Functions, Indicators, and Signals

A guide to use, understand, and build your own functions,
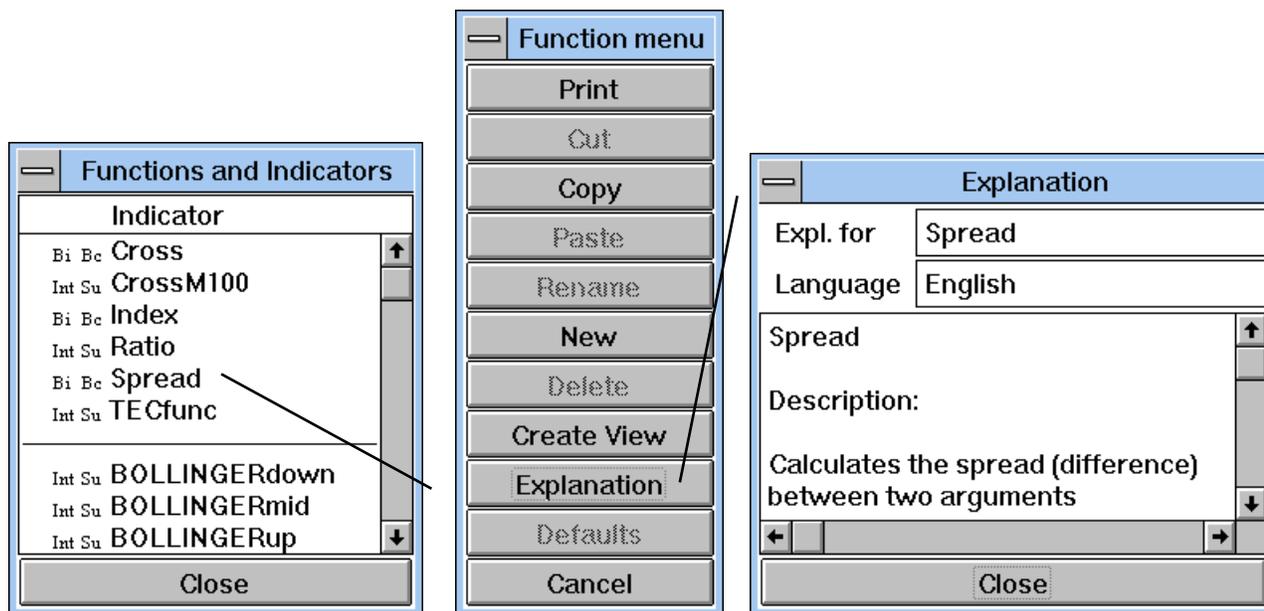indicators, and signals

1 Introduction

Functions, indicators, and signals are a vital part of BC VIEW. Functions and indicators can calculate graphs in a view and functions can further more calculate (calculated) instruments. Signals calculate when to: go long, go exit long, go short, and go exit short.

Functions, indicators, and signals can be part of BC VIEW (Build-In) as e.g. **MovAv**, **Spread**, and **Index,** or they can be made by the user as e.g. **CrossM100**, **CCI**, or **OSC**. Further more users can use the functions, indicators, and signals created by the super user. You can use the functions, indicators, and signals that you create in exact the same way as you use the build-in ones.

2 How to get an understanding of the functions, indicators, and Signals

The functions and indicators are listed in the menu entry called **Formulae** and the signals are listed in the menu entry called **Signals**. When you press **Formulae** BC VIEW opens a list of **Function and Indicators**. The list at the top shows all the functions you can call and at the bottom it shows all the indicators you can call. The functions and indicators include the build-in ones as well as the ones belonging to the super user.

You can get an explanation of a function or an indicator. You press the right mouse button on the function or indicator for which you want an explanation. BC VIEW shows a menu. To have BC VIEW show the explanation for the function or indicator select **Explanation**. You can get an explanation for a signal in the exact same way. See figure 1.

See also chapter 7-10 for lists of functions, indicators, and signals.

Figure 1. Opening explanation for a function.

3 How to edit or view own functions, indicators, or signals

You can edit functions, indicators, and signals belonging to you and you may view the build-in and the ones belonging to the super user. To edit or view functions and indicators press menu entry **Formulae** to open the list of **Functions and Indicators** - then double click the function or indicator you want to edit/view. To edit or view a signal press menu entry

