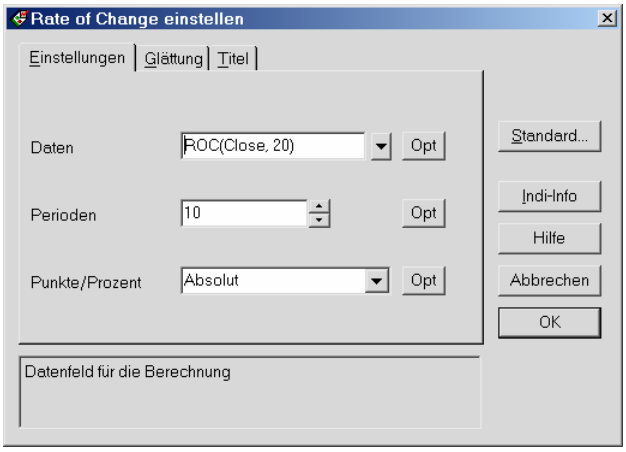
Ein hilfreiches Instrument zum nachträglichen „Entschlüsseln“ einer an sich unübersichtlichen Verschachtelung ist die Indikator-Einstellbox. Dies sehen wir uns anhand der zuletzt eingefügten Formel an:

```
HHV(ROC(ROC(Close, 20, $), 10, $), 60)
```

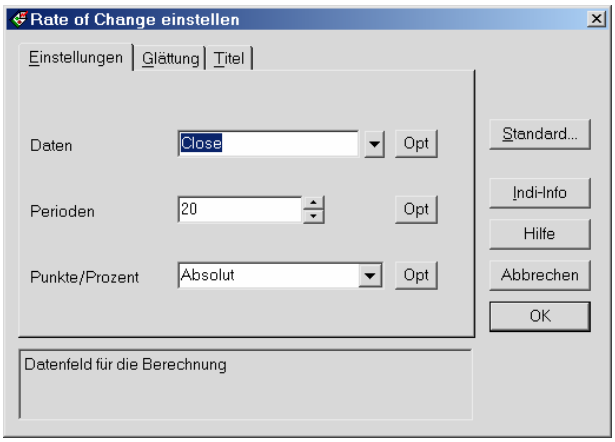
Doppelklicken Sie im Formeleditor auf den gelb eingefärbten Indikatornamen „HHV“. Es öffnet sich die Einstellbox für den Indikator HHV:



Damit nicht genug. Wählen Sie nun im Dropdown-Menü des **Daten**-Einstellfeldes den Befehl **Indikator**, so erscheint eine weitere Einstellbox, diesmal für die ROC-10 (daher ist die Periodeneinstellung=10), und im Einstellfeld **Daten** finden Sie unsere ROC-20 wieder:



Wählen Sie nun wiederum im Dropdown-Menü des **Daten**-Einstellfeldes den Befehl **Indikator**, um auch noch die ROC-20 „entschlüsseln“ zu lassen. Im Einstelldialog finden Sie dann unsere ROC der untersten Ebene, berechnet auf Close:



So können Sie bequem durch alle Ebenen der Berechnung gehen und die einzelnen Einstellungen der Parameter gegebenenfalls mit den Eingabefeldern verändern. Durch Klicken auf **OK** oder **Abbrechen** kehren Sie jeweils zur übergeordneten Ebene der Berechnung zurück. Auf diese Weise wird die Struktur einer Verschachtelung oftmals klarer und falsch gesetzte Klammern oder irrtümliche Einstellungen können relativ schnell gefunden werden.

Verschachtelte Berechnungen mit der Einstellbox zufügen

Man kann mit der Indikator-Einstellbox natürlich nicht nur vorhandene verschachtelte Berechnungen analysieren, sondern umgekehrt solche Berechnungen auch erzeugen. Hierbei ist nur zu beachten, dass Sie **mit dem äußersten, also zuletzt berechneten Indikator beginnen**.

Angenommen, Sie betrachten die tägliche Kursänderung der über 10 Tage geglätteten Schlußkurse und möchten davon den größten Wert der letzten 20 Tage erhalten. Die Formel dafür sieht (mit EoD-Daten) so aus:

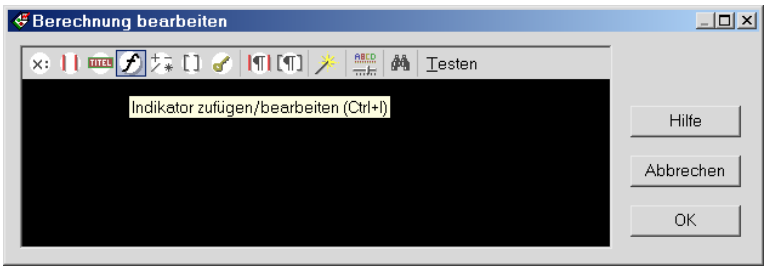
```
HHV(ROC(GD(Close, 10, S), 1, $), 20)
```

Es werden die geglätteten Schlußkurse verwendet, davon die ROC berechnet und davon wiederum der Höchstkurs. Die Abfolge der Berechnung wird in der folgenden Darstellung deutlich (von oben nach unten):

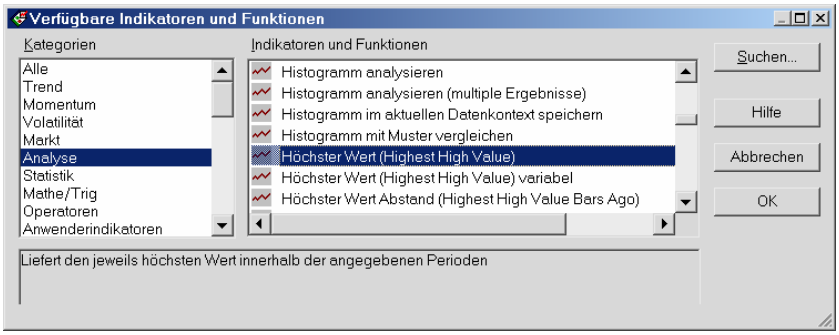
```
        GD(Close, 10, S)
   ROC(                , 1, $)
HHV(                           , 20)
```

Zuerst berechnet wird der innerste, hier also der oberste Indikator (GD). Um die Berechnung mit der Einstellbox zuzufügen, müssen Sie aber mit dem äußersten, dem zuletzt berechneten Indikator, also dem HHV beginnen.

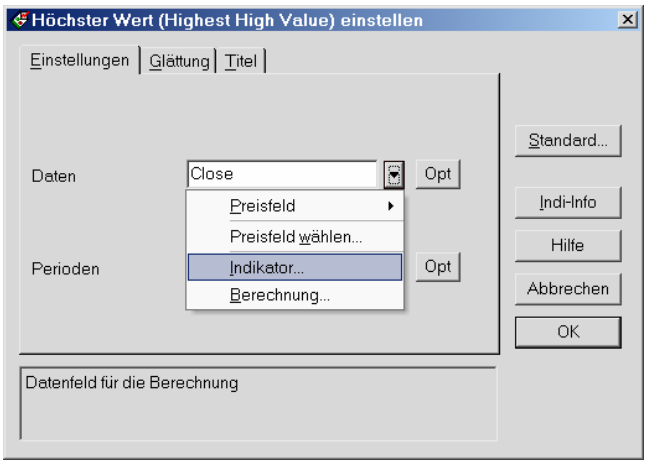
Wählen Sie im Formelfeld den Befehl „Indikator zufügen“.



In der Auswahlbox wählen Sie dann in der Kategorie „Analyse“ den Indikator „Höchster Wert (Highest High Value)“ und klicken **OK**:

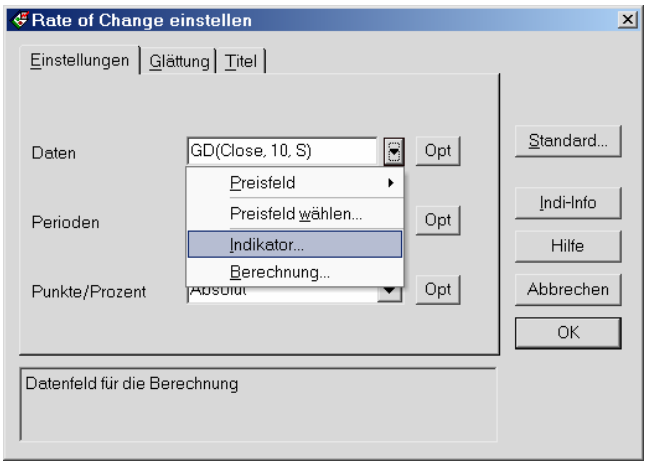


Sie gelangen in die Einstellbox für den HHV. Klicken Sie dort auf das Dropdown-Menü des „Daten“-Parameters und wählen Sie den Befehl „Indikator“:

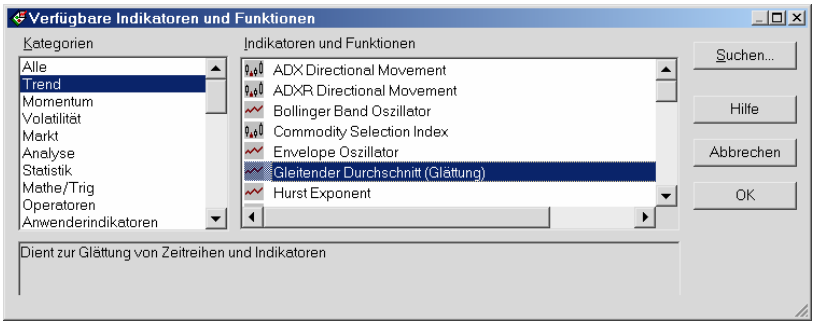


Sie gelangen damit wieder in den Auswahldialog für Indikatoren. Wählen Sie diesmal in der Kategorie „Momentum“ den Indikator „Rate of Change“, also die Wertänderung.

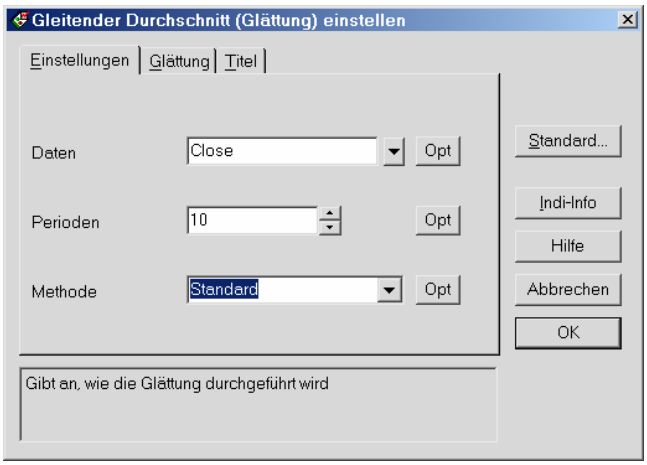
Mit **OK** gelangen Sie nun in den Einstelldialog für den ROC-Indikator, wählen Sie dort wieder den „Indikator“-Befehl im Dropdown-Menü des „Daten“-Parameters.



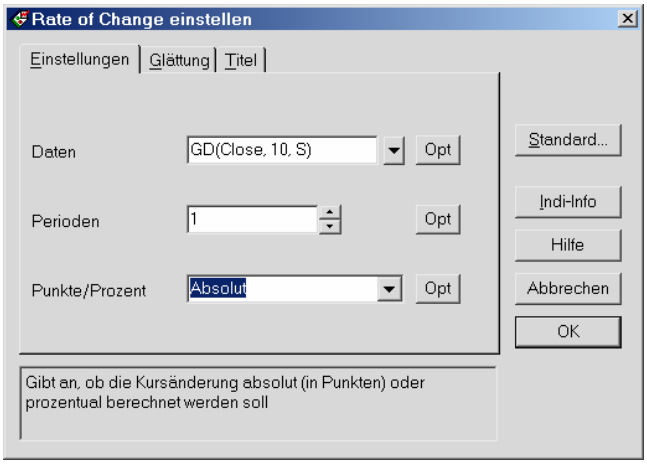
Wiederum gelangen Sie in die Auswahl der Indikatoren. Dort können wir nun unseren „innersten“ Indikator, die Glättung mit dem Gleitenden Durchschnitt auswählen. Wählen Sie also in der Kategorie „Trend“ den Indikator „Gleitender Durchschnitt“ und klicken Sie **OK**.



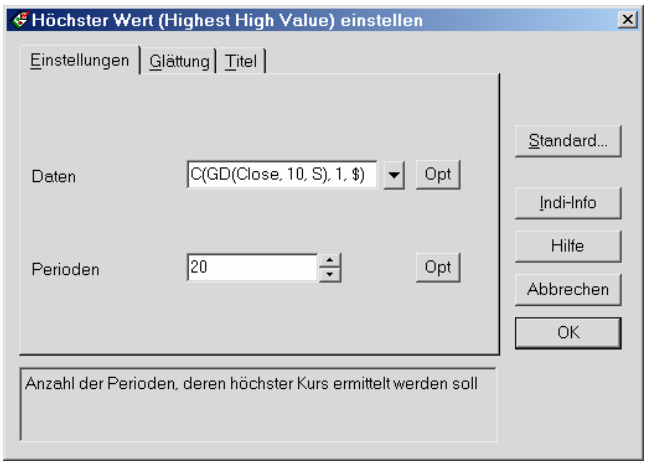
Wir befinden uns nun in der Einstellung der letzten Ebene, des ROC-Indikators. Stellen Sie also den Indikator wie gewünscht ein, auf 10 Perioden Standardglättung der Schlußkurse:



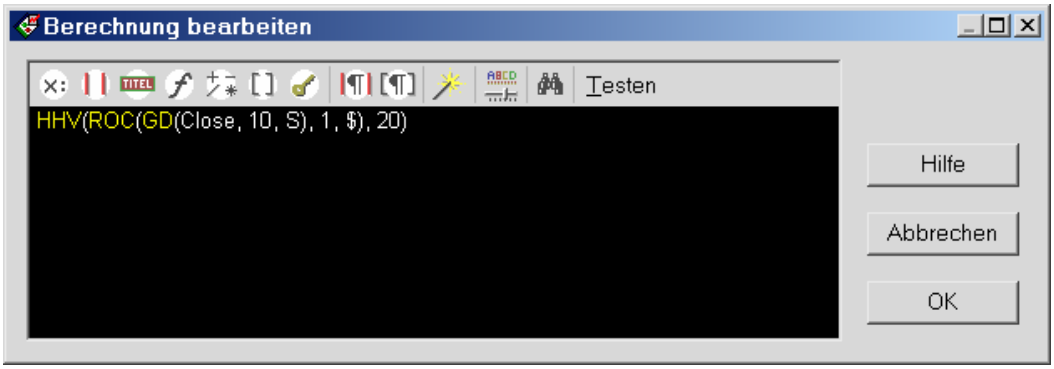
Ab jetzt geht es wieder „rückwärts“ in den Einstelldialogen. Klicken Sie OK, um in den Einstelldialog für die ROC zu gelangen. Dort sehen Sie im „Daten“-Parameter den GD. Vervollständigen Sie die Einstellung, „Perioden=1“ und „Punkte/Prozent=Absolut“.



Mit **OK** gelangen wir zur Einstellung des HHV zurück. Dort müssen wir nun nur noch die Perioden auf 20 einstellen.



Nun sind wir fertig. Sie können ein letztes Mal **OK** klicken und dann das Ergebnis im Formeleditor bestaunen:



Der Ablauf erfolgte also grob gesagt über die Stationen:

```
Formeleditor -> HHV -> ROC -> GD -> ROC -> HHV -> Formeleditor
```

Wenn Sie schon geübter mit der Formelsprache sind, kommen Sie vielleicht durch reines Schreiben ohne Dialoge schneller zum Ziel. Aber gerade beim Einstieg in die Formelsprache kann die Eingabe mit Dialogen hilfreich sein, um Fehler in der Schreibweise oder der Klammersetzung zu vermeiden.

Variablen verwenden statts Verschachteln

Mit dem Einsatz von Variablen (siehe Seite 17) lassen sich verschachtelte Berechnungen oftmals übersichtlicher darstellen. Betrachten wir wieder unsere verschachtelte Formel von oben:

```
HHV(ROC(GD(Close, 10, S), 1, $), 20)
```

Diese können wir mit Hilfe von Variablen wie folgt „entschachteln“:

```
Calc Var1: GD(Close, 10, S);
Calc Var2: ROC(Var1, 1, $);
Calc Var3: HHV(Var2, 20);
Var3
```

Wir teilen unsere Berechnung also in eine Abfolge von Variablen auf, deren Ergebnis jeweils den Daten-Parameter des nächsten Rechenschritts bildet. Zu beachten ist hier wieder die Abfolge von „innen nach außen“: Die in der Verschachtelung innen stehende Berechnung wird zuerst berechnet und muss daher auch in der Abfolge der Variablendefinitionen an erster Stelle stehen.

Eine solche Auflösung mit Variablen ist gerade bei komplexeren Verschachtelungen wesentlich übersichtlicher und damit auch leichter zu bearbeiten, wenn nachträglich etwas geändert werden soll.

Begriffe und Konventionen

Bevor wir mit Beispielen zur Formelsprache fortfahren befassen wir uns kurz mit einigen weiteren Begriffen und Konventionen der Formelsprache von Investox:

Groß/Kleinschreibung

Die Groß-/Kleinschreibung wird von Investox **nicht beachtet**. Die folgenden Schreibweisen haben also für Investox die gleiche Bedeutung:

```
mom(close, 10)
Mom(Close, 10)
MOM(CLOSE, 10)
```

Sie können die Groß-/Kleinschreibung daher auch für Ihre eigene Gestaltung der Lesbarkeit einsetzen, indem Sie verschiedene Typen von Begriffen damit unterscheiden. Eine mögliche Einteilung wäre:

Typ	Schreibweise	Beispiele
Indikator	Nur Großbuchstaben	MOM RSI
Variablen	Präfix in Kleinschreibung, z.B. „v_“ für Berechnungen und „c_“ für Konstanten	v_MeineBerechnung c_MeineKonstante
Feste Parameter, Schlüsselwörter	Anfangsbuchstabe groß	Close

Formatierung des Textes

Auch mit einer geeigneten Formatierung des Formeltextes kann man dessen Lesbarkeit verbessern. Als Trennzeichen zwischen einzelnen Begriffen können Leerzeichen, Tabs oder Zeilenschaltungen beliebiger Anzahl verwendet werden. Damit können Sie die Formatierung beeinflussen. So könnte man zum Beispiel die folgende If-Verschachtelung:

```
If(Mom(Close,10)<90,-1, If(Mom(Close,10)>110,1,0))
```

mit Tabs und Zeilenschaltungen formatieren, so dass die Struktur der Formel deutlicher wird:

```
If(Mom(Close, 10) < 90,
   1,
   If(Mom(Close, 10) > 110,
      1,
      0
   )
)
```

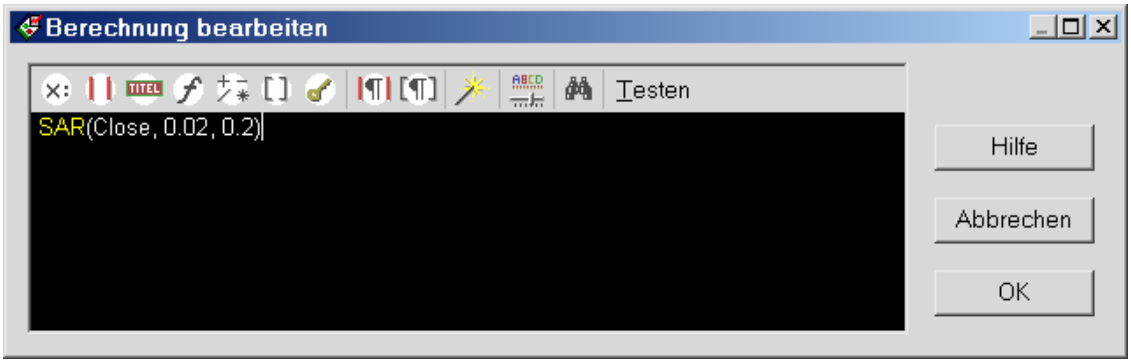
Eine solche Darstellung hat auch den Vorteil, dass die Klammersetzung leichter kontrolliert werden kann.

Dezimalzeichen

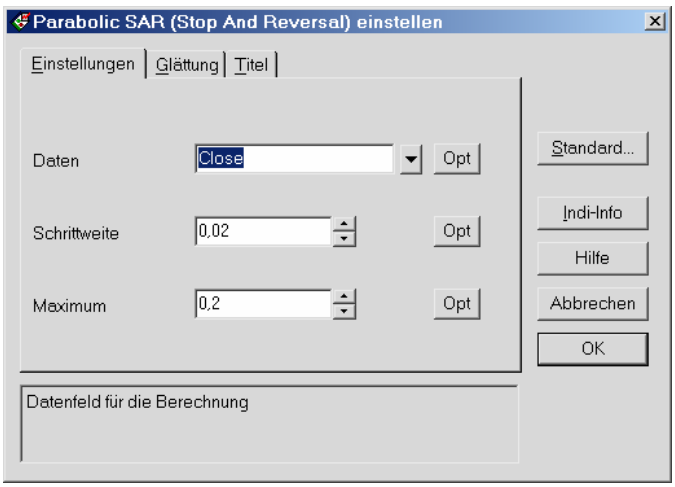
Als Dezimalzeichen in Formeln wird – unabhängig von der Ländereinstellung der Windows-Installation – ein **Punkt** verwendet. Denn das Komma wird von Investox als Trennzeichen für Listen wie zum Beispiel der Liste der Einstellparameter eines Indikators verwendet.

Zu beachten ist aber, dass bei Einstellungen von Zahlenwerten in Dialogen das Dezimalzeichen gemäß der Ländereinstellung zu verwenden ist, in Deutschland also standardmäßig ein **Komma**.

Betrachten wir als Beispiel den Indikator SAR, dessen Einstellungen Werte mit Nachkommastellen erfordern:



Das Dezimalzeichen ist hier wie gefordert ein Punkt. Wenn Sie nun auf „SAR“ doppelklicken, öffnet sich der Einstelldialog für diesen Indikator und Sie sehen, dass als Dezimalzeichen im Dialog ein Komma verwendet wird (hier mit Ländereinstellungen für Deutschland):



Beispiele für Berechnungen

In diesem Kapitel nähern wir uns der Formelsprache von Investox mit einigen konkreten Beispielen für Berechnungen, die häufig eingesetzt werden.

Bezugnahme auf Werte

Bezugnahme auf vergangene Werte

Indikator **REF**

Wenn wir Kursfelder oder Indikatoren in Berechnungen verwenden, so sprechen wir grundsätzlich erst einmal immer den aktuellen Wert der Datenperiode an. Wenn wir also schreiben:

```
Close + 10
```

so ist das Ergebnis der Schlußkurs der aktuellen Periode + 10. Oftmals ist es aber auch nötig, in Berechnungen Bezug auf einen früheren (oder vielleicht auch späteren) Wert in einer Datenreihe zu nehmen. Dazu dient in Investox der Indikator „Ref“ (Referenz auf Daten). Die Schreibweise ist:

```
Ref(Daten, Perioden)
```

Unter „Daten“ geben wir an, auf welche Daten wir Bezug nehmen, und unter „Perioden“, wie viele Perioden wir nach links (frühere Daten) oder auch nach rechts (spätere Daten) zugreifen möchten.

Der einfachste Fall ist:

```
Ref(Close, 0)
```

Dieser Ausdruck liefert einfach den aktuellen Schlusskurs (ebenso könnten Sie hierfür natürlich einfach „Close“ schreiben).

Den Schlusskurs der vorigen Periode erhalten Sie mit

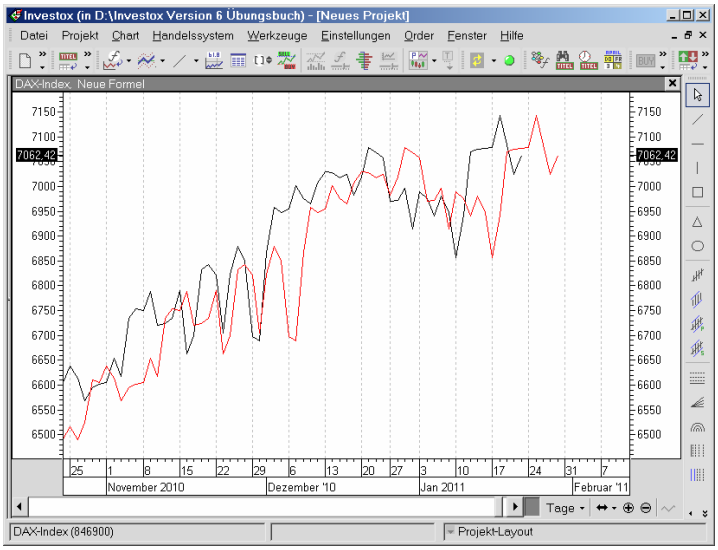
```
Ref(Close, -1)
```

Der Zugriff auf frühere Daten wird also durch eine negative Periodeneinstellung erzielt.

Den Schlusskurs vor 5 Perioden erhalten wir entsprechend durch:

```
Ref(Close, -5)
```

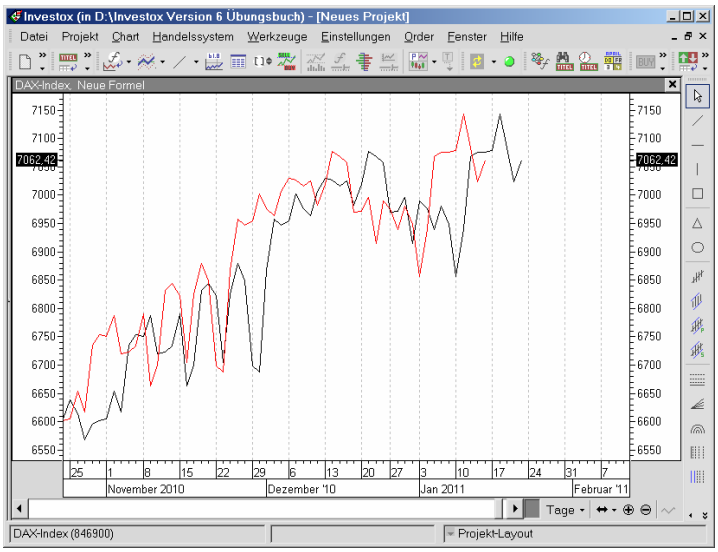
Damit die Auswirkung deutlich wird, schauen wir uns dies wieder im Chart an. Öffnen Sie den Dax-Index, zoomen Sie auf die letzten drei Monate und stellen Sie die Darstellung auf Linienchart um. Fügen Sie dann die obige Formel „Ref(Close, -5) in den Chart, und zwar in den Teilchart der Basis. Das sieht dann so aus:



Unsere REF-Berechnung (die rote Linie) eilt den Kursen um fünf Perioden voraus, ist also nach rechts verschoben. Es mag zunächst irritieren, dass eine nach vorne verschobene Berechnung nicht in die Zukunft blickt, ist aber wirklich so. Ändern Sie nun die Berechnung in:

```
Ref(Close, 5)
```

Also ein positiver Perioden-Wert. Statt fünf Perioden in die Vergangenheit blicken wir nun fünf Perioden in die Zukunft. Dies sieht im Chart so aus:



Die Kurse sind nach links, in die Vergangenheit verschoben; und dies bedeutet: sie blicken in die Zukuft. Dies ist für mechanische Handelsstrategien natürlich nicht zu gebrauchen, da Strategien mit Zukunftsblick zwar im Backtest exzellent aussehen, aber in der Praxis nicht einsetzbar sind.

Mit REF-Periodeneinstellung < 0: **Vergangenheitsblick**, Berechnung ist nach rechts verschoben.

Mit REF-Periodeneinstellung > 0: **Zukunftsblick**, Berechnung ist nach links verschoben (**Achtung!!**).

Merke: Sobald eine Berechnung nicht bis zur aktuellen Periode zur Verfügung steht (sondern weiter links endet), enthält sie vermutlich einen Zukunftsblick (oder es fehlen Daten eines Vergleichstitels, der in der Berechnung eingesetzt wird). Als **Faustregel:** Eine Berechnung schaut so viele Perioden in die Zukunft, wie sie gegenüber den Kursdaten im Chart nach links verschoben ist. Dies gilt allerdings nicht für Spezialfälle wie ZigZag, die von Haus aus in die Zukunft blicken.

Ausser in speziellen Fällen, wie zum Beispiel wenn wir das Prognoseziel für ein Neuronales Netz definieren, werden wir also für Ref( ) immer nur Perioden-Einstellungen <= 0 verwenden.

Bezug auf einen bestimmten Datenpunkt

Indikator **ValueWhen**

Eine Bezugnahme mit einem festen Perioden-Abstand ist nicht für jede Aufgabenstellung ausreichend. Angenommen, Sie möchten testen, ob der aktuelle Kurs höher ist als der Kurs war, als der MACD zuletzt unter der 0-Linie lag. Eine solche Bezugnahme auf einen Wert realisieren Sie mit dem für viele Analysen wichtigen Indikator „ValueWhen“. Die allgemeine Schreibweise des Indikators lautet:

```
ValueWhen(Daten, Bedingung, x, Richtung)
```

Liefert den Wert von „Daten“, zum Zeitpunkt als die angegebene „Bedingung“ zuletzt das x-te Mal „Wahr“ (also <> 0) war. Als Einstellung für den Parameter „Richtung“ verwenden wir normaler Weise „v“ (Vergangenheit).

Wenn wir nun unsere obige Bedingung einsetzen, so sieht die Formel wie folgt aus:

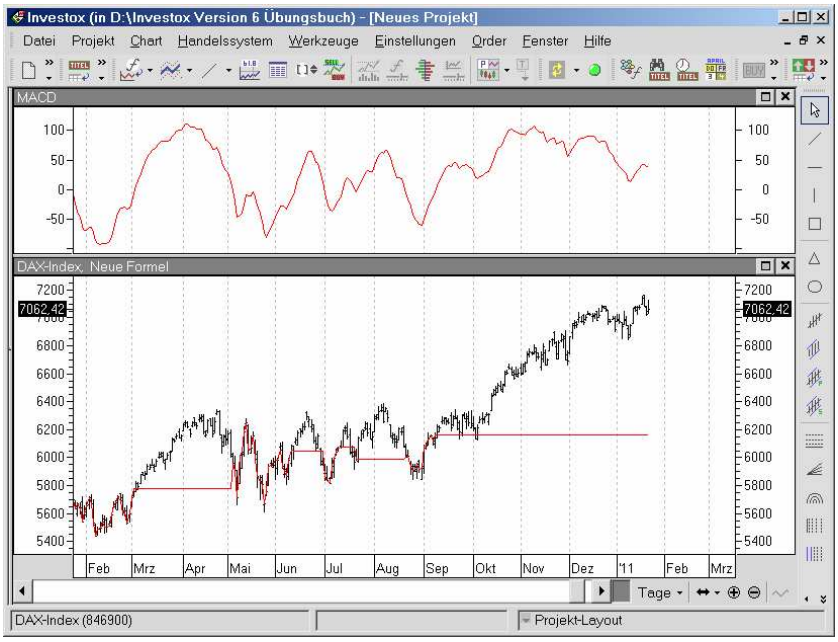
```
calc Bedingung: MACD(Close) < 0;   // MACD unter der 0-Linie
ValueWhen(Close, Bedingung, 1, V)
```

Hierbei müssen Sie nicht wissen, wie lange es jeweils her ist, dass der MACD unter der 0-Linie lag, da dies der ValueWhen-Indikator für Sie untersucht.

Fügen Sie diese Berechnung bitte in einen DAX-EoD-Chart in den Teilchart der Basis. Fügen Sie zudem in einen leeren Teilchart oben den MACD zu Vergleichszwecken zu. Die Formel dafür lautet einfach:

```
MACD(Close)
```

Im Chart sieht es dann ungefähr so aus:



Wir sehen, dass der Kurs ab dem Moment festgehalten wird, ab dem der MACD über die 0-Linie geht. Sobald der MACD unter die 0-Linie fällt, wird der Kurs ständig nachgezogen, da die Bedingung „MACD<0“ ständig in der letzten Periode zutrifft.

Sie möchten nun nicht nur den Kurs festhalten, wenn der MACD über die 0-Linie geht, sondern einfach dann, wenn er die 0-Linie kreuzt? Wie man das macht, erfahren Sie im folgenden Kapitel (Durchkreuzen von Signallinien).

Durchkreuzen von Signallinien

Indikator **Cross**

Im vorigen Kapitel ist die Frage aufgetaucht, wie wir die Stellen in der Datenreihe erhalten, an denen der MACD-Indikator die 0-Linie durchkreuzt. Am einfachsten geht dies mit dem Cross-Indikator.

Die Schreibweise für den Cross-Indikator lautet:

```
Cross(Daten, Signallinie, Perioden)
```

Unter „Daten“ geben Sie den Indikator oder die Berechnung oder die Kurse ein, die untersucht werden sollen. Der Parameter „Signallinie“ gibt wie nicht anders zu erwarten den Wert der Signallinie an, wobei dieser Wert auch eine Datenreihe wie z.B. ein anderer Indikator sein kann.

Der letzte Parameter „Perioden“ ermöglicht eine gewisse Toleranz für die Untersuchung. Er gibt an, wie viele Perioden das Durchkreuzen höchstens zurück liegen darf. Eine Einstellung >1 kann hier sinnvoll sein, wenn das Durchkreuzen noch mit anderen Bedingungen kombiniert wird.

Wie sieht es jetzt mit unserem MACD und der 0-Linie aus?

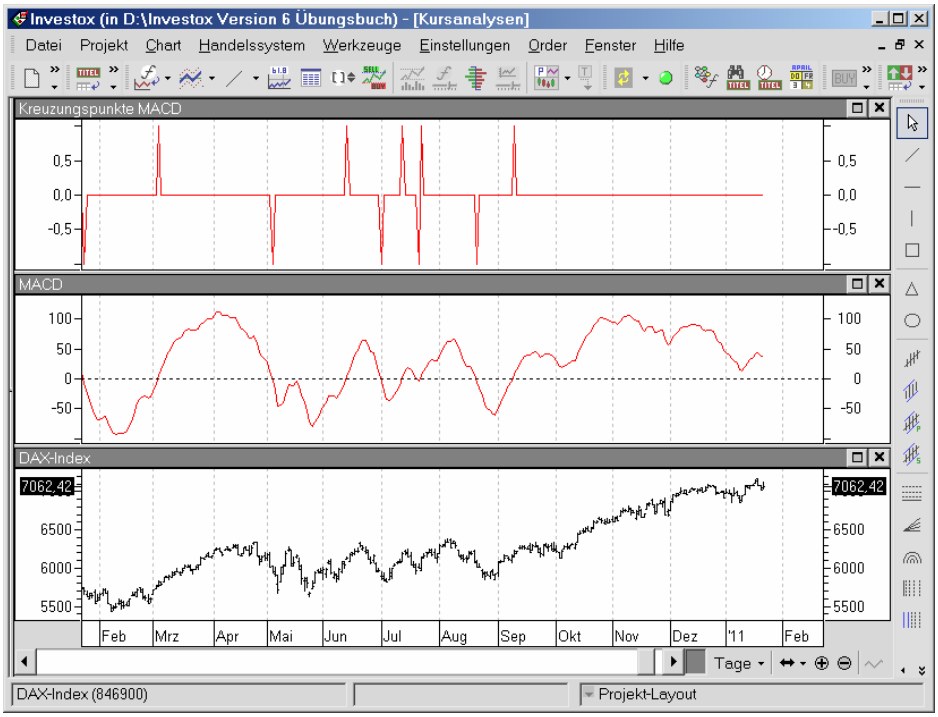
Fügen Sie in den EoD-Chart des DAX in einen leeren Teilchart oben den MACD:

```
MACD(Close)
```

und in einem weiteren Teilchart darüber die folgende Formel ein:

```
Cross(MACD(Close), 0, 1)
```

Das Ergebnis im Chart ist wie folgt:



Der Cross-Indikator (oberer Teilchart) liefert die meiste Zeit den Wert 0, nämlich dann wenn der MACD (mittlerer Teilchart) gerade die 0-Linie nicht kreuzt. An manchen Stellen ist das Ergebnis dagegen „1“ und an anderen Stellen „-1“. Wie Sie schnell erkennen, liefert „Cross“ einen unterschiedlichen Wert je nachdem, ob die Signallinie nach oben oder nach unten durchkreuzt wird.

Durchkreuzen von **unten nach oben**: Ergebnis = **1**

Durchkreuzen von **oben nach unten**: Ergebnis = **-1**

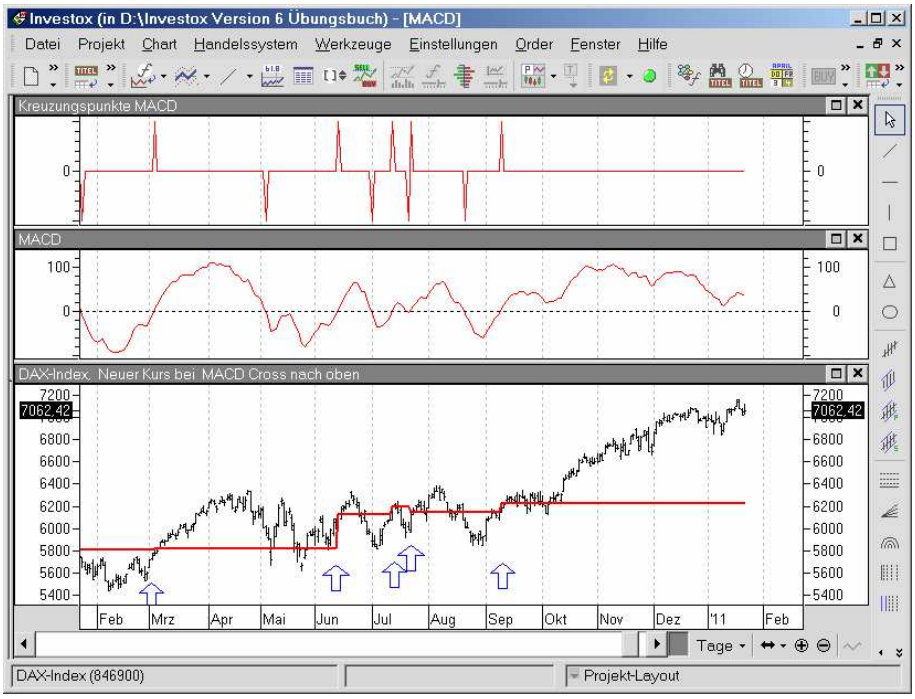
Der Cross-Indikator kann also sowohl für das Durchkreuzen einer Signallinie von unten nach oben, als auch für das Durchkreuzen der Linie von oben nach unten eingesetzt werden. Im ersten Falle liefert er als Ergebnis 1, im zweiten Fall dagegen –1.

Was nützt uns das, wenn uns nur der Fall interessiert, dass die Signallinie von unten nach oben durchkreuzt wird? Ganz einfach: Wir müssen dann nur den Fall Cross = 1 untersuchen. Wenden wir dies auf unser Beispiel im vorigen Kapitel an. Mit ValueWhen() soll der Schlußkurs festgehalten werden, wenn der MACD die 0-Linie nach oben durchkreuzt.

Fügen Sie hierzu in den Teilchart der Basis die folgende Formel ein.

```
// MACD kreuzt 0-Linie von unten nach oben:
calc Bedingung: Cross(MACD(Close), 0, 1) = 1;
ValueWhen(Close, Bedingung, 1, V)
```

Im Chart erhalten wir:



Der Kurs wird also nur neu gesetzt (Pfeile), wenn der MACD die 0-Linie nach oben kreuzt, in allen anderen Fällen bleibt er unverändert.

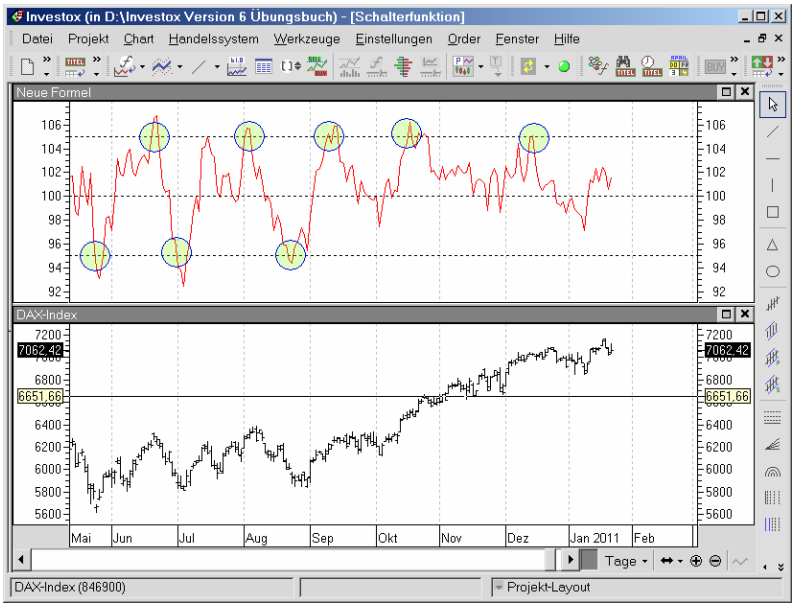
Indikatorenzustände festhalten

Indikator **Schalter**

Angenommen, wir möchten die Trendrichtung des Marktes mit Hilfe des Momentum-Indikators bestimmen. Die Berechnung soll je nach Trend den Wert 1 (Aufwärtstrend) bzw. -1 (Abwärtstrend) ausgeben. Und zwar nach folgender Regel:

- Betrachte das Momentum über 10 Perioden
- Nehme einen Aufwärtstrend (= 1) an, sobald das Momentum die 105er-Linie nach oben durchkreuzt
- Nehme einen Abwärtstrend (= -1) an, sobald das Momentum die 95er-Linie nach unten durchkreuzt

Schauen wir uns dies einmal im Chart an. Öffnen Sie den Dax-Index und zoomen Sie auf die Kurse vom Mai bis Dezember 2010. Fügen Sie dann in den oberen Teilchart ein Momentum über 10 Perioden ein. Im folgenden Bild haben wir zudem horizontale Linien auf 95 und 105 eingefügt und die Kreuzungspunkte mit Kreisen markiert:



Fassen wir unsere drei Bedingungen von oben nun in eine Formel, sieht diese wie folgt aus:

```
calc Mom: MOM(Close, 10); // Definition des Momentums über 10 Perioden
calc CrossNachOben: Cross(Mom, 105, 1) = 1;  // Momentum kreuzt die 105er-Linie
nach oben
calc CrossNachUnten: Cross(Mom, 95, 1) = -1;  // Momentum kreuzt die 95-Linie
nach unten
```

Wie bekommen wir daraus nun aber unser Ergebnis, nämlich einen fortgeführten Wert -1 / 1 je nach Trendrichtung? Ein erster (verzweifelter) Versuch könnte so aussehen, dass wir die folgende Zeile ergänzen:

```
If(CrossNachOben, 1, If(CrossNachUnten, -1, 0)) // Gebe -1 oder 1 aus
```

Auch bei der Verschachtelung des If-Indikators ist die Klammersetzung zu beachten. In diesem Fall vor allem die zwei schließenden Klammern am Ende.

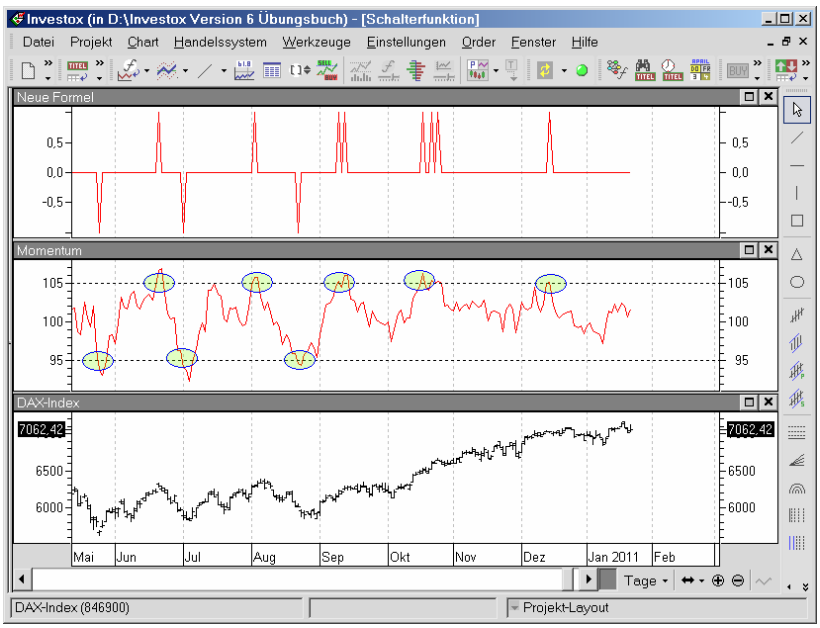
Der If-Indikator vollzieht für uns eine „Wenn-Dann-Ansonsten“-Betrachtung. Umgangssprachlich lässt sich dies so darstellen:

Wenn die Bedingung „CrossNachOben“ zutrifft: **gebe 1 aus,**

ansonsten, wenn die Bedingung „CrossNachUnten“ zutrifft: **gebe -1 aus**,

ansonsten: **gebe 0 aus**.

Dies schauen wir uns wieder im Chart an. Fügen Sie also die vier Zeilen von oben in den oberen Teilchart als Formel ein. Dies sieht dann so aus:



Dies ist allerdings noch nicht das, was wir eigentlich wollten. Wir möchten ja nicht nur die Perioden der Kreuzungspunkte markieren, sondern auch die Bereiche dazwischen mit dem jeweils letzten Zustand, ohne dass die Berechnung zwischendurch auf den Wert 0 wechselt. Hierbei hilft uns nun der **Schalter**-Indikator:

```
Schalter(Startwert, 1. Bedingung, 1. Wert, 2. Bedingung, 2. Wert)
```

Dieser Indikator schaltet zwischen zwei Werten um, wenn die jeweilige Bedingung zutrifft. Die Berechnung beginnt mit dem angegebenen Startwert. Sobald eine der beiden Bedingungen zutrifft, wechselt der Indikator in den zugehörigen Wert. Dieser Wert wird solange beibehalten, bis wiederum eine der beiden Bedingungen zutrifft.

In unserem Fall sieht dies konkret so aus:

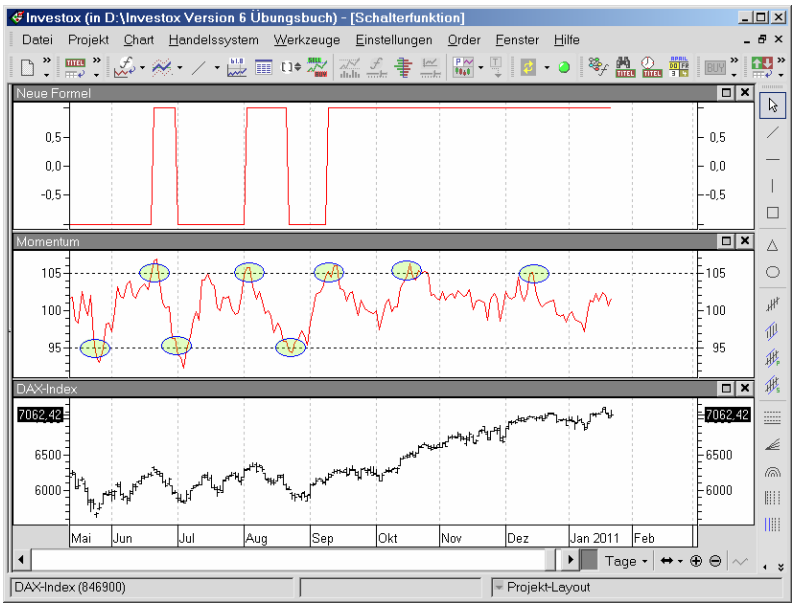
```
Schalter(0, CrossNachOben, 1, CrossNachUnten, -1)  //Gebe 1 oder -1 aus
```

Umschrieben ausgedrückt: Beginne bei 0. Sobald „CrossNachOben“ zutrifft, wechsle auf 1; sobald „CrossNachUnten“ zutrifft, wechsle auf -1. Sobald der Startwert 0 verlassen wurde, wechselt der Indikator also nur noch zwischen -1 und 1.

Unsere gesamte Berechnung sieht jetzt also wie folgt aus:

```
calc Mom: MOM(Close, 10); // Definition des Momentums über 10 Perioden
// Momentum kreuzt die 105er-Linie nach oben:
calc CrossNachOben: Cross(Mom, 105, 1) = 1;
// Momentum kreuzt die 95-Linie nach unten:
calc CrossNachUnten: Cross(Mom, 95, 1) = -1;
// Gebe 1 oder -1 aus:
Schalter(0, CrossNachOben, 1, CrossNachUnten, -1)
```

Wenn Sie diese Berechnung im Chart eingeben, sollte es so aussehen:



Damit sind die Bereiche „Aufwärtstrend“ (1) und „Abwärtstrend“ (-1) wie anfangs erwünscht klar gekennzeichnet. Das Beispiel zeigt in der ersten Hälfte auch gleich einige „Fehlsignale“ im Seitwärtsmarkt, während sich in der zweiten Hälfte der angezeigte Trend dann auch durchsetzt.

Analyse von Kursverläufen und Indikatoren

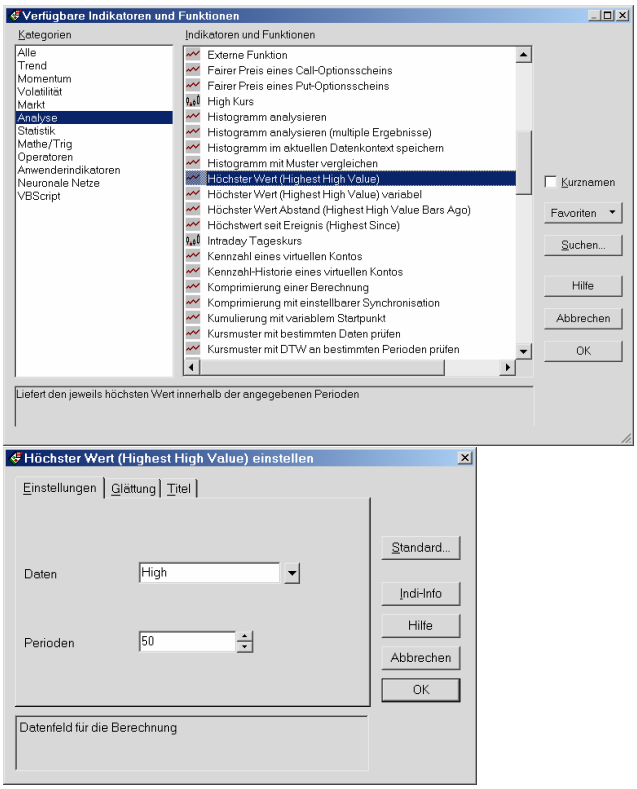
Ein wichtiger Bereich bei der Entwicklung von Strategien ist die Analyse von Kursverläufen und von Indikatoren. Dazu gehört, wie sich der Kurs in den letzten Perioden entwickelt hat, welche Höchst- und Tiefstwerte es gab, wann es zuletzt einen Hoch- oder Tiefpunkt gab und so fort.

Höchstwert / Tiefstwert

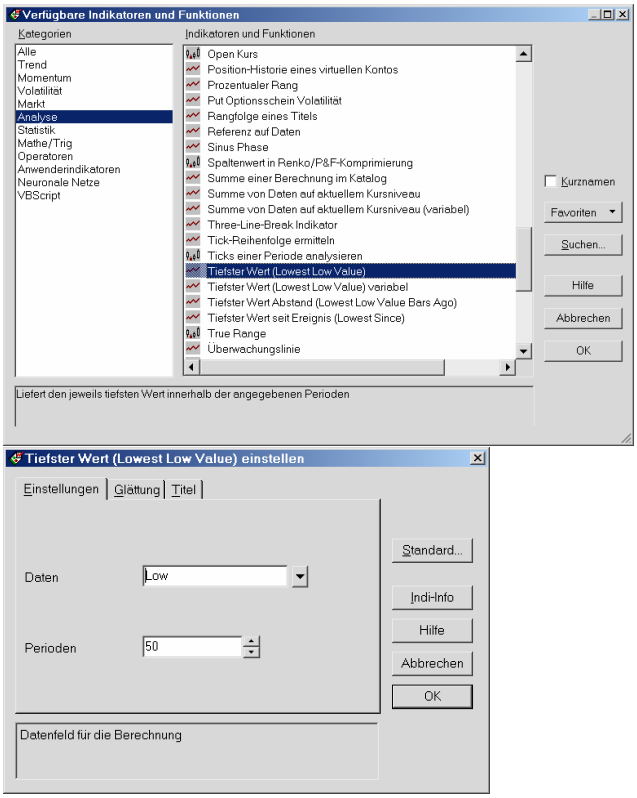
Indikatoren **HHV** und **LLV**

Mit den Indikatoren HHV und LLV können Sie den höchsten bzw. tiefsten Wert eines vergangenen Zeitraums ermitteln. Damit kann zum Beispiel festgestellt werden, in welchem Kursband sich die Kurse aufhalten oder ob sie dieses Kursband durchbrechen. Aber auch die Weite des Kursbandes kann aufschlussreich sein: je enger das Kursband, desto geringer waren die Kursschwankungen im betrachteten Zeitraum. Eine solche Ermittlung der Volatilität schauen wir uns nun genauer an:

Öffnen Sie einen Chart mit dem Dax-Index und zoomen Sie diesmal auf einen längerfristigen Bereich, nämlich auf die Jahre 2007 bis 2010. Fügen Sie dann zunächst den Indikator „Höchster Wert“ (HHV) aus der Kategorie „Analyse“ in den Chart ein. Wählen Sie als Einstellung für die Daten „High“ und 50 für die Perioden:

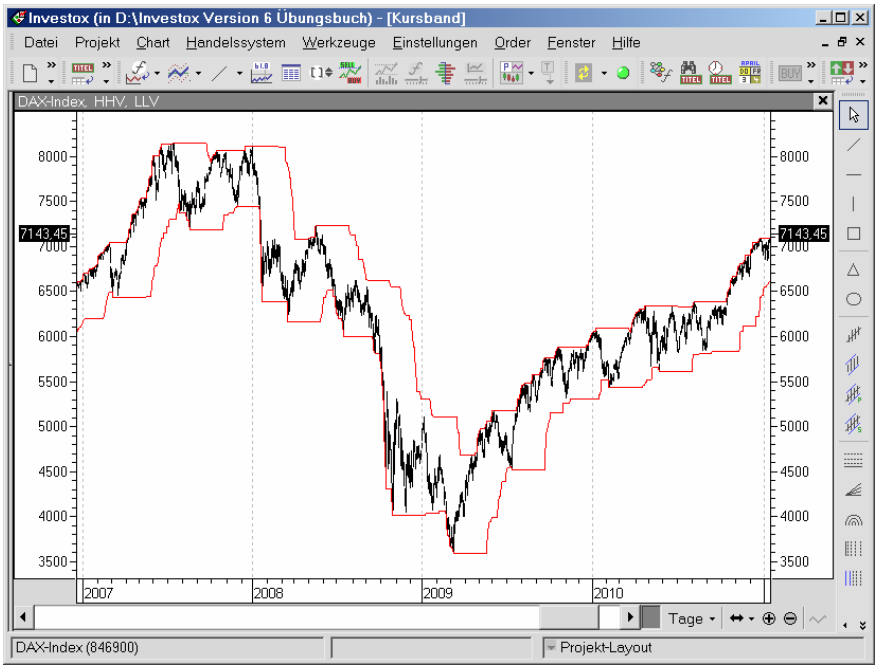


Nun fügen wir noch den Indikator „Tiefster Wert“ (LLV) hinzu: Stellen Sie diesmal die Daten auf „Low“, die Perioden wiederum auf 50:



Im Chart sehen wir nun das Kursband über 50 Perioden:

Die Indikatoren HHV und LLV und die Datenreihe sollten auf dieselbe Achse skaliert werden (standardmäßig auf die linke Achse).



Es ist zu erkennen, dass sich das Kursband dann ausweitet, wenn die Kurse schnell in eine Richtung ausbrechen. Umgekehrt wird es enger, wenn die Kurse wieder eher in eine Seitwärtsbewegung übergehen – jeweils betrachtet über einen Zeithorizont von 50 Tagen.

Wie können wir nun diese Beobachtung mit einer Formel genau messen? Der erste Schritt in unserer Berechnung ist die Formulierung der verwendeten Komponenten, also die Begrenzung des Kursbandes durch HHV und LLV:

```
const P: 50;  // Die Periodeneinstellung als Konstante
calc BandOben: HHV(High, P);  // Der Hochkurs
calc BandUnten: LLV(Low, P);   // Der Tiefkurs
```

Die Periodeneinstellung erfolgt über eine Konstante (so müssen wir nur eine Zahl ändern, wenn wir den Zeithorizont ändern möchten). Die Benennungen „BandOben“ und „BandUnten“ sind dabei willkürlich von uns gewählt.

Die Breite des Kursbandes erhalten wir nun einfach durch die Subtraktion des unteren Bandes (LLV) vom oberen Band (HHV):

```
BandOben - BandUnten
```

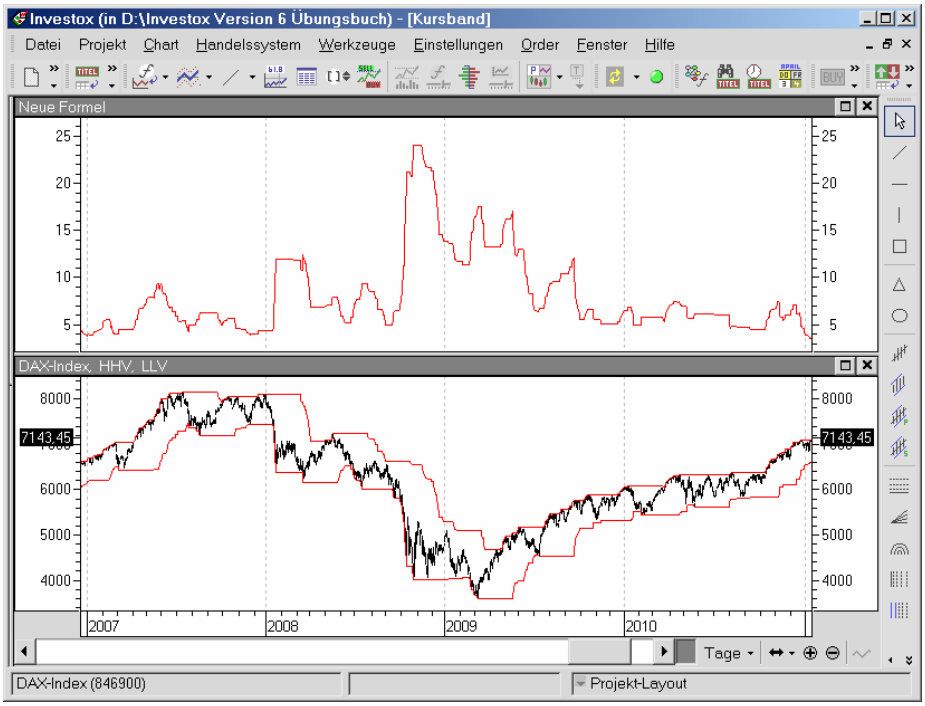
Gerade bei längerfristigen Betrachtungen ist es aber meistens sinnvoll, nicht mit absoluten, sondern mit prozentualen Werten zu rechnen, da sonst das Ergebnis von der absoluten Höhe der Kurse abhängt. Und diese kann sich über längere Zeit hinweg stark ändern, so dass der Indikator in verschiedenen Zeitbereichen nicht mehr vergleichbar ist. Daher setzen wir die Breite des Kursbandes nun noch in ein Verhältnis zur absoluten Kurshöhe, nämlich zur Summe des oberen und des unteren Bandes:

```
(BandOben - BandUnten) / (BandOben +  BandUnten) * 100
```

Der Faktor 100 dient hier nur zur „Verschönerung“ der Ergebnisse, so dass diese in gewohnten Prozentwerten, nicht Dezimal angezeigt werden. Fügen Sie nun also die folgende komplette Formel in einen neuen oberen Teilchart ein:

```
const P: 50;  // Die Periodeneinstellung als Konstante
calc BandOben: HHV(High, P);  // Der Hochkurs
calc BandUnten: LLV(Low, P);  // Der Tiefkurs
// Die prozentuale Bandbreite:
(BandOben - BandUnten) / (BandOben +  BandUnten) * 100
```

Dies sollte dann so aussehen:



Der Indikator im oberen Teilchart zeigt schön an, in welchen Bereichen sich die Kurse schnell in eine Richtung entwickeln (der Indikator steigt) bzw. umgekehrt, wo eher eine Seitwärtsbewegung vorliegt (niedrige Werte des Indikators). Dabei betrachten wir natürlich immer nur jeweils die letzten 50 Perioden.

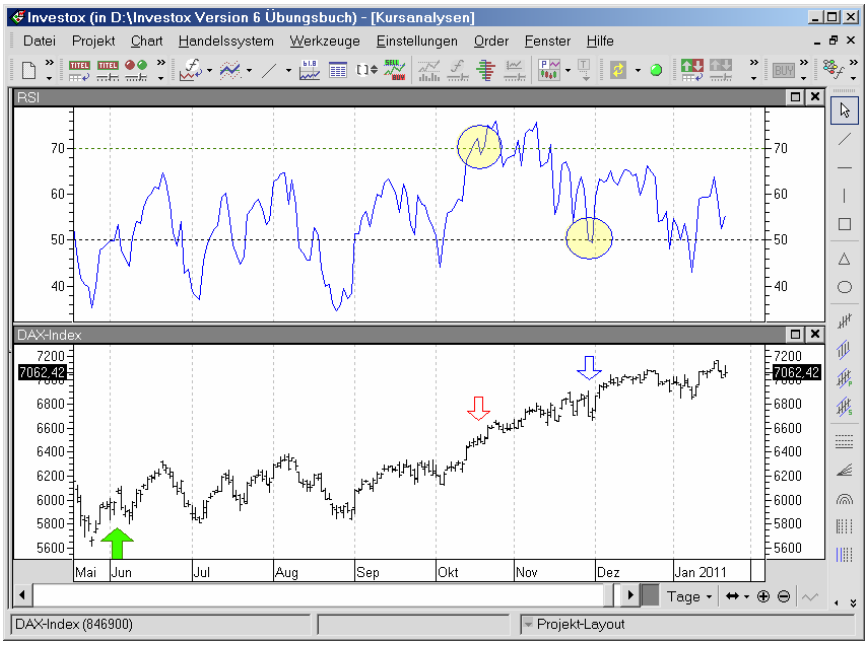
Verändern Sie als Übung nun die Einstellung für „P“ und beobachten Sie, wie sich der Indikator ändert.

Variable Hoch-/Tiefpunkt-Bestimmung

Indikatoren **HighestSince**, **LowestSince**

In manchen Fällen ist es bei der Bestimmung des höchsten bzw. tiefsten Kurses nicht zielführend oder ausreichend, einen festen Zeitraum zu betrachten. Daher gibt es mit HighestSince/LowestSince in Investox auch Indikatoren, die den Hoch-/Tiefpunkt seit einem bestimmten „Ereignis“ liefern. Dies sehen wir uns an einem konkreten Beispiel einer Analyse mit dem RSI-Indikator an.

Im folgenden EoD-Chart des DAX haben wir die Chartstudie „RSI“ eingefügt, die einen RSI mit der Standardeinstellung 14 Perioden zeichnet, und diesen durch eine 50er-Linie ergänzt:



Wir nehmen nun den Fall an, dass wir Anfang Juni 2011 eine Long-Position eingegangen wären (grüner Aufwärtspfeil) und mit Hilfe des RSI einen guten Ausstiegspunkt finden wollen. Ein klassisches (Überkauf-) Ausstiegssignal ist es, wenn der RSI die 70er-Linie nach unten durchkreuzt. Dieser Fall tritt im Chart Mitte Oktober ein (am RSI durch einen Kreis markiert). Wir sehen in diesem Fall aber, dass die Kurse weiter steigen und der RSI entsprechend auch hohe Werte beibehält.

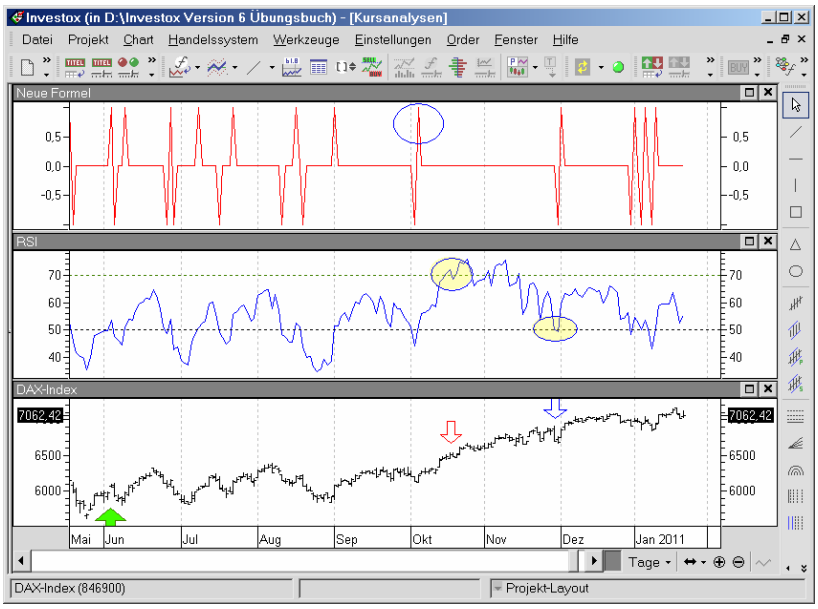
Um die weiter steigenden Kurse mitzunehmen, müssen wir also abzuwarten, bis der RSI ein deutlicheres Verkaufssignal gibt, also zum Beispiel die 50er-Linie nach unten durchkreuzt. Dies ist Ende November der Fall (wiederum am RSI mit einem Kreis markiert). Ein Blick auf den Chart zeigt aber, dass der RSI in den Monaten zuvor im Seitwärtsmarkt schon des öfteren die 50er-Linie durchkreuzt hat, so dass wir dort viele Fehlsignale erhalten hätten. Um einen Ausstieg erst Ende November zu erhalten müssen wir also sagen:

Handelsregel Exit Long: „Steige aus, wenn der RSI unter 50 fällt – aber nur, wenn er zuvor Werte über 70 erreicht hat.“

Den Bestandteil „unter 50 fällt“ haben wir schnell formuliert, da wir den Cross-Indikator oben schon kennengelernt haben. Fügen Sie die folgende Formel in den oberen Teilchart ein:

```
calc RSI: RSI(Close, 14);              //Der RSI mit Standardeinstellung
calc Cross50: Cross(RSI,50,1);         //Durchkreuzen der 50-Linie
Cross50
```

Cross50 liefert eine „1“, wenn die 50er Linie nach oben und eine „-1“, wenn die Linie nach unten durchkreuzt wird:



Betrachten wir nun den Bestandteil der Handelsregel „wenn er zuvor Werte über 70 erreicht hat“. Was genau meint hier nun „zuvor“? Sicherlich nicht: „*irgendwann* davor“, denn ein RSI erreicht des öfteren die 70er-Linie, so dass dieser Bestandteil dann ja immer zutreffen würde. Wenn wir auf den Chart blicken, können wir „zuvor“ genauer bestimmten als: „seit dem letzten Durchkreuzen der 50er-Linie nach oben“ (mit Kreis markierte „1“ im oberen Teilchart).

Wir müssen also mit anderen Worten den höchsten Wert des RSI seit dem letzten Durchkreuzen der 50er-Linie nach oben ermitteln und dann prüfen, ob dieser Wert > 70 ist. Hierbei hilft uns nun der Indikator „HighestSince“, der den Höchstwert von Daten seit einem Ereignis liefert:

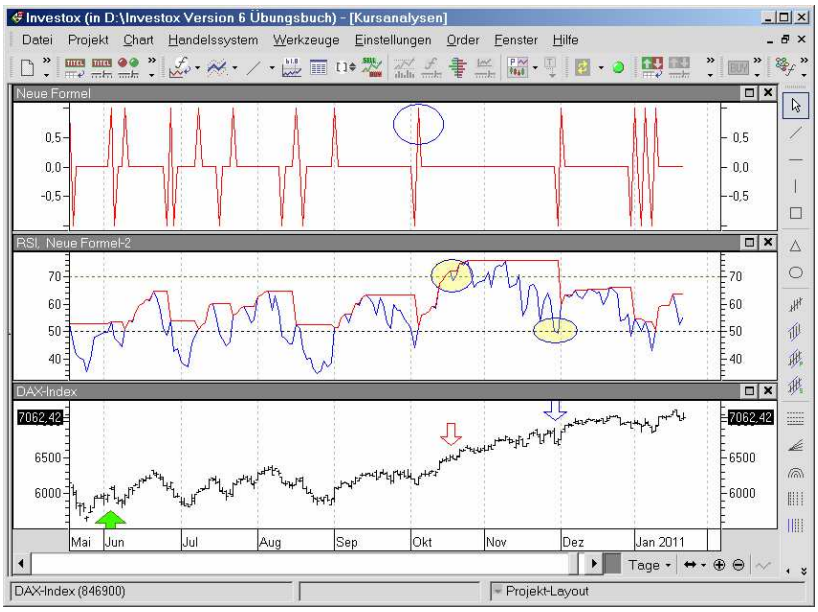
```
HighestSince(Daten, Ereignis, X)
```

In unserem Fall sind die „Daten“ der RSI und das „Ereignis“ ist das Durchkreuzen der 50er-Linie nach oben. Den dritten Parameter „X“ setzen wir auf „1“, was bedeutet, dass wir genau das letzte Ereignis meinen (mit z.B. „2“ würden wir uns auf das vorletzte Ereignis beziehen). Wir erhalten also:

```
calc RSI: RSI(Close, 14);              //Der RSI mit Standardeinstellung
calc Cross50: Cross(RSI, 50, 1);       //Durchkreuzen der 50-Linie
HighestSince(RSI, Cross50 = 1, 1)
```

Wenn Sie diese Formel in den Teilchart des RSI einfügen, erhalten Sie folgendes Bild:

Die Formel und der RSI müssen für eine korrekte Anzeige auf dieselbe Y-Achse skaliert werden (standardmäßig auf die linke Achse).

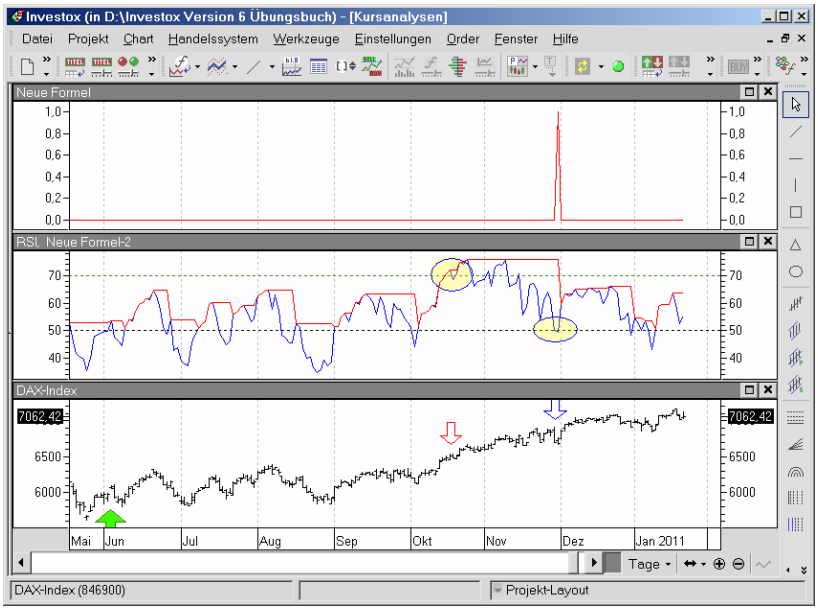


Wie man sieht, hält unsere Berechnung (rote Linie im mittleren Teilchart) den Höchstwert des RSI seit dem letzten Durchkreuzen der 50er-Linie nach oben fest. Nun müssen wir die Formel nur noch um die beiden Bestandteile ergänzen:

- Der festgehaltene Höchstwert des RSI muss > **70** sein und
- Es muss ein Durchkreuzen der 50er-Linie nach unten stattfinden (**AND Cross50 = -1**)

```
calc RSI: RSI(Close, 14);              //Der RSI mit Standardeinstellung
calc Cross50: Cross(RSI, 50, 1);       //Durchkreuzen der 50-Linie
HighestSince(RSI, Cross50 = 1, 1) > 70
AND
Cross50 = -1
```

Ersetzen Sie die Berechnung im oberesten Teilchart durch diese Formel, dann sehen Sie folgendes Endergebnis:



Die gewünschte Periode ist damit korrekt markiert und wir könnten diese Berechnung in dieser Form als Exit-Regel einsetzen.

Pivots

Indikator **LastDP**

Eine beliebte Analysemethode sind die sogenannten Pivot-Punkte, die Widerstand- und Unterstützungspunkte aus den Kursen des Vortages ableiten sollen. Typischer Weise werden Pivots in Intraday-Charts angewendet. Grundlage des Pivot-Punkt-Systems ist zunächst der eigentliche Pivot-Punkt (PP), der wie folgt berechnet wird:

```
(„Vortageshoch“ + „Vortagestief“ + „VortagesClose“) / 3
```

Der Drehpunkt des Tages wird also durch das Mittel aus Hoch-, Tief- und Schlußkursen des Vortages beschrieben.

In Investox steht uns für die Berechnung der OHLC-Kurse des Vortages mit Intraday-Kursen der effiziente Indikator „LastDP“ (letzter Daily Price) zur Verfügung. Damit berechnen wir den Pivot-Punkt wie folgt:

```
calc PP: (LastDP(High) + LastDP(Low) + LastDP(Close)) / 3;
```

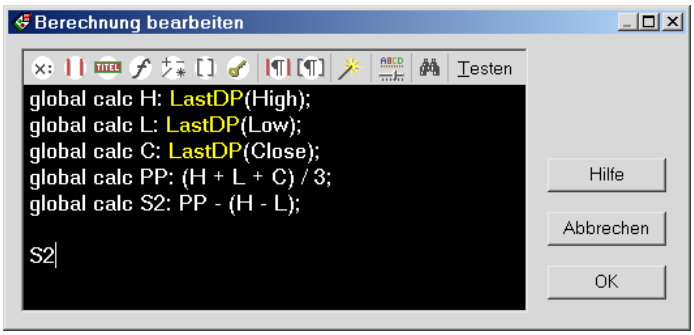
Ausgehend vom Drehpunkt PP stehen dann jeweils drei Widerstandslinien (Resist) R1, R2 und R3 und Unterstützungslinien (Support) S1, S2 und S3 zur Verfügung.

Hier folgt ein Formelblock, wie man die Pivot-Punkte effizient berechnen kann. Wir definieren alle Variablen mit „global calc“, damit der spätere Einsatz in einem Handelssystem möglichst effizient ist:

```
global calc H: LastDP(High);
global calc L: LastDP(Low);
global calc C: LastDP(Close);
global calc PP: (H + L + C) / 3;
global calc R1: 2 * PP – L;
global calc R2: PP + (H - L);
global calc R3: 2 * (PP- L) + H;
global calc S1: 2 * PP – H;
global calc S2: PP - (H - L);
global calc S3: L - 2 * (H - PP);
```

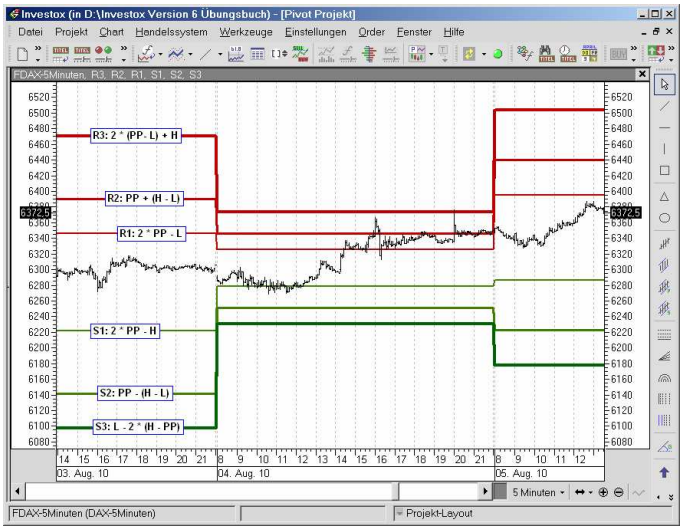
Die obersten vier Zeilen (in Fettschrift) bilden das Gerüst. Es werden hier zunächst die Vortageskurse definiert und den Variablen H, L und C zugewiesen. Die vierte Zeile berechnet dann den Pivot-Punkt PP.

Möchte man zum Beispiel die Linie S2 in den Chart einfügen, verwendet man die ersten vier Zeilen sowie die Zeile, in der S2 definiert ist. Die letzte Zeile besagt dann, dass das Ergebnis von S2 verwendet (gezeichnet) werden soll:



Fügt man nacheinander die Linien R1, R2, R3, S1, S2 und S3 in den Chart (FDAX 3. – 5. August 2010, 5-Minuten) ein, sieht dies etwa wie folgt aus:

Achten Sie darauf, dass die Linien auf dieselbe Achse skaliert werden wie die Basisdatenreihe, also alle Datenreihen auf die linke Achse oder auf die rechte Achse mischen.



Es werden also für die gesamte geladene Datenreihe laufend die Pivots für den jeweiligen Tag berechnet.

Die Linien zeigen nicht nur Widerstände und Unterstützungen, die von Händlern beachtet werden, sondern verdeutlichen auch die Entwicklung der Volatilität, also der Schwankungsbreite der Kurse des Vortages. Wird diese geringer, ziehen sich die Linien enger zusammen.

Hinweis: Wird der gesamte obige Formelblock in die Definitionen eines Handelssystems eingefügt, stehen die sieben Linien des Pivot-Systems in allen Handelsregeln zur Verfügung. Ebenso auch in Formeln im Chart, wenn dort die globalen Variablen mit dem Schlüsselwort #_LoadDefs# eingebunden werden.

Datumsberechnungen

Für einige Zwecke ist es notwendig, Datumsfunktionen in Berechnungen zu verwenden. Dies ist dann der Fall, wenn ein bestimmter Zeitstempel, ein bestimmter Zeitraum oder ein periodischer Datumswechsel benötigt wird. Für die unterschiedlichen Zwecke bietet Investox hierfür spezielle Indikatoren. Wir lernen sie in den folgenden Beispielen kennen:

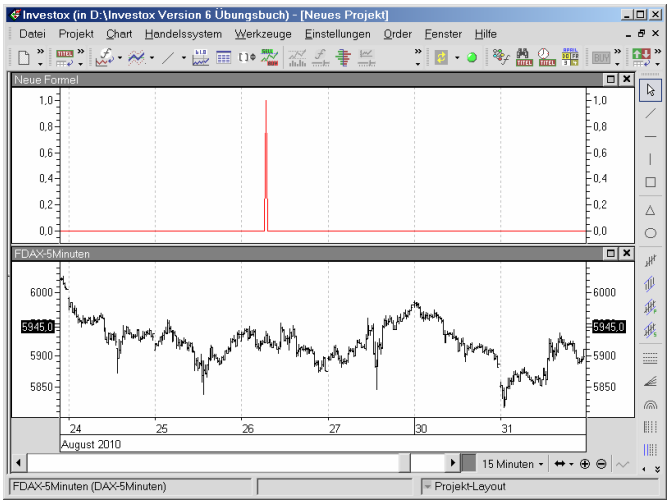
Markierung eines bestimmten Zeitstempels

Indikatoren **DateMark**, **ValueWhen**

Mit dem Indikator „DateMark“ können wir einen bestimmten Zeitstempel „markieren“. Um dies auszuprobieren, öffnen Sie bitte einen Intraday-FDAX-Chart mit 15-Minuten und stellen den Chartausschnitt 24. – 31. August 2010 ein. Wir möchten nun genau den Zeitstempel 26. August 2010 um 12:00 Uhr markieren. Fügen Sie dazu in den oberen Teilchart die folgende Formel ein (der Indikator lässt sich am bequemsten mit der Indikator-Einstellbox einstellen):

```
DateMark(26, 8, 2010, 12, 0)
```

Im Chart sollte dies dann ungefähr so aussehen:



Der Indikator liefert also überall den Wert 0, ausser genau an dem von uns angegebenen Zeitstempel, an dem er den Wert 1 liefert. Für sich betrachtet ist dies relativ nutzlos. Das Markieren von Datenpunkten ist aber zum Beispiel für Überwachungslinien in Strategien (siehe Indikator „Überwachungslinie“) oder auch zum Festhalten des Kurses an einem bestimmten Datum nützlich.

Dieses „Festhalten“ eines Kurses betrachten wir nun näher. Wir benötigen dazu die wichtige Funktion „ValueWhen“, die den Wert liefert, den eine Datenreihe an einem bestimmten Punkt hatte, nämlich genau dann, wenn die angegebene Bedingung zuletzt zutraf.

```
ValueWhen(Daten, Bedingung, 1, V)
```

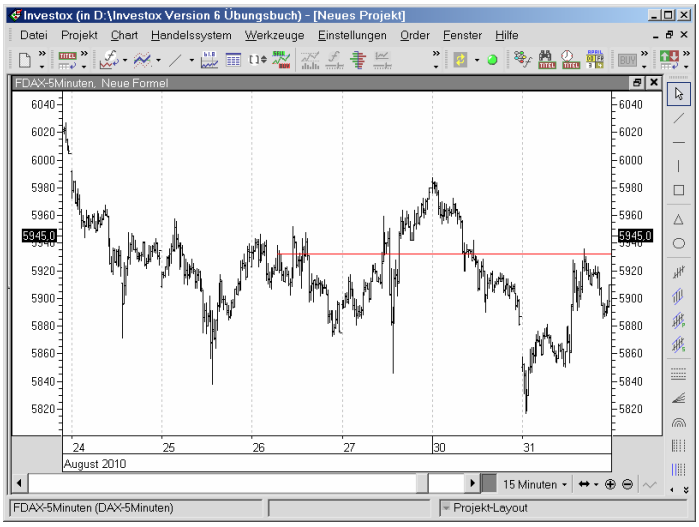
Daten: Daten, deren Wert geliefert werden soll.

Bedingung: Gibt an, ab welchem Punkt der Wert geliefert werden soll.

Ändern Sie die obige Formel wie folgt um in:

```
Calc Datum: DateMark(26, 8, 2010, 12, 0);
ValueWhen(Close, Datum, 1, V)
```

Verschieben Sie dann die Berechnung in den Teilchart der Basis und vergrößeren Sie diesen Teilchart. Der Chart sieht nun wie folgt aus:



Wie gewünscht zeichnet unsere Formel den Schlußkurs des 26.8.2010 - 12:00 Uhr fort und wir können dann untersuchen, wann dieser Wert über- oder unterschritten wurde.

Beachte: Die Linie mit dem festgehaltenen Kurs beginnt erst an dem definierten Datum. Davor kann ValueWhen keinen Wert liefern, da hierfür nichts definiert ist. Beim Einsatz von ValueWhen muss man daher darauf achten, dass die angegebene Bedingung mindestens einmal in der vorhandenen Datenhistorie zutrifft, da sonst die gesamte Berechnung, in der ValueWhen auftritt, nicht berechnet werden kann (es wird dann eine Fehlermeldung ausgegeben).

Einen bestimmten Zeitraum verwenden

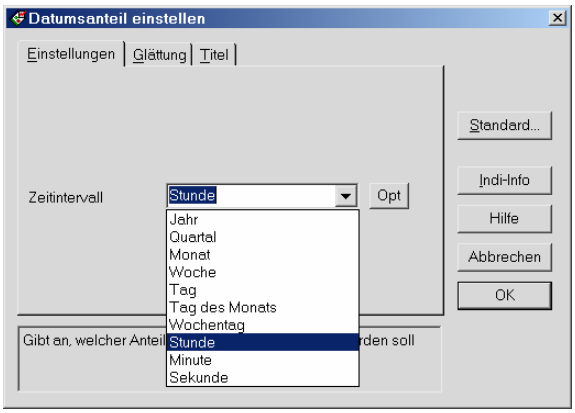
Indikator **DatePart**

Manchmal ist es wichtig, einen bestimmten Zeitraum zu kennzeichnen. Ein praktisches Beispiel dafür ist: Handle nur zwischen 9:30 und 12:00 sowie zwischen 14:00 und 17:30, oder handle nach unterschiedlichen Regeln in diesen Zeiträumen. Für solche Zwecke steht uns der DatePart()-Indikator zur Verfügung. Dieser Indikator zeigt uns den Teil aus dem Datum oder der Uhrzeit an. Was dies bedeutet verstehen wir am besten in einem Beispiel.

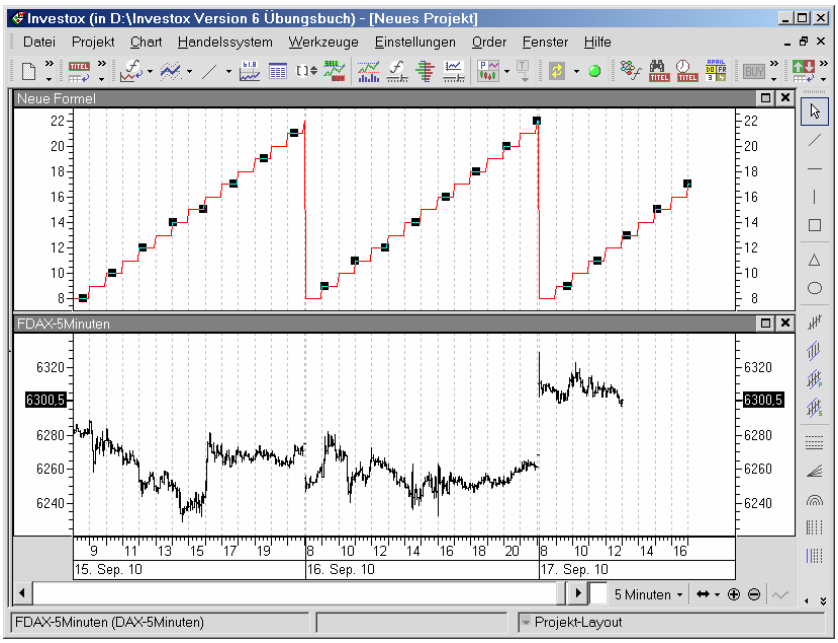
Öffnen Sie einen Intraday-Chart (FDAX, 5-Minuten) und zoomen Sie auf die letzten drei Tage. Fügen Sie dann in einen leeren oberen Teilchart die folgende Formel ein:

```
DatePart(h)
```

Doppelklicken Sie im Formeleditor auf die gelbe „Datepart“-Bezeichnung. In der Indikator-Einstellbox können Sie sich dann in der Dropdown-Liste die Einstellmöglichkeiten des Indikators ansehen:



Wir lassen uns also die Stunde des Tages anzeigen. Eingefügt im Chart sieht dies so aus:



Wie erwartet steigt der Wert des Indikators im Verlauf jeden Tages entsprechend der Uhrzeit an. Für unsere obige Aufgabe benötigen wir aber eine minutengenaue Zeitangabe. Minuten liefert uns der DatePart-Indikator mit der Einstellung „n“. Wir haben also:

DatePart(h)	Stunden
DatePart(n)	Minuten

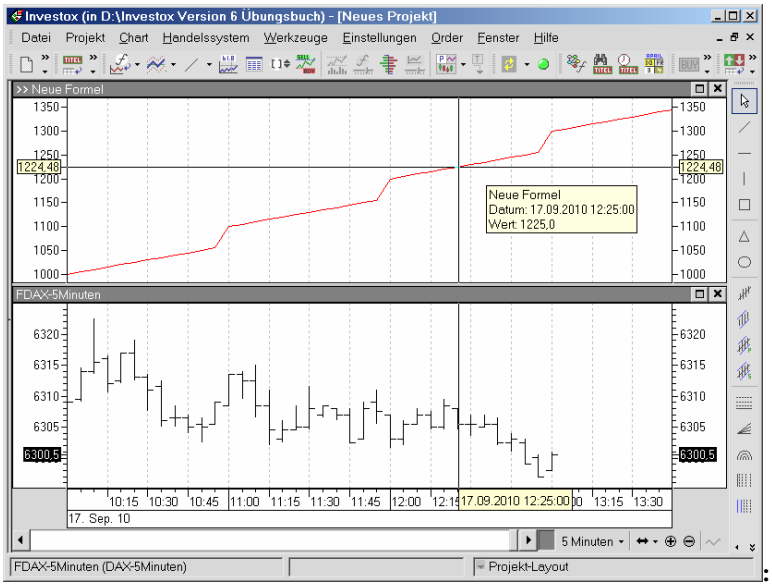
Diese beiden Werte möchten wir nun in einem einzelnen Indikator kombinieren. Dazu multiplizieren wir die Stunden mit 100, so dass die Werte 0-90 für die Minuten übrig bleiben (wobei wir nur die Werte 0-59 benötigen). Das Ergebnis ist ähnlich wie eine digitale Uhrzeitanzeige, allerdings ohne Stundentrennzeichen:

Uhrzeit	Umrechnung	Ergibt Uhrzeitwert
0:00	0 * 100 + 0	0
9:30	9 * 100 + 30	930
14:12	14 * 100 + 12	1412
21:43	21 * 100 + 43	2143
23:59	23 * 100 + 59	2359

Ändern Sie die Formel also in:

```
DatePart(h) * 100 + DatePart(n)
```

Im Chart sieht dies dann so aus (wir zoomen auf die letzten drei Stunden):



Der Wert des Indikators steigt also auch innerhalb der Stunden gemäß der Minuten an. Nun können wir unseren Wunsch von oben, die Begrenzung auf zwei Zeitfenster, umsetzen. Die Formel hierfür sieht wie folgt aus:

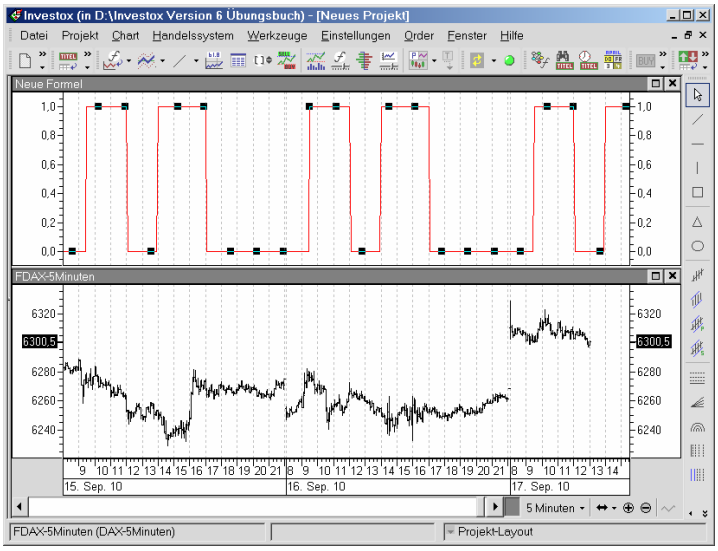
```
global calc Uhrzeit: DatePart(h) * 100 + DatePart(n);
Uhrzeit >= 930 AND Uhrzeit <= 1200 OR Uhrzeit >= 1400 AND Uhrzeit <= 1700
```

Aufgrund der Priorität der Operatoren (siehe Kapitel „Rangfolge der Operatoren“, Seite 15) benötigen wir keine Klammern.

Eine alternative Schreibeweise, die vielleicht sogar übersichtlicher ist, verwendet den Zwischen-Indikator:

```
global calc Uhrzeit: DatePart(h) * 100 + DatePart(n);
Zwischen(Uhrzeit, 930, 1200) OR Zwischen(Uhrzeit, 1400, 1700)
```

Das Ergebnis im Chart ist in beiden Fällen wie folgt (Zoomausschnitt über drei Tage):



In den gewünschten Zeiträumen ergibt die Berechnung den Wert 1 (Wahr), ausserhalb der Zeiträume aber 0 (Falsch). Diese Berechnung lässt sich dann als Zusatzbedingung in Handelsregeln einsetzen.

Periodische Datums- oder Zeiteinheiten

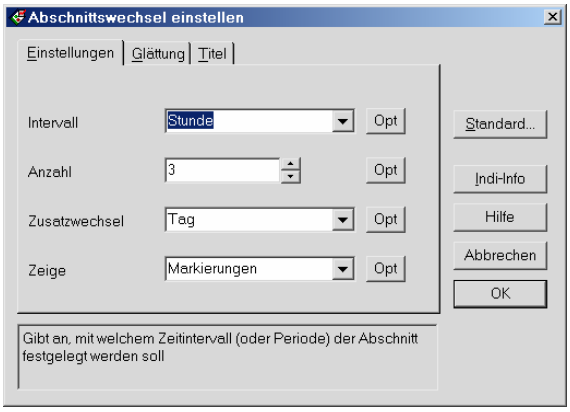
Indikator **Abschnitt**

Wenn es nicht um einen einzelnen Zeitstempel und auch nicht um einen Datumsbereich geht, sondern um einen periodischen Datums- oder Zeitwechsel, ist der **Abschnitt-Indikator** das richtige Mittel.

Und schon geht es los: Öffnen Sie einen Chart auf 5-Minuten-Basis und fügen Sie in einen leeren Teilchart oben die folgende Formel ein:

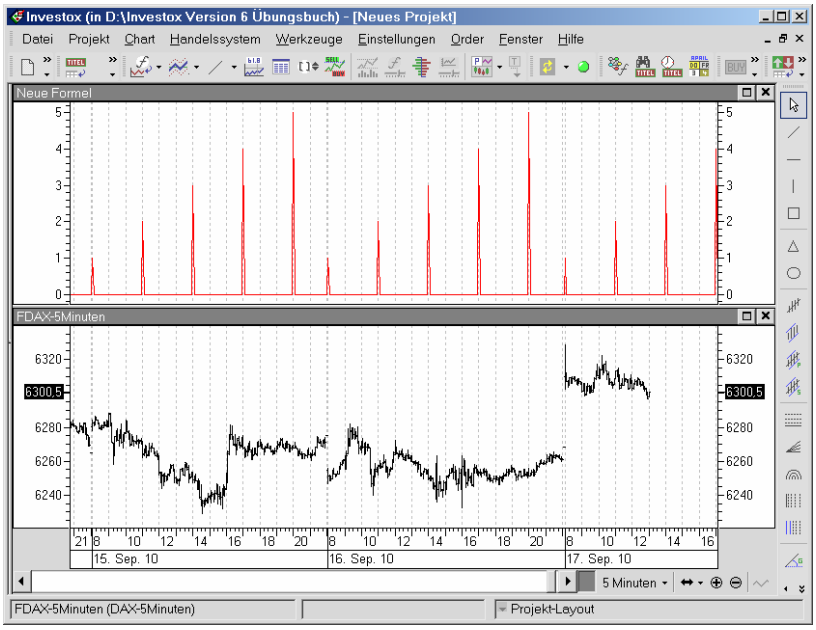
```
Abschnitt(h, 3, y, m)
```

Bevor Sie den Formeleditor verlassen: Doppelklicken Sie einmal auf den gelben Indikatornamen „Abschnitt“. Es öffnet sich die Indikator-Einstellbox und Sie sehen die Einstellungen im Klartext:



Es soll also eine Markierung nach jeweils drei Stunden erfolgen, wobei die Zählung auf jeden Fall an jedem Tag neu beginnt (Einstellung „Zusatzwechsel“). Wir möchten zudem „Markierungen“ und nicht „Levels“.

Eingefügt im Chart sieht unsere Datums-Markierung wie folgt aus (zoomen Sie auf die letzten drei Tage, um die Markierungen gut sehen zu können):



Wie gewünscht, wird jeder neue 3-Stunden-Abschnitt (gezählt vom Tagesanfang) mit einem Wert<>0 markiert. Neben der reinen Markierung (<> 0) werden die Abschnitte bis zum Tageswechsel auch noch durchnummeriert: Der erste Abschnitt hat die Markierung „1“, der zweite die Markierung „2“ usw. Die Nummerierung startet immer ab dem „Zusatzwechsel“ neu. Wird kein Zusatzwechsel verwendet, läuft die Nummerierung durch die gesamte geladene Datenreihe.

Die Datums-Markierung einsetzen: Pivots auf 3-Stunden-Ebene

Wir haben damit eine flexible Möglichkeit, periodische Abschnitte zu markieren. Einsetzen können wir dies zum Beispiel für Berechnungen, in denen die Höchst- und Tiefst-Kurse von Zeitabschnitten benötigt werden - wie den Pivot-Punkten.

Die Standard-Pivots beziehen sich, wie wir im Kapitel „Pivots“ (Seite 43) gesehen haben, auf die Höchst- und Tiefstkurse des Vortages. Wir können diese natürlich auch einmal individueller gestalten, indem wir zum Beispiel statt der Entwicklung des Vortages jeweils die Entwicklung der vorgehenden drei Stunden betrachten.

Unsere Definition des Pivot-Punktes sah ja wie folgt aus:

```
global calc H: LastDP(High);
global calc L: LastDP(Low);
global calc C: LastDP(Close);
global calc PP: (H + L + C) / 3;
```

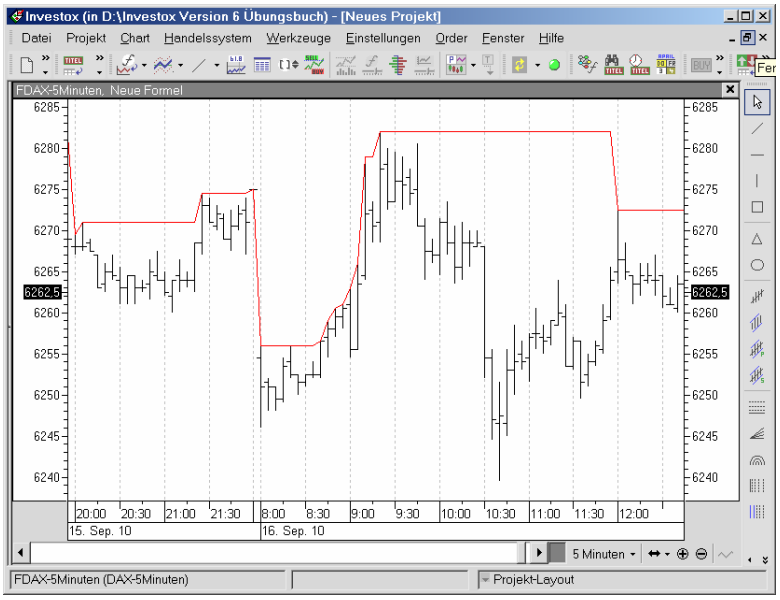
Wir müssen nun also die Definitionen von H, L und C verändern, so dass jeweils der Hoch-, Tief- bzw. Schlußkurs eines 3-Stunden-Abschnittes geliefert wird. Die Definition des Zeitabschnitts können wir ja von oben verwenden:

```
global calc Zeitabschnitt: Abschnitt(h, 3, y, m);
```

Wie bekommen wir nun den Höchstkurs pro Abschnitt? Hierzu gibt es die Analysefunktion HighestSince(Daten, Bedingung, x). Diese liefert den höchsten Wert der „Daten“ seitdem die „Bedingung“ zum x-ten Mal zutraf. Wir schreiben nun:

```
global calc Zeitabschnitt: Abschnitt(h, 3, y, m);
HighestSince(High, Zeitabschnitt, 1)
```

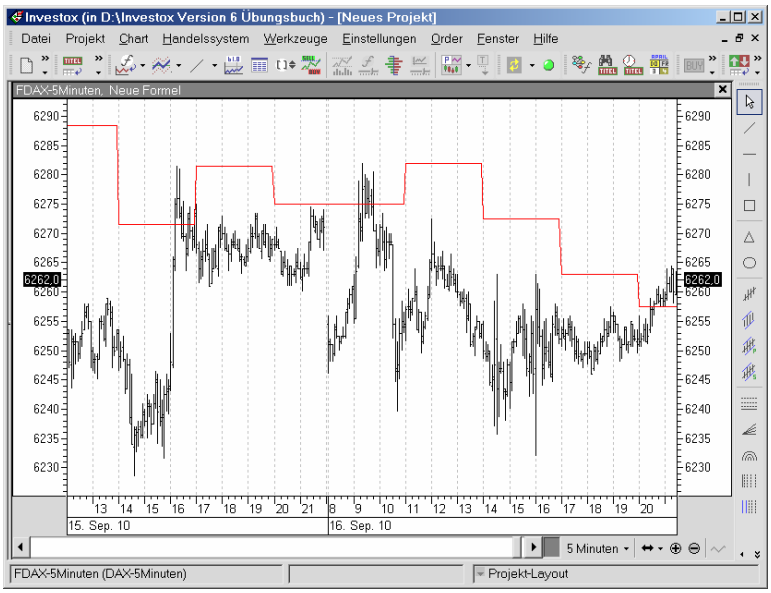
und erhalten den Höchst-Kurs seit Beginn des letzten Abschnitts. Wenn wir diese Formel in den Chart einfügen (nun in den Teilchart der Basis, da wir ja mit den Kursen selbst vergleichen), sieht dies schon gar nicht schlecht aus:



Allerdings sieht man, dass die High-Kurse innerhalb des Abschnittes noch nachgezogen werden und erst am Ende des Abschnittes fest stehen. Wir müssen nun also noch den Endpunkt an jedem Abschnitt festhalten, also den Wert der Höchstkurs-Berechnung in der Periode vor dem neuen Abschnitt. Hierzu verwenden wir die uns schon bekannte ValueWhen-Funktion:

```
global calc Zeitabschnitt: Abschnitt(h, 3, y, m);
global calc HochDesAbschnitts: HighestSince(High, Zeitabschnitt, 1);
ValueWhen(Ref(HochDesAbschnitts,-1), Zeitabschnitt, 1, V)
```

Wir erhalten nun jeweils zu Beginn eines Abschnitts den „HochDesAbschnitts“-Wert der Vorperiode (wegen Ref -1). Im Chart sieht dies jetzt sehr gut aus:



Nun benötigen wir für unseren Pivot-Punkt nur noch den Tiefst- und den Schlußkurs. Der Tiefstkurs berechnet sich analog zum Höchstkurs, indem wir nun statt HighestSince einfach LowestSince und Low-Kurse verwenden:

```
global calc Zeitabschnitt: Abschnitt(h, 3, y, m);
global calc TiefDesAbschnitts: LowestSince(Low, Zeitabschnitt, 1);
```

Den Schlußkurs des Abschnitts erhalten wir direkt mit ValueWhen. Unsere gesamte Pivot-Punkt-Berechnung mit individuellen Abschnitten sieht jetzt so aus:

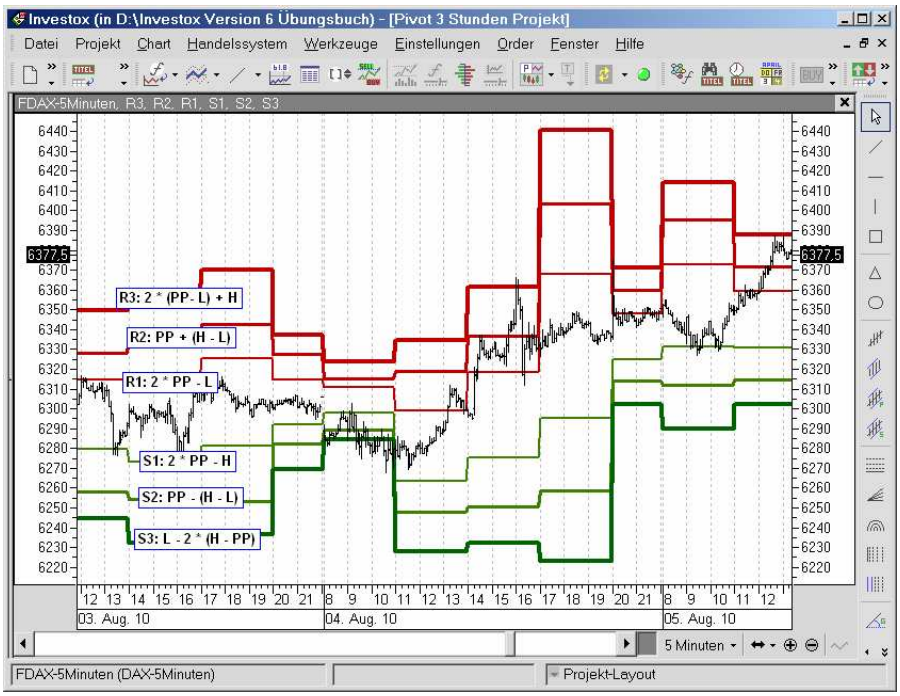
```
global calc Zeitabschnitt: Abschnitt(h, 3, y, m);
global calc HochDesAbschnitts: HighestSince(High, Zeitabschnitt, 1);
global calc TiefDesAbschnitts: LowestSince(Low, Zeitabschnitt, 1);
global calc H: ValueWhen(Ref(HochDesAbschnitts, -1), Zeitabschnitt, 1, V);
global calc L: ValueWhen(Ref(TiefDesAbschnitts, -1), Zeitabschnitt, 1, V);
global calc C: ValueWhen(Ref(Close,-1), Zeitabschnitt, 1, V);
global calc PP: (H + L + C) / 3;
```

Um andere Zeitabschnitte zu untersuchen, muss lediglich der Abschnitts-Indikator anders eingestellt werden.

Die einzelnen Pivot-Linien erhalten wir unter Verwendung von H, L und PP durch folgende Berechnungen:

```
global calc R1: 2 * PP – L;
global calc R2: PP + (H - L);
global calc R3: 2 * (PP- L) + H;
global calc S1: 2 * PP – H;
global calc S2: PP - (H - L);
global calc S3: L - 2 * (H - PP);
```

Fügen wir alle sechs Pivots nacheinander in den Chart ein (gleicher Ausschnitt wie im Kapitel „Pivots“), sieht das wie folgt aus:



Wie man sieht, wechseln die Pivots nun alle drei Stunden und verwenden zur Analyse jeweils die Daten der vorangehenden drei Stunden. Ob man mit solchen individuellen Einstellungen Erfolg haben kann, zeigt am besten ein Strategie-Backtest.

Den aktuellen Tag verwenden

Indikatoren **DailyPrice**, **ErsterWert**, **Abschnitt**

Es ist auch möglich, einen Indikator oder eine Berechnung im Chart nicht über die gesamte Datenreihe, sondern nur für den aktuellen Tag anzeigen zu lassen. Betrachten wir als Beispiel die Anzeige des täglichen Hochkurses im Intradaychart. Fügen Sie dazu in einen Intradaychart die folgende Formel in den Teilchart der Basis ein:

Achten Sie zur korrekten Anzeige darauf, dass die Formel auf dieselbe Achse skaliert wird wie die Basisdatenreihe

```
DailyPrice(High)
```

Der Indikator hält jeweils den aktuellen Höchststand des Tages fest. Das sieht dann wie folgt aus:

Wenn wir nun lediglich den Höchstkurs des aktuellen Tages als horizontale Linie darstellen möchten, verwenden wir dazu die Funktion „ErsterWert“, mit der wir auf einen bestimmten Einzelwert einer Datenreihe zugreifen und diesen als Datenreihe verwenden können.

```
ErsterWert(Daten, Offset)
```

Ein Offset von 25 liefert zum Beispiel nicht den ersten, sondern den 25. Wert der Datenreihe (vom Anfang betrachtet). Jetzt kommen wir zu einer Spezialeinstellung: Bei einem Offset von -1 wird der letzte (also jüngste) Wert der Datenreihe geliefert. Wir müssen hierzu nicht die Anzahl der vorhandenen Werte kennen.

Um den letzten, aktuellen Wert der Tageshöchstkurse zu erhalten schreiben wir also:

```
ErsterWert(DailyPrice(High), -1)
```

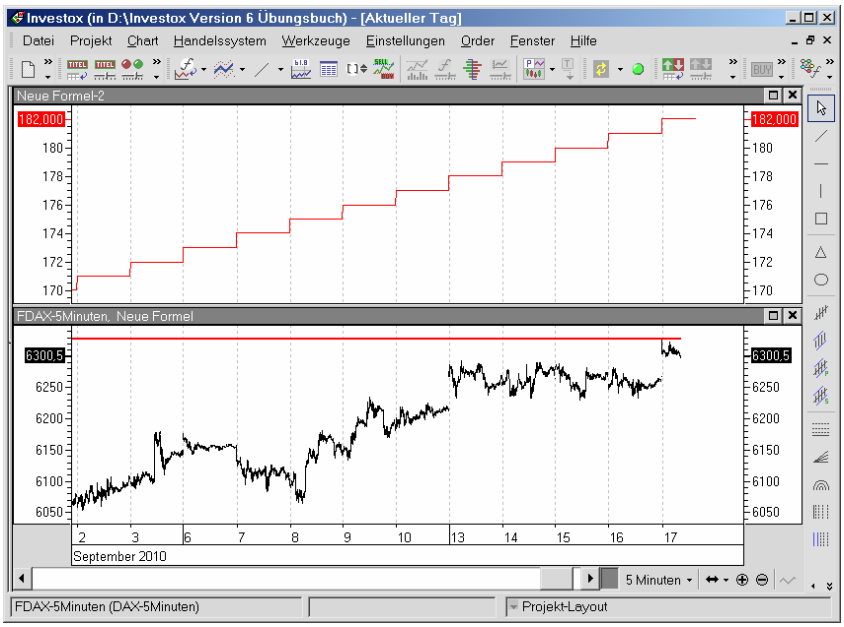
Im Chart sieht dies dann so aus:

Nun stört uns nur noch, dass die Linie nach links über den aktuellen Tag hinaus gezeichnet wird. Dies können wird ändern, indem wir als Zusatzbedingung angeben: „Verwende nur den aktuellen Tag“. Doch wie sieht die Berechnung dafür aus?

Fügen Sie als ersten Schritt dazu bitte die folgende Formel in den oberen Teilchart ein:

```
Abschnitt(y, 1, k, l)
```

Der Abschnitt-Indikator zählt, wenn er so eingestellt ist, die Tage im Chart hoch. Dies sieht so aus (etwas zurückgezoomt, damit mehrere Tage sichtbar werden):



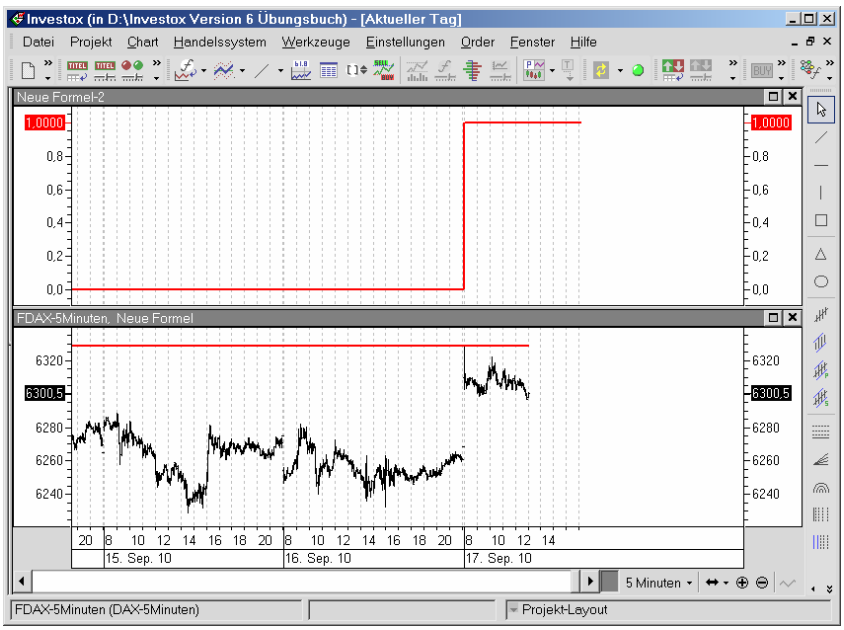
Wir sehen, dass im Chart insgesamt offenbar 182 Tage enthalten sind, da der letzte Wert des Indikators diesen Wert anzeigt. Den aktuellen Tag könnten wir also so markieren:

```
Abschnitt(y, 1, k, l) = 182
```

Diese Formulierung hat aber den Nachteil, dass sie nicht mehr passt, wenn eine andere Anzahl Tage im Chart geladen ist. Wir müssen also die Anzahl der Tage dynamisch berechnen. Hierzu verwenden wir einfach wieder den Indikator „ErsterWert“ mit dem Offset „-1“ (= letzter Wert):

```
Abschnitt(y, 1, k, l) = ErsterWert(Abschnitt(y, 1, k, l), -1)
```

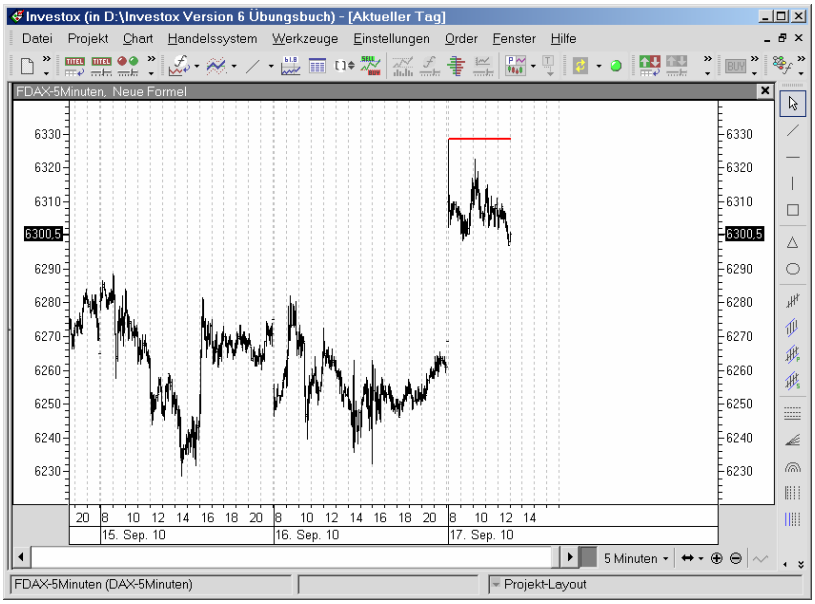
Wenn Sie die Formel im oberen Teilchart auf diese Weise ändern, erhalten Sie folgendes Bild:



Wie erwartet liefert die Formel nun für den aktuellen Tag eine 1, ansonsten 0. Damit ist der aktuelle Tag markiert und wir können diese Formel nun für unseren Tages-Höchstkurs verwenden:

```
calc AktuellerTag: Abschnitt(y, 1, k, l) = ErsterWert(Abschnitt(y, 1, k, l),-1);
ValueWhen(ErsterWert(DailyPrice(high),-1), AktuellerTag, 1,V)
```

Wenn wir diese Formel im Teilchart der Basis darstellen, erhalten wir wie gewünscht:



Eine solche Darstellung können wir zum Beispiel auch gut für die bereits besprochenen Pivots verwenden, wenn wir diese nur für den aktuellen Tag anzeigen lassen möchten.

Daten eines Zeitabschnitts verwenden (**NEU**)

Indikatoren **DatePart**, **Abschnitt, ValueWhen, HighestSince, LowestSince**

Die bereits kennen gelernten Indikatoren können wir auch dazu einsetzen, um Daten einer bestimmten Zeitspanne zu analysieren. Für manche Intraday-Strategien kann es zum Beispiel sinnvoll sein zu ermitteln, mit welcher Stärke sich die Kurse am Anfang des Tages bewegen, also welcher Abstand dort zwischen Höchst- und Tiefstkurs erreicht wird.

Im folgenden Beispiel erhalten wir als Ergebnis die Kursspanne High - Low, die zwischen dem Tages-Open und 9:30 entstanden ist:

```
global calc TagesBeginn: Abschnitt(y, 1, k, m);
global calc Uhrzeit: DatePart(h)*100+DatePart(n);
global calc Uhrzeit930: Ref(Uhrzeit,-1)<930 and Uhrzeit>=930;
global calc HighAbOpen: HighestSince(High, Tagesbeginn, 1);
global calc High930: ValueWhen(HighAbOpen, Uhrzeit930, 1, v);
global calc LowAbOpen: LowestSince(Low, Tagesbeginn, 1);
global calc Low930: ValueWhen(LowAbOpen, Uhrzeit930, 1, v);
global calc OpenSpanne: High930-Low930;
OpenSpanne
```

Wir berechnen dabei (im Intraday-Chart) den fortlaufenden Höchstkurs („HighAbOpen“) und den fortlaufenden Tiefstkurs („LowAbOpen“) des Tages und halten diese zum Zeitpunkt „Uhrzeit930“ als „High930“ bzw. „Low930“ fest. Die Kursspanne ergibt sich dann einfach aus der Differenz „High930-Low930“.

Beachten Sie, dass wir zur Berechnung von „Uhrzeit930“ nicht einfach schreiben: „Uhrzeit = 930“, sondern mit „Ref(Uhrzeit,-1)<930 and Uhrzeit>=930“ nur festlegen, dass die aktuelle Periode 9:30 oder später ist und die vorige Periode vor 9:30 lag. So funktioniert die Berechnung auch dann, wenn keine Periode mit genau der Uhrzeit 9:30 im Chart vorkommt.

In einem 5-Minuten-Chart sieht die Berechnung so aus (rote Levels im oberen Teilchart):

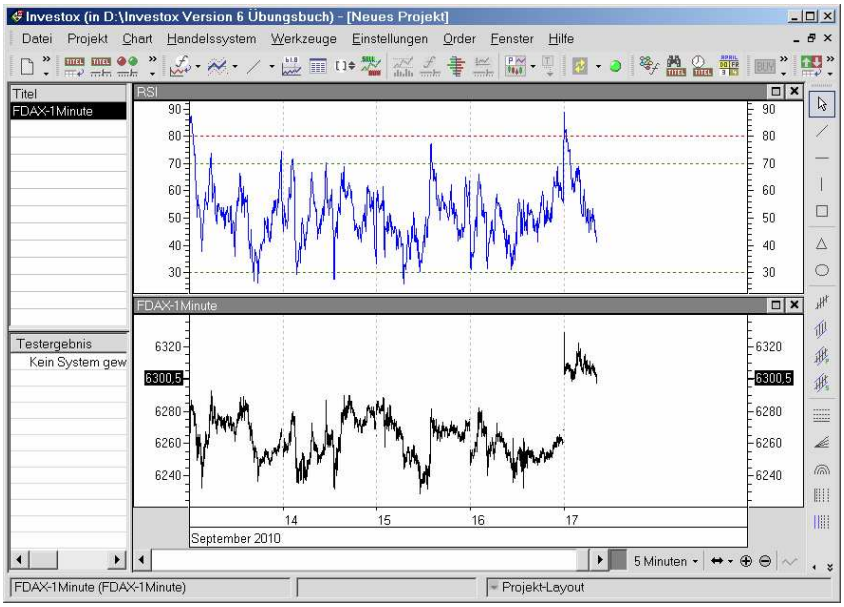
Unterschiedliche Zeitebenen kombinieren

Indikator **Komp**

Insbesondere in Intraday-Systemen kann es hilfreich sein, auch die Analyse von mehr oder weniger längerfristigen Trends einzusetzen. Dazu stellt sich dann die Frage, welchen Wert ein Indikator oder ein Kursfeld in einer anderen Zeitebene gerade hat. So kann es auch für kurzfristige Systeme interessant sein, ob sich der Kurs im langfristigen Trend gerade über der 200-Tage-Linie befindet oder nicht, oder ob ein neues 14-Tage-Hoch erreicht wurde. Oder man möchte verschiedene Gleitende Durchschnitte auf 5-, 15-, 30- und 60-Minuten-Ebene kombinieren.

Eine solche Analyse auf unterschiedlichen Zeitebenen ist mit Investox recht leicht möglich mit Hilfe des Komp-Indikators.

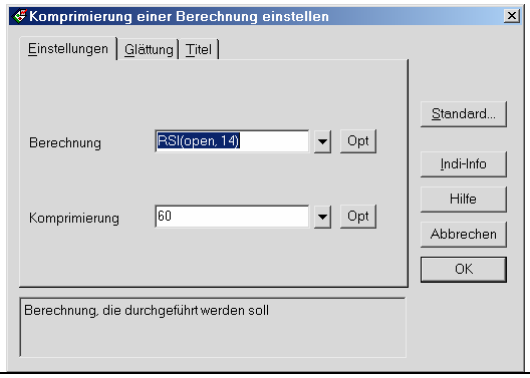
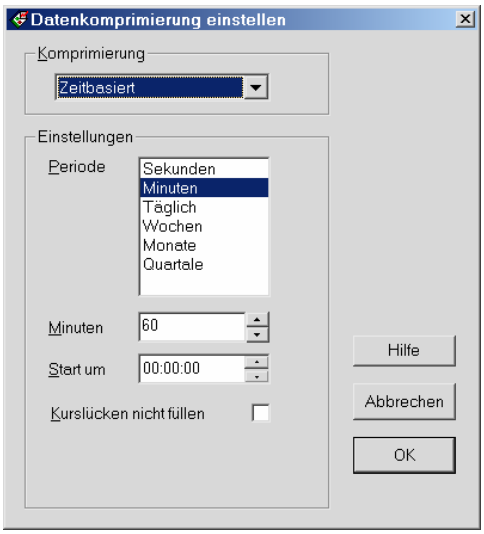
Als Beispiel öffnen wir ein Projekt mit dem mitgelieferten 1-Minuten-FDAX, wählen eine 5-Minuten-Komprimierung und zoomen auf die letzten fünf Tage. Schließlich fügen wir aus den Chartstudien die „RSI“-Studie mit 14 Perioden ein. Der Chart sieht dann ungefähr so aus:



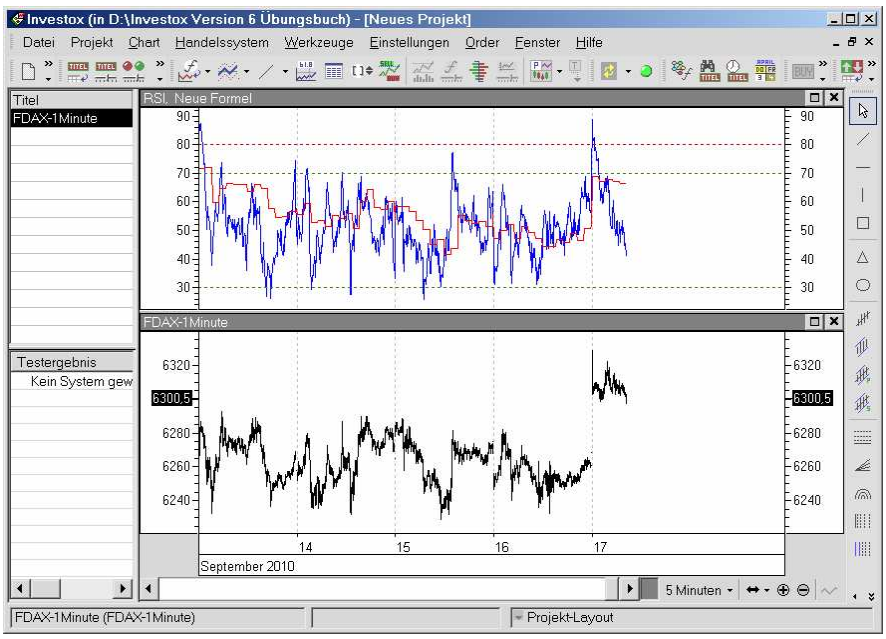
In dieser Form wird der RSI ja mit der Daten-Komprimierung der Basis, also mit 5-Minuten-Daten berechnet. Nun möchten wir den 14er-RSI zusätzlich auf Stundenbasis im Chart anzeigen lassen. Dazu fügen wir eine Formel in den Chart ein, und zwar in den Teilchart des bereits vorhandenen RSI:

```
Komp(#RSI(Open, 14)#, #60#)
```

Gerade am Anfang stellen Sie den Indikator am besten über die Indikator-Einstellbox ein. Wir gehen dazu wie folgt vor:

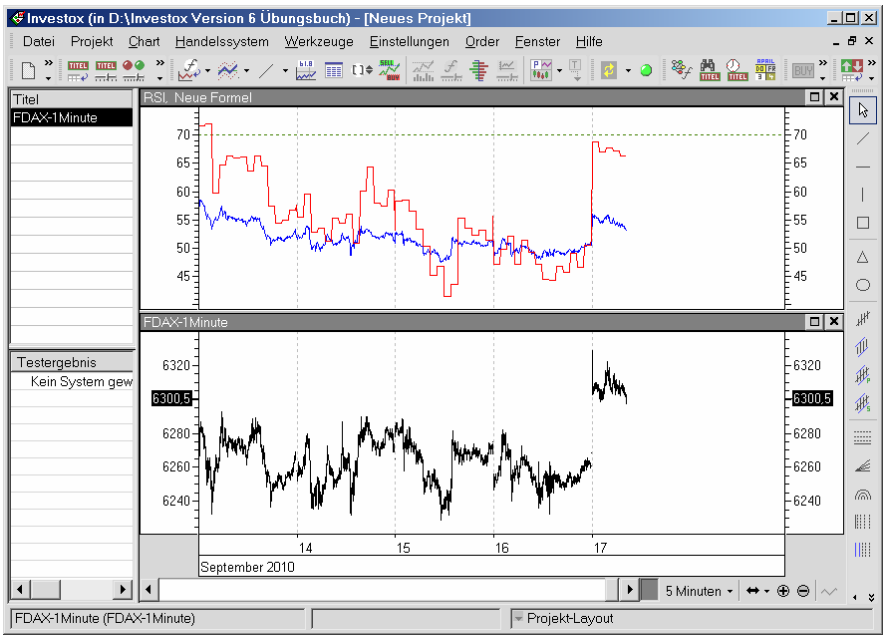
Klicken Sie im leeren Formeleditor das Werkzeug „Indikator zufügen“ und wählen Sie den Indikator „Komprimierung einer Berechnung“ (Kategorie „Analyse“). Klicken Sie **OK**.	
Es öffnet sich die Indikator-Einstellbox. Geben Sie dort unter „Berechnung“ ein: RSI(Open, 14)	
Klicken Sie dann auf die Dropdown-Schaltfläche des Parameters „Komprimierung“. Es öffnet sich der Dialog „Datenkomprimierung einstellen“. Dort wählen Sie die Komprimierung „Zeitbasiert / Minuten / 60“.	
Nach OK gelangen Sie wieder in die Einstellbox und mit nochmal OK zurück in den Formeleditor.	

Wir verwenden den Open-Kurs, da dieser zu Beginn der Stunde schon feststeht (siehe dazu weiter unten). Das Ergebnis sieht dann so aus (rot ist die neue Formel):



Der RSI auf Stundenbasis ist erwartungsgemäß stärker geglättet als der RSI auf 5-Minuten-Basis.

Warum aber verwenden wir eine andere Komprimierung und erhöhen nicht einfach die Periodeneinstellung des 5-Minuten-RSI entsprechend? Da eine Stunde aus 12 Einheiten à 5 Minuten besteht, könnten wir ja einfach die Periodeneinstellung von 14 auf 14*12 = 168 erhöhen. Das tun wir nun und erhalten folgendes Bild:



Der 5-Minuten-RSI (blaue Linie) mit 168 Perioden ist nun auch glatter, jedoch wesentlich stärker gedämpft als der Stunden-RSI. Durch die starke Erhöhung der Perioden zeigt der RSI nicht mehr die typischen Extremwerte. Die Verwendung einer anderen Komprimierung kann also durchaus sinnvoll sein.

Exkurs

Bei anderen Indikatoren ist die Verwendung einer höheren Komprimierung an Stelle einer Erhöhung der Perioden dagegen nicht so relevant. Dazu gehört zum Beispiel der Stochastik, der das Verhältnis des Schlußkurses zur High-Low-Spanne des Zeitraums angibt, ohne dabei zusätzlich zu glätten. So ergibt es durchaus Sinn auf täglicher Basis statts

```
Komp(#Stoch(5, 3)#, #w#)
```

die Einstellungen des Stochastik mit dem Wert 5 zu multiplizieren und zu schreiben

```
Stoch(25, 15)
```

Der Unterschied bzw. Vorteil der Version ohne „Komp“ ist hier zudem, das der Stochastikwert täglich, nicht wöchentlich angepasst wird. Verschaffen Sie sich im Einzelfall am besten selbst ein Bild darüber, welches der richtige Weg ist, indem Sie mit den unterschiedlichen Möglichkeiten im Chart experimentieren.

Openkurse oder Ref-1 verwenden?

Im Beispiel haben wir Openkurse verwendet, damit sich der Wert der größeren Komprimierung nicht mehr ändert. Wenn wir Closekurse verwenden, müssen wir mit Ref(*Berechnung*, -1) arbeiten. Warum?

Wenn Sie zum Beispiel Daten in Tageskomprimierung in einen 5-Minuten-Chart legen, werden die High-, Low- und Closekurse der größeren Perioden „vorausblicken“. Verwenden Sie daher in solchen Zusammenhängen nicht den letzten, sondern den vorigen Wert der Berechnung, also anstatt:

```
Komp(#RSI(Close, 10)#, #T#)
```

schreiben Sie

```
Komp(#Ref(RSI(Close, 10), -1)#, #T#)
```

Wer dies nun genauer nachvollziehen möchte, lese die folgenden Abschnitte.

Hintergrund: Warum Ref(, -1) im Komp-Indikator nötig sein kann

Die folgenden Abschnitte behandeln eine eher trockene Materie und sind mühsam zu lesen, bieten dafür aber interessante Einblicke in die Arbeitsweise der Komprimierung und Synchronisierung der Daten.

Ein wichtiger Gesichtspunkt beim Einsatz des Komp-Indikators ist die Synchronisierung der Daten. Diese erfolgt über den Zeitstempel der Daten. Um dies zu verstehen, müssen wir zunächst wissen, wie genau die Datenkomprimierung arbeitet.

Grundsätzlich ist es so, dass der Zeitstempel einer Datenperiode deren Beginn festhält. Eine Periode enthält also immer alle Daten ab dem Zeitstempel bis zum letzten Tick, bevor ein neuer Zeitstempel beginnt. Die 1-Minuten-Zeitperiode von 9:28 Uhr enthält also alle Daten, die ab 9:28 eintreffen bis zum letzten Kurs vor 9:29. Neben dem Open- und dem Schlußkurs werden natürlich auch der Höchstkurs und der Tiefstkurs, sowie auch, falls vorhanden, das Volumen (als Summe) festgehalten. Dies soll das folgende Beispiel eines Kursverlaufs verdeutlichen (Datenperiode 26.6.2002 - 9:28:00):

Datum	Uhrzeit	Kurs	Volumen	1-Minuten-Komprimierung
26.06.2002	09:27:56	4015	1	
26.06.2002	09:27:58	**4019**	1	= Close von 9:27
26.06.2002	**09:28:01**	**4017,5**	1	= Open von 9:28
26.06.2002	09:28:05	4016,5	2	
26.06.2002	09:28:11	4018,5	1	
26.06.2002	09:28:17	4020	3	
26.06.2002	09:28:27	4023	2	
26.06.2002	09:28:31	4022,5	1	
26.06.2002	09:28:37	4023	1	
26.06.2002	09:28:39	**4023,5**	5	= High von 9:28
26.06.2002	09:28:43	4020	1	
26.06.2002	09:28:49	4018,5	1	
26.06.2002	09:28:51	4017	2	
26.06.2002	09:28:53	**4014,5**	1	= Low von 9:28
26.06.2002	09:28:55	4019	1	
26.06.2002	09:28:59	**4018,5**	1	= Close von 9:28, Volumen=23
26.06.2002	**09:29:00**	**4017,5**	2	= Open von 9:29
26.06.2002	09:29:02	4017	1	
26.06.2002	09:29:04	4019	1	

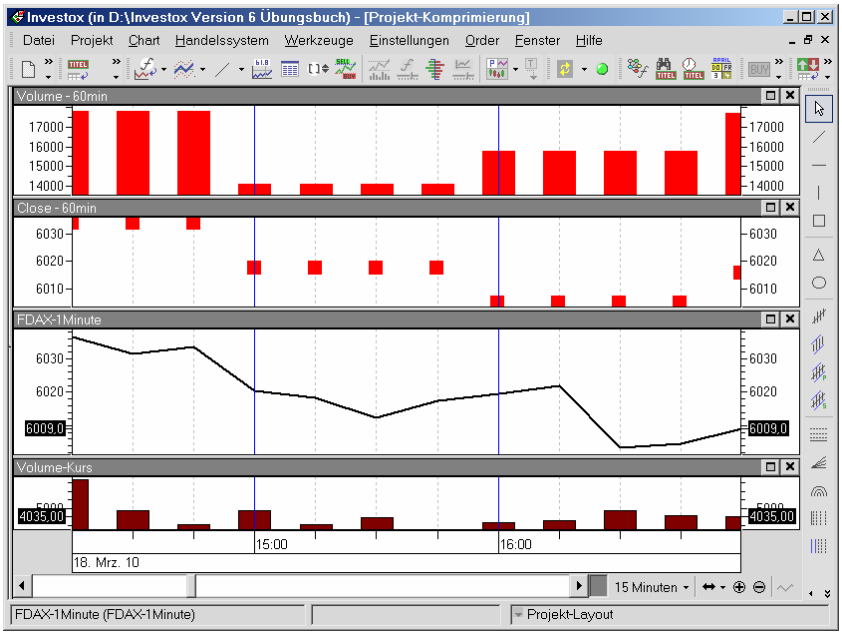
Nachdem wir nun wissen, wie die Datenkomprimierung arbeitet, können wir uns ansehen, wie zwei Datenreihen unterschiedlicher Komprimierung synchronisiert werden. Synchronisieren meint dabei, dass die Daten so „übereinander“ gelegt werden, dass die Zeitstempel der einzelnen Perioden möglichst gut zusammen passen.

Synchronisierung mit linearer Zeitachse

Unter einer linearen Zeitachse verstehen wir eine Daten-Komprimierung, bei der die Zeitabstände zwischen den Datenperioden jeweils gleich sind, die also eine feste Zeitachse verwenden. Das sind alle zeitbasierten Komprimierungen: Sekunden, Minuten, Tage, Wochen etc.

Betrachten wir die Schlußkurse einer Datenreihe in 15-Minuten-Komprimierung, über die wir jeweils in 60-min-Komprimierung die Schlußkurse und das Volumen gelegt haben, also die beiden Berechnungen:

```
Komp(#Volume#, #60#) // oberster Teilchart
Komp(#Close#, #60#) // 2. Teilchart von oben
```



Wir sehen uns einen Ausschnitt der Daten ab 15:00 Uhr in Tabellenform an (15-Minuten-Komprimierung):

	Uhrzeit	Close	Volume	Komp(#Close#, #60#)	Komp(#Volume#, #60#)
A	**15:00**	6020,5	4688	**6017,5**	**14098** (A+B+C+D)
B	15:15	6018,5	3080	6017,5	14098
C	15:30	6012,5	3899	6017,5	14098
D	15:45	**6017,5**	2431	6017,5	14098
E	**16:00**	6019,5	3343	**6004,5**	**15769** (E+F+G+H)
F	16:15	6022	3529	6004,5	15769
G	16:30	6003,5	4744	6004,5	15769
H	16:45	**6004,5**	4153	6004,5	15769
I	**17:00**	etc…	…	…	…

Man sieht, dass in der 60-Minuten-Berechnung bereits um 15:00 Uhr der Schlußkurs der 15-Minuten-Daten von 15:45 angezeigt wird, da dies gerade auch der Schlußkurs der 60-Minuten-Periode von 15:00 Uhr ist (selber Zeitstempel).

Die 60-Minuten-Komprimierung ist hier also mit anderen Worten vorausschauend – unbrauchbar für ein Handelssystem, das mit solchen Signalgebern unrealistisch gute Systemergebnisse liefern würde.

Im Realeinsatz wird sich dies so auswirken, dass sich die Werte der 60-Minuten-Komprimierung fortlaufend ändern. In der Zeile A wird sie zunächst den Wert 6020,5 anzeigen. Sobald die Zeile B dann fertig berechnet ist, wird sich der Wert der Zeile A in 6018,5 ändern, dann in 6012,5 und schließlich, um 16:00 Uhr dann, in den endgültigen Wert 6017,5. Diese rückwirkende Änderung von Werten einer Berechnung führt natürlich zu instabilen Signalen.

Setzen wir die 60-Minuten-Daten dagegen eine 60-Minuten-Periode zurück, sieht es wie folgt aus:

	Uhrzeit	Close	Volume	Close -60min	Volume -60min
	14:45	**6033,5**	3051	6029	9743
A	**15:00**	6020,5	4688	**6033,5**	17817
B	15:15	6018,5	3080	6033,5	17817
C	15:30	6012,5	3899	6033,5	17817
D	15:45	**6017,5**	2431	6033,5	17817
E	**16:00**	6019,5	3343	**6017,5**	**14098** (A+B+C+D)
F	16:15	6022	3529	6017,5	14098
G	16:30	6003,5	4744	6017,5	14098
H	16:45	6004,5	4153	6017,5	14098
I	**17:00**	etc…	…	…	…

Hier findet kein Zukunftsblick mehr statt, da der Komp()-Indikator stets nur die Schlußkurse der vorhergehenden, bereits abgeschlossenen Datenperiode liefert. Dieses Zurücksetzen um eine Periode innerhalb des Komp-Indikators erreichen wir mit Hilfe des Ref()-Indikators:

```
Komp(#Ref(Volume, -1)#, #60#) // oberster Teilchart
Komp(#Ref(Close, -1)#, #60#) // 2. Teilchart von oben
```

Ebenso korrekt wäre es übrigens, wenn wir statts

```
Komp(#Ref(Close, -1)#, 60)
```

schreiben:

```
Komp(#Open#, 60)
```

Der Openkurs einer Periode ändert sich nicht mehr und kann daher ohne Ref(, -1) eingesetzt werden.

Synchronisierung mit nicht-linearer Zeitachse

Bei der Verwendung von Komprimierungen mit nicht-linearer Zeitachse kann ein zusätzliches äußeres Ref(, -1) nötig sein.

Beispiel:

Betrachten wir als Beispiel eine Renko-Komprimierung mit 0,1%-Bricks. Ein Signal soll erfolgen, wenn der letzte Renko-Brick aufwärts zeigt:

```
Komp(#Spalte(SR)>0#, #Renko/0.1 %/1/A#)
```

Wird diese Berechnung sagen wir in einer 30-Minuten-Komprimierung eingesetzt, ergibt sich innerhalb der unvollendeten Perioden kein stabiles Signal, dieses kann vielmehr flattern – und das, obwohl sich die Richtung eines Renko-Bricks ja nicht nachträglich ändern kann. Der Grund dafür: Innerhalb einer 30-Minuten-Periode der Basis können mehrere Renko-Bricks unterschiedlicher Richtung auftreten. Verwendet wird aber nur der jeweils letzte – so dass sich das Signal dann eben ändert.

Das Signal kann sich also bei Verwendung unvollendeter Perioden innerhalb einer Periode ändern, auch wenn innerhalb der Komp-Berechnung nur Open-Kurse oder ein Ref(,-1) verwendet werden. Dies lässt sich durch ein äußeres Ref(,-1) korrigieren:

```
Ref(Komp(#Spalte(SR)>0#, #Renko/0.1 %/1/A#), -1)
```

Wöchentliche Komprimierungen mit Intraday-Daten

Die Verwendung von Tagesdaten im Intradaychart haben wir bereits oben besprochen. Hier ist ein Ref(, -1) in der komprimierten Berechnung erforderlich.

Dasselbe gilt im Prinzip auch für Daten auf Wochenbasis. Auch diese haben naturgemäß keine Zeitangaben im Zeitstempel, sondern lediglich Datumsangaben (und zwar den letzten Börsentag der Woche). Was dies für die Synchronisierung bedeutet, sehen wir uns in einem Beispiel an. In einen 15-Minuten-Chart des FDAX fügen wir die folgende Formel in den Teilchart der Basis ein:

```
Komp(#Close#, #W#)
```

Also eine reine wöchentliche Komprimierung ohne Zusatz. Das Ergebnis sieht wie folgt aus:

Für eine korrekte Anzeige muss die Berechnung auf dieselbe Achse skaliert werden wie die Basis.

Die Wochendaten (rote Linie) welchseln also wie erwartet jeweils zum Open des Freitags auf den neuen Wochenwert (20. 8., 27. 8. und 3. 9.), und zwar auf den Schlußkurs der Woche – der allerdings erst am Ende des Freitags feststeht (im Chart mit Kreisen markiert). Ohne Zusatz blicken die Wochendaten also jeweils ab Freitagmorgen in die Zukunft (auf den Freitagabend).

Wir können nun natürlich schreiben:

```
Komp(#Ref(Close, -1)#, #W#)
```

und sind damit auf der sicheren Seite. Allerdings erhalten wir so immer Freitags den Schlußkurs der vorigen Woche, und damit reichlich verspätet. Eigentlich möchten wir den Wochen-Schlußkurs ja bereits zum Close des Freitags erhalten. Auch dies ist möglich. Dabei kommt uns zu Hilfe, dass man Komprimierungen in Investox sogar verschachteln kann. Beachten Sie, dass wir beim Verschachteln von Komprimierungen die „erweiterte Rautensyntax“ verwenden müssen, damit Investox weiß, wo die Berechnung beginnt und endet.

Normale Rauten:
#Berechnung#
Erweiterte Rauten:
#>>Berechnung<<#

```
Komp(#>>Ref( Komp(#>>Close<<#,#W#), -1)<<#, #T#)
```

Was macht nun diese Formel? Im Inneren sehen Sie in Fettschrift unsere Wochen-Komprimierung von oben. Diese wird nun zusätzlich noch auf Tagesbasis dargestellt und zwar mit Ref(, -1). Dadurch verschieben wir die wöchentlichen Kurse um einen Tag nach rechts. Im Chart sieht das dann so aus:

Wie gewünscht, wechselt also der Wochen-Schlusskurs nun genau am Ende der Woche seinen Wert.

Eine alternative Berechnung, die ohne doppelte Komprimierung auskommt, wäre übrigens:

```
ValueWhen(Komp(#>>Close<<#, #W#), Abschnitt(y, 5, ww, m),1,v)
```

Überlegen Sie selbst, warum diese Formel ebenso funktioniert. Dabei hilft es, wenn Sie den Abschnitts-Indikator mit dieser Einstellung in einem Intraday-Chart ansehen (eigener Teilchart):

```
Abschnitt(y, 5, ww, m)
```

Der ValueWhen-Indikator liefert nämlich in der obigen Formel die Daten der wöchentlichen Kurse immer ab der letzten Periode, in der der Abschnitts-Indikator einen Wert <> 0 liefert.

Ungenaue Limits? (**NEU**)

Wenn Sie in Ihren Handelsregeln berechnete Limits oder Signallinien einsetzen, sollten Sie beachten, dass in manchen Fällen der Computer etwas anders arbeitet, als wir es erwarten.

Geben Sie bitte die folgende Formel in den Chart ein:

```
(1.2-1.1) = 0.1
```

Als Ergebnis dieser Vergleichsoperation erwarten wir natürlich den Wert „1“, also „wahr“. Im Chart werden Sie aber leider den Wert „0“ für „falsch“ sehen.

Der Grund dafür kann nur sein, dass das Ergebnis der Berechnung (1,2-1,1) nicht exakt 0,1 ergibt. In der Tat: Wenn man sich das Ergebnis von (1,2-1,1) mit sehr vielen Nachkommastellen anzeigen lässt, sieht man, dass es den etwas ungenauen Wert von 0,0999999999999999 annimmt.

Dies ist nun keine spezielle Schlampigkeit von Investox, sondern betrifft prinzipiell alle Berechnungen mit Gleitkommazahlen, die durch eine Programmiersprache durchgeführt werden und den mathematischen Coprozessor verwenden. Das Verhalten hängt laut Microsoft mit „der Norm 754 des Institute of Electrical and Electronics Engineers (IEEE) zu Gleitkommas zusammen, die verlangt, dass Zahlen im binären Format gespeichert werden“.

Es wäre nun natürlich möglich, die Berechnungen mit verbesserten arithmetischen Funktionen durchzuführen, die nicht direkt vom Coprozessor des Rechners abhängen. Dies würde allerdings die Rechenzeit deutlich erhöhen und steht daher unseres Erachtens nicht in einem sinnvollen Verhältnis zum Nutzen.

Denn zum Einen werden absolute Limits eher selten in mechanischen Handelssystemen verwendet und zum Anderen lässt sich das Problem relativ leicht beheben.

Ändern Sie die Formel oben bitte wie folgt:

```
Prec(1.2-1.1, 1) = 0.1
```

Das Ergebnis ist nun der von uns erwartete Wert „1“ für „wahr“. Erreicht haben wir dies durch eine Rundung der Subtraktion auf 1 Nachkommastelle mit der Prec( )-Funktion.

Im Fall von Limit-Berechnungen ist die Verwendung eines Offsets statt der Rundung oftmals sinnvoller. Betrachten wir hierfür ein Beispiel, wie es etwa beim Handeln des Bund Futures vorkommen könnte:

```
Low <= (Ref( LLV(Low, 20), -1) - 0.05)
```

Diese Berechnung soll (für den Bund Future) ein Signal liefern, wenn der Tiefstkurs ein Limit berührt oder unterschreitet, und zwar den tiefsten Kurs der letzten 20 Perioden abzüglich 0,05 Punkten. Aufgrund der obigen Ausführungen kann es passieren, dass das Limit touchiert und dennoch kein Signal erzeugt wird.

Dieses Problem können wir umgehen, wenn wir als Spielraum einen kleinen Offset von 0,001 auf das Limit addieren:

```
Low <= (Ref( LLV(Low, 20), -1) - 0.051)      // denn: 0.05 + 0.001 = 0.051
```

oder arbeiten Sie mit Prec( ):

```
Low <= Prec(Ref( LLV(Low, 20), -1) - 0.05, 2)
```

Die erste Version, einen kleinen Offset einzuberechnen, ist in der Regel vorzuziehen, da sie schneller berechnet wird und meistens übersichtlicher ist.

Faustregel: Wenn Sie in den Handelsregeln Kurse oder Werte auf Gleichheit (=) mit einem Vergleichswert prüfen, sollten Sie die Werte runden oder einen Offset verwenden, wenn die Kurse des Titels Nachkommawerte verwenden (also z.B. bei Devisen oder beim Bund Future).
Dies betrifft wohl gemerkt nur Ihre eigenen Berechnungen in den Handelsregeln oder im Chart. Bei allen internen Vorgängen wie bei der Berechnung von Stops, bei Überwachungslinien oder bei der kursabhängigen Komprimierung von Daten arbeitet Investox automatisch mit Offsets.

Stops (**NEU**)

Die Funktion von Stops in Investox

Unter Stops wird in Investox wie üblich ein Mechanismus verstanden, mit dem man eine bestehende Position schließen kann. Hierzu werden zum Beispiel ein Gewinnziel oder ein maximaler Verlust für die Position definiert. Die Formelsprache benötigt man hierzu nicht, da die Stops fest eingebaut sind und in den Testbedingungen einfach per Mausklick zugefügt und eingestellt werden können. Neben

kursbasierten Stops gibt es dort auch Stops in Bezug auf die Kapitalentwicklung oder die Tradedauer zur Auswahl.

Zu unterscheiden von Stops ist der regelbasierte Ausstieg aus einer Position, wie er in Investox mit einer Exit-Handelsregel formuliert werden kann, also zum Beispiel „Exit Long, wenn der RSI über 70 geht“.

Darüber hinaus gibt es aber auch Verbindungen zwischen den eingebauten Stops und dem regelbasierten Ausstieg - und das ist der Grund, warum wir das Thema „Stops“ hier im Formelbuch behandeln: Stops werden in Investox nicht nur zur Festlegung von maximalem Gewinn und Verlust, sondern immer auch dann benötigt, wenn für die Bestimmung des Ausstiegs der Einstiegskurs oder sonstige Informationen in Bezug auf die Positionseröffnung oder den Positionsablauf (z.B. wie lange die Position schon läuft) verwendet werden.

Beispiele hierfür sind:

- Setze einen Stop auf das Low vor der Einstiegsperiode.
- Steige aus, wenn die Position schon mindestens 10 Perioden läuft und dann der RSI unter 50 fällt.
- Schließe die Position, wenn das 20-Perioden-Momentum seit Eröffnen der Position um 3 Punkte gefallen ist.
- Steige aus, wenn der Close-Kurs unter seinen gleitenden Durchschnitt fällt, aber nur, wenn die Position bereits mindestens 5% Gewinn gemacht hat.

Der Grund dafür, dass solche Positionsausstiege nicht über die Exit-Regeln, sondern mit Hilfe der Stops von Investox umgesetzt werden, ist, dass die Formelsprache Zeitreihen-basiert ist (siehe erste Kapitel des Formelbuchs). Die Positionen werden erst nach Berechnung der Handelsregeln ermittelt. Die Handelsregeln können daher Informationen der Positionen wie Einstiegspreis und Dauer nicht verwenden. Die Stops dagegen können dies und haben daher in Investox eine viel wichtigere Rolle als nur das Setzen von festen Limits zum Ausstieg.

Neben den Basiseinstellungen (Limit, Gültigkeitsdauer) besitzen die Stops in Investox für diesen Zweck weitere Einstellmöglichkeiten in den Registerkarten „Zusatzbedingung“ und „Optionen“. Behandeln wir nun in ein paar Beispielen genauer, wie Sie diese Einstellmöglichkeiten verwenden können.

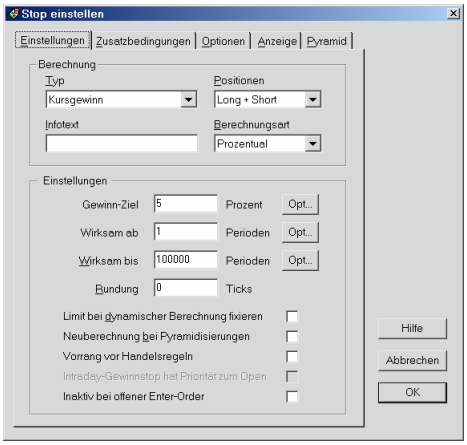
Eine Zusatzbedingung für einen Stop verwenden

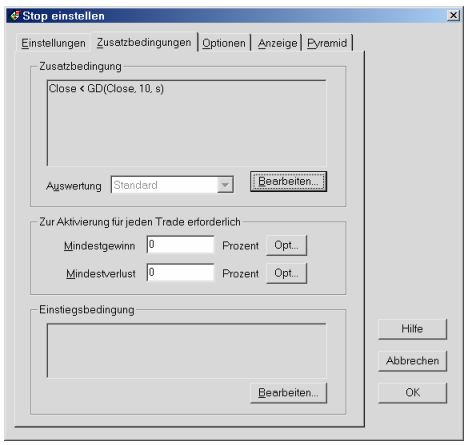
Beginnen wir mit dem zuletzt genannten Beispiel der obigen Liste:

- Steige aus, wenn der Close-Kurs unter seinen gleitenden Durchschnitt der letzten 10 Perioden fällt, aber nur, wenn die Position bereits mindestens 5% Gewinn gemacht hat.

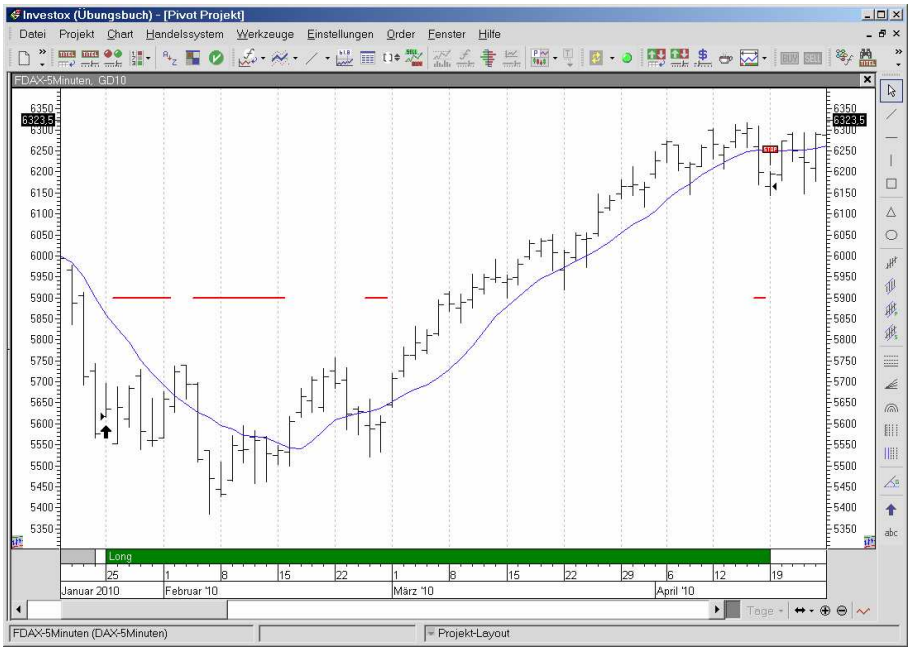
Um diesen mit einem Stop umzusetzen, müssen wir den Ansatz gedanklich umkehren: Steige bei 5% Gewinn aus, aber nur, wenn der Close-Kurs unter dem 10-Perioden-GD liegt. Entsprechend wählen wir einen Kursgewinn-Stop mit 5% Gewinnziel und geben als Zusatzbedingung an:

```
Close < GD(Close, 10, s)
```





Im folgenden Beispielchart sieht man, dass der Stop (rote horizontale Linie) nur in solchen Perioden aktiv ist, in denen der Close-Kurs unter seinem GD liegt. In diesen Perioden ist aber zunächst das Gewinnziel nicht erreicht. Erst am 18.3. treffen beide Bedingungen zu und die Position wird mit Open, Delay = 1 beendet:

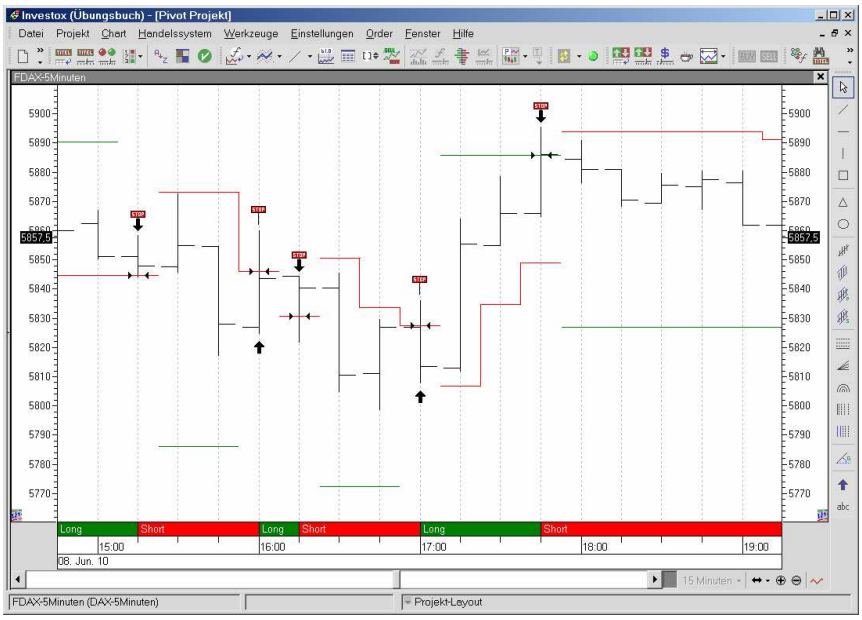


Mit Stops automatisch eine Gegenposition eröffnen

Recht einfach umsetzen lässt sich mit den Investox Stops ein Drehsystem: Gehe entweder Long oder Short, je nachdem, ob der 14-Perioden-RSI zuerst unter 30 sinkt (long) oder über 70 steigt (short). Drehe dann die Position bei jeweils 0,5% Verlust (nachgezogen) oder 1% Gewinn.

- Enter Long: RSI(Close, 14) < 30
- Enter Short: RSI(close,14) > 70
- In den Testbedingungen müssen Long+Short Positionen aktiviert sein.
- Intraday-Trailingstop Long+Short bei 0,5% mit aktivierter Option **Gegenposition eröffnen**.
- Intraday-Gewinnstop Long+Short bei 1% mit aktivierter Option **Gegenposition eröffnen**.
- Testbedingungen: Option **Kein direkter Positionswechsel** aktivieren (die Position soll nur aufgrund des Stops gedreht werden können).

Dieses System eröffnet sofort Long oder Short und wechselt danach jeweils in die Gegenposition, abhängig vom Auslösen der Stops. Im Chart zeigt sich folgendes Bild (grün = Gewinnstop, rot = Trailingstop):



Zu beachten hierbei ist, dass die Abfolge der Trades vom Start der Daten abhängt, also davon, ob aufgrund der Enter-Long- / Enter-Short-Bedingung zuerst eine Long- oder eine Shortposition eröffnet wird.

Reihenfolge der Berechnung von Stops

Hierbei stellt sich noch die Frage, welcher Stop ausgelöst wird, wenn beide Stops innerhalb einer Periode greifen: Denn Investox hat keine Möglichkeit festzustellen, welcher Stop innhalb der Periode zuerst erreicht wurde.

Es gelten daher diesbezüglich folgende Regeln:

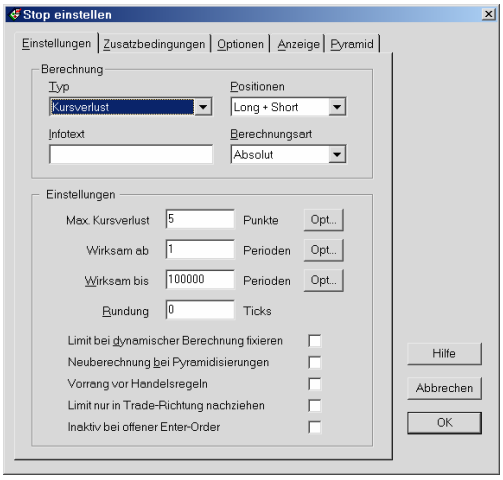
- Die Stops eines Handelssystems werden gemäß der Reihenfolge der Stops in der Liste berechnet.
- Intraday-Stops haben jedoch in jedem Fall Priorität vor allen anderen Stops.
- Für alle Intraday-Stops gilt zudem: Verluststops werden in jedem Fall vor Gewinnstops berechnet und Stops mit engerem Limit vor Stops mit weiterem Limit, unabhängig von der Reihenfolge in der Stopliste.

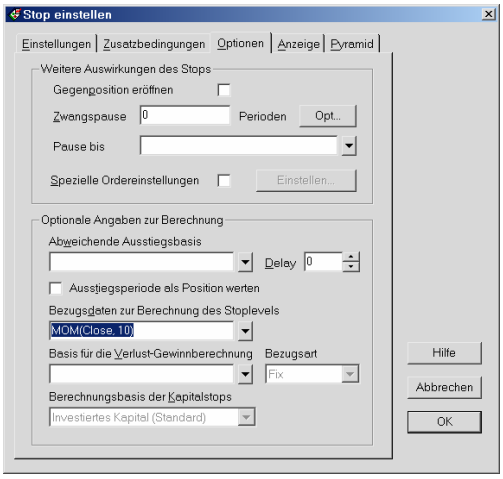
Alternative Bezugsdaten für Stops

Standardmäßig werden Stops auf die gehandelten Kurse berechnet. In Investox können Sie die Berechnung jedoch relativ einfach auch auf andere Datenreihen und Indikatoren beziehen.

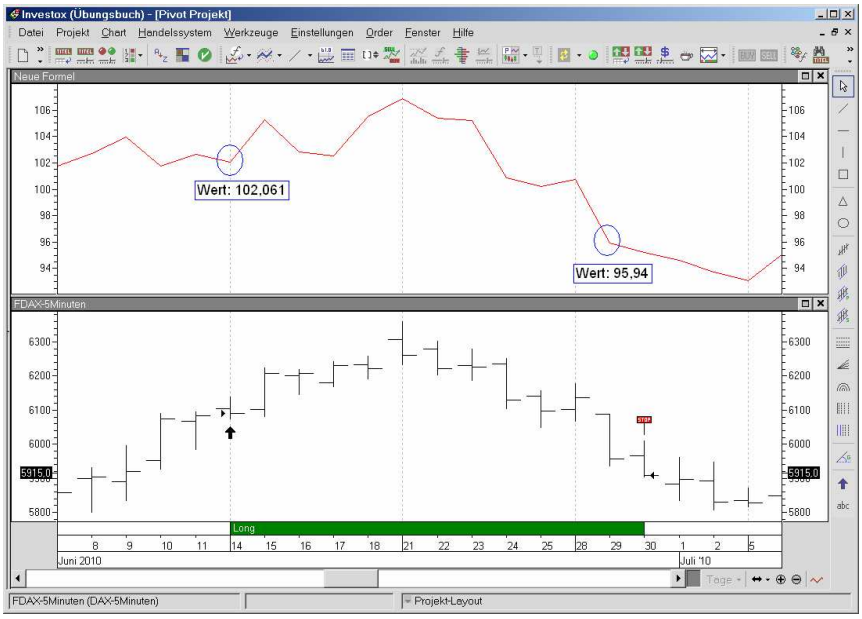
Als Beispiel: Sie möchten, dass ein Stop ausgelöst wird, sobald das 10-Perioden-Momentum um fünf Punkte gegenüber dem Wert des Momentums bei Eröffnung der Position gefallen (Long) bzw. gestiegen (Short) ist.

Wählen Sie hierfür einen Kursverlust-Stop mit fünf Punkten Maximalverlust und geben Sie in der Registerkarte „Optionen“ unter „Bezugsdaten“ den Indikator „Mom(Close, 10)" an.





Die Arbeitsweise des Stops wird mit folgendem Bild klar. Am 29. Juni fällt das Momentum zum ersten Mal fünf Punkte unter den Wert des Einstiegstags. Daher erfolgt der Ausstieg zur nächsten Periode (da in den Testbedingungen Delay = 1 eingestellt ist):



Basis für die Verlust-/Gewinnberechnung von Stops

Ein wichtiges Werkzeug zur Erweiterung der Möglichkeiten für die Berechnung von Stoplevels ist in der Registerkarte „Optionen“ der Stops die Einstellung „Basis für die Verlust-/Gewinnberechnung“. Damit lässt sich festlegen, in Bezug auf welche Daten das Verlust- bzw. Gewinnlimit berechnet werden soll. Standardmäßig – wenn bei dieser Einstellung also nichts angegeben ist - ist die Basis für die Verlust-/Gewinnberechnung natürlich der Einstiegskurs des Trades. Nun können abweichend davon aber auch andere Gewinn-/Verlustziele interessant sein - wie die folgenden Beispiele zeigen:

Beispiel: Einen Stop auf den Low-Kurs bei Positionseröffnung setzen

Es soll ein Stop auf den Low-Kurs der Periode vor dem Einstieg in die Position gesetzt werden. Der Stop soll greifen, wenn der Close-Kurs dieses Limit erreicht.

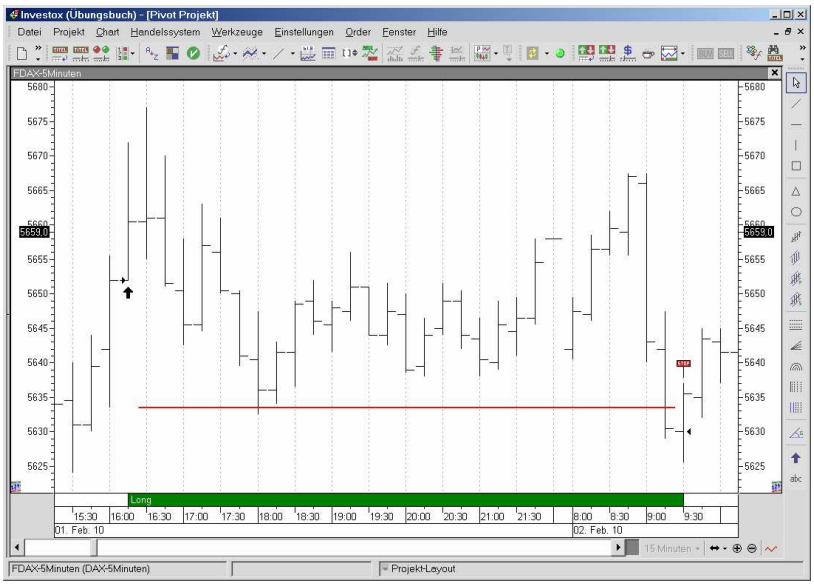
Einstellungen in der Registerkarte „Einstellungen“:

- **Stoptyp:** Kursverlust-Stop, Berechnungsart absolut
- **Maximaler Kursverlust:** 0 Punkte

Einstellungen in der Registerkarte „Optionen“:

- **Basis für Verlust-/Gewinnberechnung:** Ref(Low, -1)

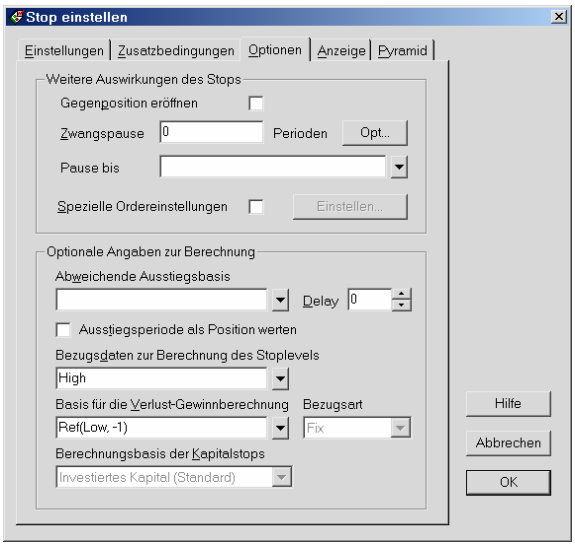
Ein Beispiel hierzu im Chart:



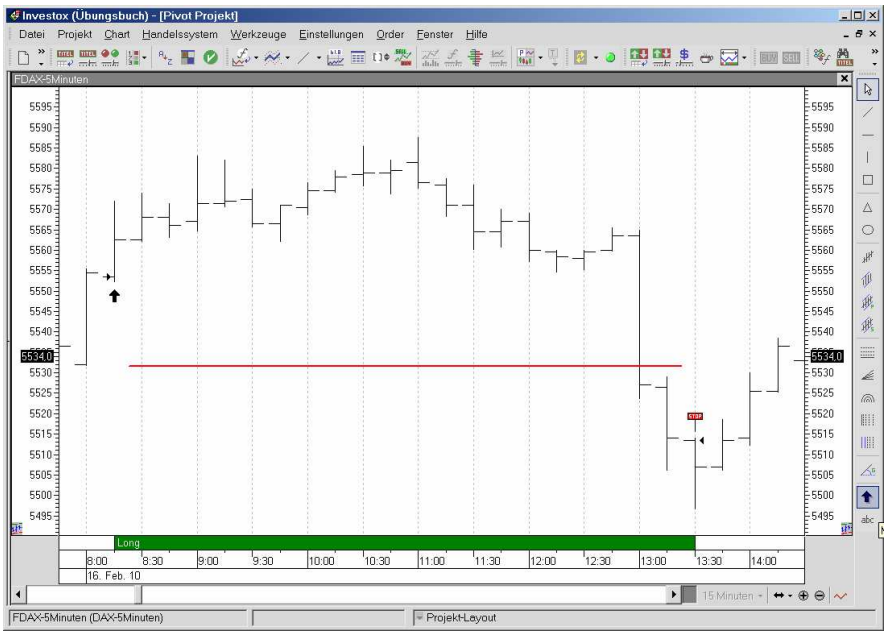
Das Stop-Level sitzt erkennbar auf dem Low der Periode vor dem Einstieg. Um 18:00 fällt bereits das Low unter das Limit, der Stop greift aber erst am nächsten Tag, wenn der Close-Kurs unter das Limit fällt. Wäre dagegen gewünscht, dass auch der Low-Kurs berücksichtigt wird, müssten wir statt des Kursverlust-Stops einen Intraday-Verluststop verwenden.

Wenn man möchte, dass der Ausstieg nur erfolgt, wenn auch das *Hoch* der aktuellen Periode unter das Stop-Limit fällt, so gibt man zusätzlich an:

- **Bezugsdaten zur Berechnung des Stoplevels:** High



Die Auswirkung davon zeigt der folgende Chart. Um 13:00 Uhr fällt der Closekurs unter das Limit. Der Stop greift aber erst eine Periode später, als auch das High unter das Limit gefallen ist:



Die Bezugsart der Basis für die Gewinn-/Verlustrechnung bei Intradaystops

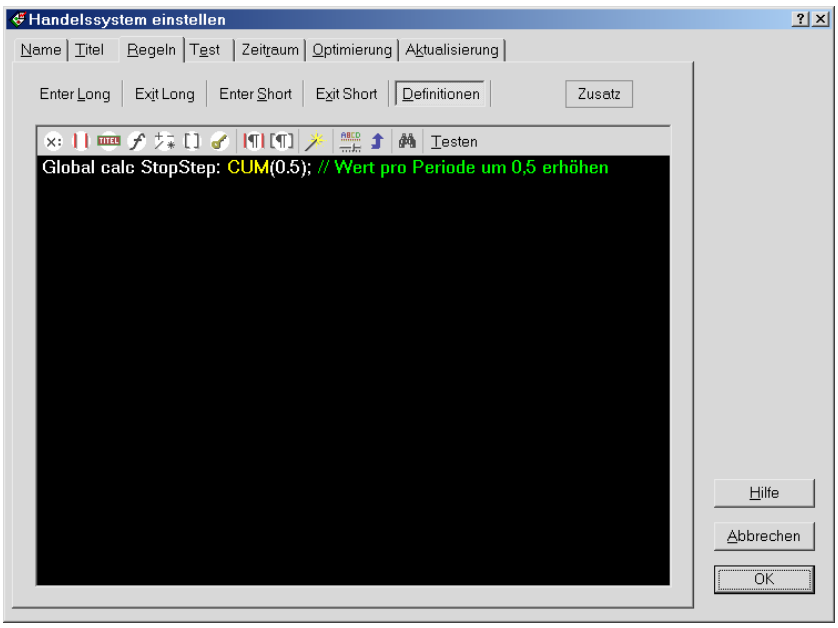
Weitere Möglichkeiten bietet die Stop-Einstellung zur Bezugsart (Dropdown-Liste rechts neben der Einstellung „Basis für die Gewinn-/Verlustberechnung“), die bei Intraday-Gewinn/Verluststops zur Verfügung steht.

Hierzu einige Beispiele zur Anwendung:

1. Beispiel zur Bezugsart (relativ%)

Der Stop soll beginnend bei 1% Abstand pro Periode konstant um 0,5% nachgezogen werden:

Unter Definitionen der Handelsregeln schreiben wir zunächst:



Der Cum-Indikator kumuliert den angegebenen Wert (hier: 0,5) pro Periode. Die global definierte Berechnung „StopStep“ können wir dann in einem Stop einsetzen. In den Testbedingungen fügen wir hierzu einen Intraday-Verluststop mit folgenden Einstellungen zu:

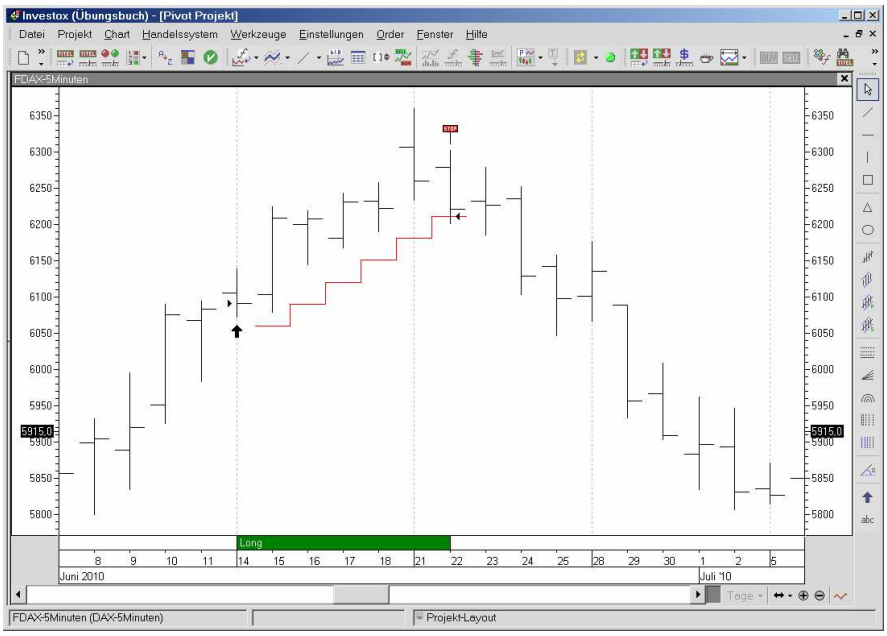
Registerkarte „Einstellungen“:

- **Verlust-Limit** auf 1% stellen.

Registerkarte „Optionen“:

- **Unter Basis für die Gewinn-/Verlustrechnung:** Die definierte Variable „StopStep“ angeben.
- **Bezugsart:** „Relativ-%“ wählen.

Dies bewirkt wie gewünscht, dass der Verluststop mit einem anfänglichen Abstand von 1% pro Periode um 0,5% des Einstiegspreises nachgezogen wird:



Hätten wir als Bezugsart „Relativ“ statt „Relativ%“ eingestellt, würde der Stop pro Periode statt um 0,5% des Einstandspreises jeweils um 0,5 Punkte absolut nachgezogen werden (was beim Dax Future kein sinnvoller Wert wäre).

2. Beispiel zur Bezugsart (Relativ)

Hier ein anderes Beispiel für die Berechnung einer relativen Verlust-Gewinn-Basis, diesmal nicht relativ%, sondern relativ absolut. Der Stop soll mit 1% Abstand vom Einstiegskurs beginnen und in jeder Periode um die halbe High-Low-Spanne nachgezogen werden.

Hierzu definieren wir zunächst in den Handelsregeln/Definitionen die Variable „StopStep“, in der die halbe Kursspanne High-Low kumuliert wird:

```
Global Calc StopStep: CUM(Ref(High-Low,-1)/2);
```

Wichtig ist es hier, jeweils die Kursspanne der Vorperiode zu verwenden (Ref-1), damit die Berechnung nicht in die Zukunft blickt. In den Testbedingungen fügen wir dann einen Intraday-Verluststop mit folgenden Einstellungen zu:

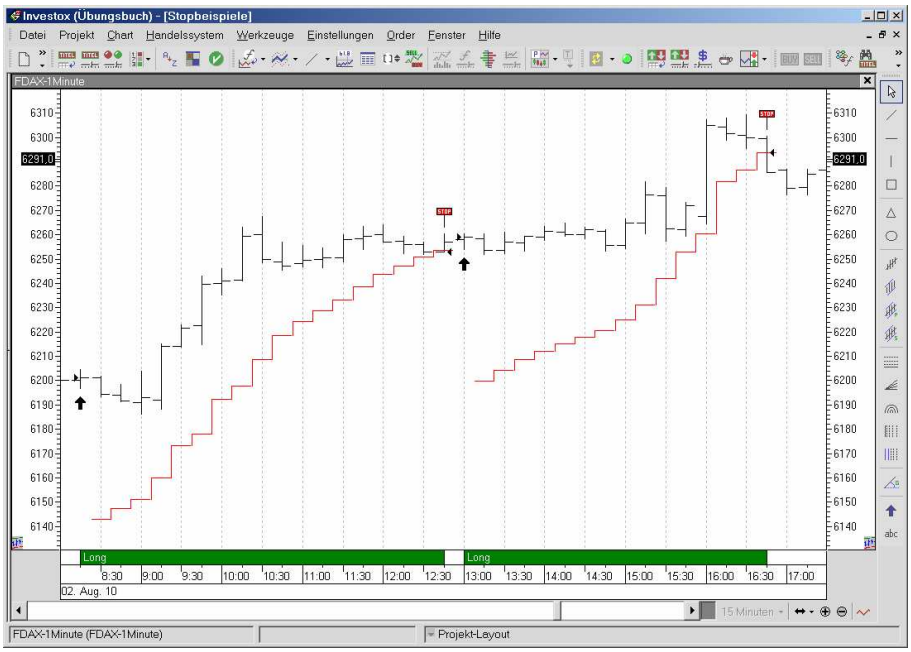
Registerkarte „Einstellungen“:

- **Verlust-Limit** auf 1% stellen.

Registerkarte „Optionen“:

- **Unter Basis für die Gewinn-/Verlustrechnung:** Die definierte Variable „StopStep“ angeben.
- **Bezugsart:** „Relativ“ wählen.

Im Chart sieht man, dass der Stop schneller nachgezogen wird, wenn die High-Low-Spanne eine höhrere Dynamik anzeigt:



3. Beispiel zu Bezugsart (dynamisch)

Dies ist ein Beispiel für die Bezugsart „Dynamisch“. Unsere Aufgabe lautet: Setze (für Long-Positionen) einen Stop auf das jeweils tiefste Tief der letzten 6 Perioden.

Das tiefste Tief über sechs Perioden erhalten wir mit:

```
LLV(Low, 6)
```

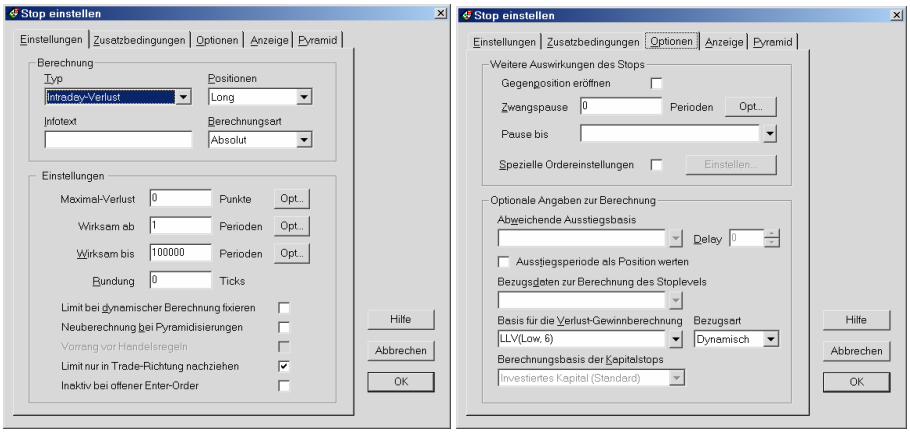
Wir verwenden dazu einen Intraday-Verluststop mit folgenden Einstellungen:

Registerkarte "Einstellungen":

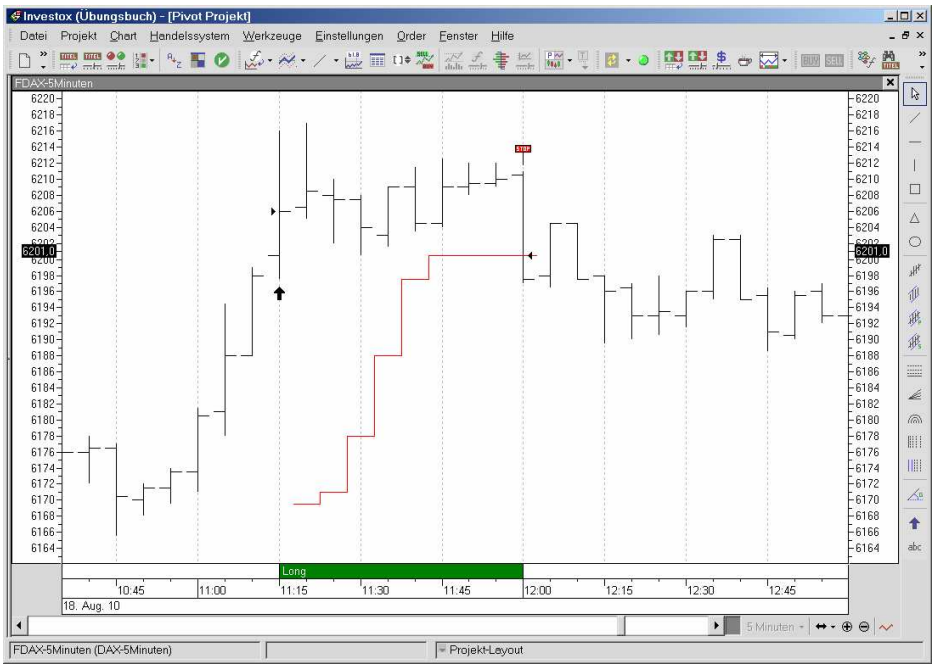
- **Positionen:** Long
- **Berechnungart:** Absolut
- **Maximalverlust**: 0
- **Limit nur in Trade-Richtung nachziehen:** Aktivieren

Registerkarte "Optionen":

- **Basis für die Gewinn-/Verlustrechnung:** LLV(Low, 6)
- **Bezugsart:** Dynamisch



Indem wir die Option „Limit nur in Trade-Richtung nachziehen“ aktivieren, erhalten wir einen richtigen Trailingstop. Hier ein Beispiel für einen Trade in einem Handelssystem auf 5-Minuten-Basis mit dieser Stopeinstellung (Anzeige des Stops in der Registerkarte „Anzeige“ aktivieren):



Das erste Stoplevel ergibt sich um 11:20 aus dem Low um 10:55 (tiefstes Tief der sechs Perioden beginnend mit der Periode um 11:20). Danach wird der Stop immer nachgezogen, wenn sich ein höheres neues 6-Perioden-Tief ergeben hat.

Der „Anwender-Stop“

Die obigen Beispiele haben gezeigt, dass man mit den erweiterten Einstellmöglichkeiten der Stops in Investox fast jede denkbare Aufgabenstellung umsetzen kann. Sollte dies doch einmal nicht möglich sein, gibt es noch den Anwender-Stop. Dieser erlaubt eine freie Programmierung der Ausstiegsbedingung. Im Unterschied zur programmierbaren Exit-Regel stehen dabei zusätzliche Schlüsselwörter zur Verfügung, welche Auskunft über den gerade laufenden Trade geben.

Aber: Der Anwender-Stop muss zu Beginn jeder Position von Investox neu berechnet werden, was sehr rechenintensiv ist. Dies wird umso stärker spürbar, je mehr Trades für ein System zu berechnen sind. Der Anwender-Stop sollte daher nur in solchen seltenen Fällen verwendet werden, in denen kein anderer Stoptyp und keine Exit-Regel oder deren Kombination die gewünschte Strategie ermöglichen.

Zudem sollte beim Einsatz einer Handelsstrategie mit Anwender-Stops im Realhandel besonders gründlich darauf geachtet werden, dass nur so viele Trades wie nötig für die aktuellen Signale berechnet werden – denn je weniger Trades zu berechnen sind, desto weniger fallen die Anwender-Stops dem Rechner zur Last (Perioden-Einstellung unter Handelssystem / Aktualisierung, Signalzeitraum im Chart anzeigen).

Hier nun zwei Beispiele zum Anwender-Stop:

Anwender-Stop Beispiel 1

Wir möchten folgende Stopstrategie umsetzen: Setze ein Anfangslimit 30 Punkte vom Einstiegspreis entfernt und ziehe das Limit bei jeweils 5 Kurspunkten Gewinn um 10 Punkte nach. Dazu verwenden wir einen Anwender-Stop und folgende Schlüsselwörter, die nur in Anwender-Stops zur Verfügung stehen:

TradePosition: Zeigt an, ob ein Long-Trade (= 1) oder ein Short-Trade (= -1) vorliegt

TradeEntryPrice: Der Einstiegspreis des Trades (gemäß Enterbasis)

TradePrice: Der aktuelle Preis (Ausstiegskurs) des Trades (gemäß Exitbasis)

TradePeriods: Die Anzahl Perioden, seit denen der Trade läuft

#_StopLevel#: Das Stoplevel, das im Chart angezeigt werden kann (Registerkarte „Anzeige“)

#_ExitLevel#: Der Ausstiegspreis des Stops (wenn nicht vorhanden, wird die Exitbasis verwendet)

Wir möchten den Stop sowohl für Long- wie für Shortpositionen einsetzen und müssen daher die Berechnung je nach „TradePosition“ differenzieren. Daher geben wir in der Registerkarte „Zusatzbedingungen“ unter „Definition“ die folgende Formel an:

```
calc Gewinn:
If(TradePosition = 1, TradePrice-TradeEntryPrice, TradeEntryPrice-TradePrice);
calc Offset:
Ref(HighestSince(INT(Gewinn/5)*10, TradePeriods=1, 1), -1);
calc #_StopLevel#:
If(TradePosition = 1, TradeEntryPrice-30+Offset, TradeEntryPrice+30-Offset);
calc #_ExitLevel#:
if(TradePosition = 1, Min(Open, #_StopLevel#), Max(Open, #_StopLevel#));
//////// Ausstieg:
If(TradePosition = 1, Low <= #_StopLevel#, High >= #_StopLevel#)
```

Die Berechnung enthält die folgenden Komponenten:

Gewinn: Es wird der Abstand vom aktuellen Preis zum Einstiegspreis berechnet (für Short umgekehrt).

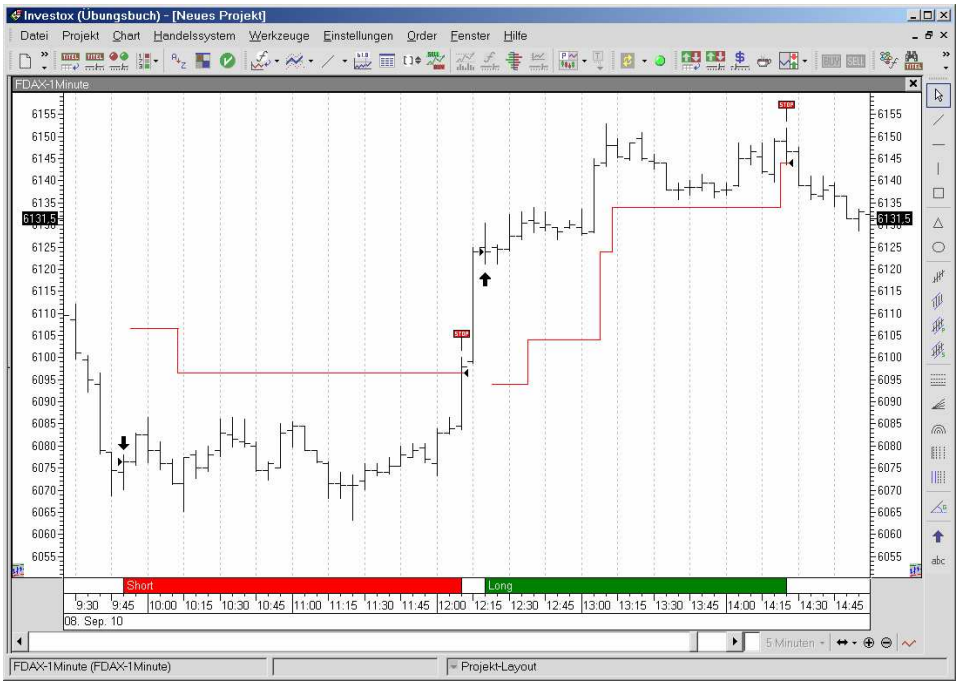
Offset: Der Gewinn wird durch 5 geteilt, abgerundet und dann mit 10 multipliziert – so ergeben sich 10-Punkte-Schritte bei jeweils 5 Kurspunkten Gewinn. Davon wird dann mit „HighestSince“ der größte Wert seit Beginn der Position festgehalten (der Stop soll nur in Richtung Kurs nachgezogen werden).

StopLevel: Hier wird das Stoplevel berechnet (Anfangsabstand 30 Punkte und aktueller Offset).

ExitLevel: Der Ausstiegspreis ist das Stoplevel oder aber „Open“, wenn bereits der Open-Kurs unter dem Stoplevel (bei Long-Positionen) bzw. über dem Stoplevel (bei Short-Positionen) eröffnet.

In der letzten Zeile, der Definition der Variablen folgend, wird dann noch die Stopbedingung selbst angegeben, also wann der Stop greifen soll (Low <= Stoplevel für Long und High>= Stoplevel für Short).

Hier ein Beispiel, wie dieser Stop im Chart angezeigt wird und arbeitet:



Anwender-Stop Beispiel 2

Ein weiterer Grund für den Einsatz von Anwender-Stops kann die individuellere Anpassung in pyramidisierenden Systemen sein. Für diese Systeme stehen nämlich spezielle Schlüsselwörter zur Verfügung.

TradePyramiden: Gibt an, wie viele Pyramiden in der laufenden Position bereits erfolgt sind.

TradeAVGEntryPrice: Liefert den durchschnittlichen Einstiegspreis der Gesamtposition.

Eine Aufgabenstellung könnte zum Beispiel sein: Aktiviere ab der zweiten Pyramide einen Verluststop fünf Punkte unter dem durchschnittlichen Einstiegspreis (für Long Positionen):

```
calc #_StopLevel#: If(TradePyramiden>1,TradeAVGEntryPrice-5,#_KeinWert#);
calc #_ExitLevel#: If(Open<#_StopLevel#, Open, #_StopLevel#);
TradePyramiden > 1 AND Low < #_StopLevel#   // Die Ausstiegsbedingung
```

Das Schlüsselwort #_KeinWert# sorgt hier dafür, dass während der ersten Pyramide noch kein Stoplevel im Chart angezeigt wird. Geht es nur um die Wirkung des Stops, könnten wir statt #_KeinWert# auch 0 schreiben – dann würde aber in der ersten Pyramide im Chart eine 0 als Stoplevel angezeigt, wodurch die Chartdarstellung stark verzerrt würde.

Wie es weitergeht

Sie haben nun einige Übungen zur Formelsprache von Investox absolviert. Mit dieser Grundlage können Sie auch die mitgelieferten Beispiele in Projekten und Einflussfaktoren sicherlich nachvollziehen. Wie geht es dann weiter? Am besten durch „learning by doing“: Versuchen Sie zunächst einmal, eine einfache Strategie umzusetzen. Wenn es nicht weitergeht, fragen Sie im Investox-Forum nach, wo Sie in der Regel bald eine Antwort erhalten!

Wir wünschen Ihnen viel Erfolg mit Investox!

Anhang

Referenz der Kürzel wichtiger Indikator-Einstellungen

Im folgenden finden Sie ein Verzeichnis der Kürzel der Einstellungen einiger wichtiger Indikatoren von Investox. Verwenden Sie diese Kürzel nicht als Namen für eigene Const-Definitionen, da dies zu Fehlermeldungen führt, wenn Sie Indikatoren einsetzen, die diese Kürzel ebenfalls als Einstellung verwenden.

Zwischen Groß-/Kleinschreibung wird bei den Kürzeln – wie auch sonst in der Formelsprache von Investox – nicht unterschieden.

Am bequemsten stellen Sie die Indikatoren natürlich über die Indikator-Einstellbox ein, da dort bei jedem Parameter alle verwendbaren Einstellungen in der Dropdown-Liste aufgelistet werden.

Preisfelder-Auswahl

Zur Auswahl eines Preisfeldes, bei vielen Indikatoren verwendet:

Kürzel	Beschreibung
C	Close
O	Open
H	High
L	Low
V	Volume
I	OpenInt

Spaltenwert-Auswahl

Welcher Wert einer Spalte (beim Spalten-Indikator, nur Renko und P&F):

Kürzel	Beschreibung
SO	Startwert
SH	Hochpunkt
ST	Tiefpunkt
SC	Schlußwert
SR	Richtung
SA	Anzahl Boxes/Bricks
SE	Eröffnungskurs
SB	Richtung bestätigt

Glättungsmethoden des GD (Gleitender Durchschnitt)

Bei vielen Indikatoren, wie GD, Preisoszillator etc.:

Kürzel	Beschreibung
S	Standard
W	Gewichtet
E	Exponentiell
VAR	Variabel

TRI	Triangular
LR	Lineare Regression
AES	Adaptive Exp. Smoothing
AMA	Adaptive Moving Average

Auswahl Punkte/Prozent

Zur Auswahl der Berechnungsart, bei vielen Indikatoren verwendet:

Kürzel	Beschreibung
$	Absolut
%	Prozent

Datums-Intervallangaben

Verwendet von DatePart, DateDiff, Datums-Markierung:

Kürzel	Beschreibung
yyyy	Jahr
q	Quartal
m	Monat
ww	Woche
y	Tag
d	Tag des Monats
w	Wochentag
h	Stunde
n	Minute
s	Sekunde

Sortierung

Sortierung der Rangfolge eines Titels:

Kürzel	Beschreibung
Auf	Aufsteigend
Ab	Absteigend

Fuzzy-Grenze

„Richtung“, gibt an, ob ein Fuzzy-Größer-Als oder ein Fuzzy-Kleiner-Als berechnet werden soll:

Kürzel	Beschreibung
G	Größer
K	Kleiner

Tickanalyse

Art der Tickberechnung des Tickanalyse-Indikators:

Kürzel	Beschreibung
Ticks	Anzahl Ticks
UpTicks	Anzahl UpTicks
DownTicks	Anzahl DownTicks
UpVolume	Volumen UpTicks
DownVolume	Volumen DownTicks

TimeZone / ValueWhen

Richtung der Berechnung:

Kürzel	Beschreibung
V	Vergangenheit
Z	Zukunft

Überwachungslinie

Art der Skalierung der Überwachungslinie:

Kürzel	Beschreibung
Lin	Linear
Log	Logarithmisch

Kursprognose

Art der Berechnung (Stärke der Kursänderung oder nur ein Signal -1/1):

Kürzel	Beschreibung
K	Kurs (Stärke der Kursänderung)
S	Signal

DepotHistorie

Welche Information geliefert werden soll („Methode“):

Kürzel	Beschreibung
S	Stückzahl
P	Orderpreis
R	Return
B	Basis bei Signal
G	Gebühren
SLI	Slippage
K	Kapitalentwicklung
KH	Kapital - Hoch
KL	Kapital - Tief
KD	Kapital - Maximaler Drawdown

Abschnittsmarkierung

Was gezeigt wird (Parameter „Zeige“):

Kürzel	Beschreibung
M	Markierungen
L	Levels

Histogramm-Analyse

Welche Kennzahl geliefert werden soll:

Kürzel	Beschreibung
MaxKurs	Maximum-Kurs
MinKurs	Minimum-Kurs
AbsMaxKurs	Absoluter Maximum-Kurs (immer positiv)
VAOKurs	VA-Obergrenze-Kurs
VAUKurs	VA-Untergrenze-Kurs
VAMittelKurs	VA-Mittelwert-Kurs
VAAnzahl	Anzahl Balken in VA

Anzahl	Anzahl Balken des Histogramms
HistoMax	Größter Wert
HistoMin	Kleinster Wert
HistoAbsMax	Größter absoluter Wert
HistoMittel	Mittel der Histo-Werte
HistoSumme	Summe der Histo-Werte
HistoProzAbw	Prozentuale mittlere Abweichung
HistoStdAbw	Standardabweichung
HistoSlope	Steigung der Histo-Werte
HistoVZWechsel	Anzahl Vorzeichenwechsel
HistoForm	Form des Histogramms (Muster-Analyse)