Signals. The editors for functions, indicators, and signals differ slightly. The following figure shows the **Indicator Editor** editing **Volatility**:
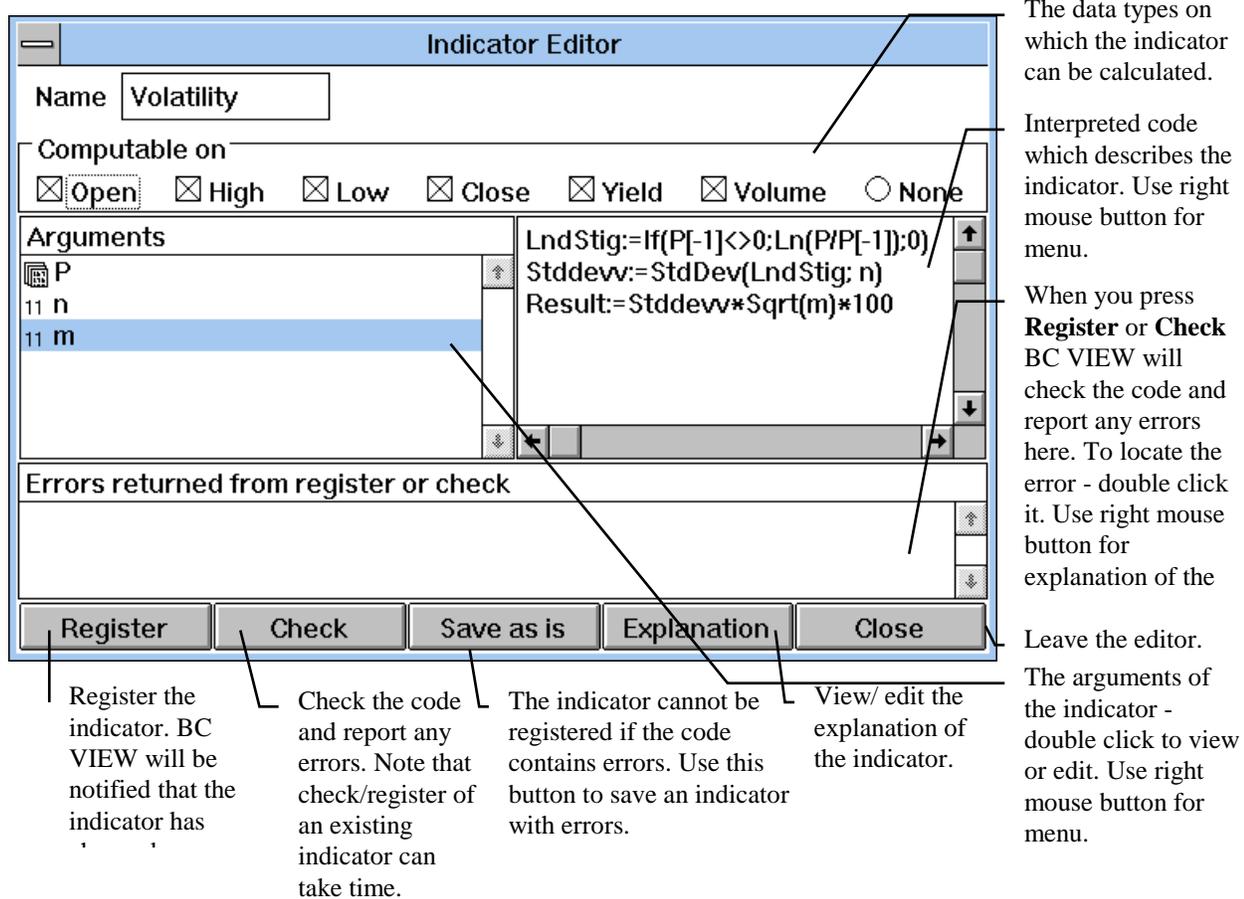

Figure 2. Indicator editor.

The callouts around the figure read:

- The data types on which the indicator can be calculated.
- Interpreted code which describes the indicator. Use right mouse button for menu.
- When you press **Register** or **Check** BC VIEW will check the code and report any errors here. To locate the error - double click it. Use right mouse button for explanation of the
- Leave the editor.
- The arguments of the indicator - double click to view or edit. Use right mouse button for menu.
- Register the indicator. BC VIEW will be notified that the indicator has
- Check the code and report any errors. Note that check/register of an existing indicator can take time.
- The indicator cannot be registered if the code contains errors. Use this button to save an indicator with errors.
- View/ edit the explanation of the indicator.

Code shown in the editor:

```
LndStig:=If(P[-1]<>0;Ln(P/P[-1]);0)
Stddevv:=StdDev(LndStig; n)
Result:=Stddevv*Sqrt(m)*100
```

4   Calculation of functions, indicators, and signals

When you see an instrument or a graph in a view the time-axis shows periods. Each period may be a number of minutes, a day, a week, a month, or a year. Within the same period the view may show more different data types (prices) namely: Open, Low, High, Close, Yield, and Volume.

BC VIEW stores instruments as Open, High, Low, Close, Yield, and Volume on a daily basis (period = day) or/and simply as prices for a given time.

Since a view may show other periods than daily BC VIEW will automatically calculate prices for other periods e.g. if the user requests to have prices displayed on a monthly basis BC VIEW will need to convert from daily prices to monthly prices.

Data types are defined (converted) as follows:

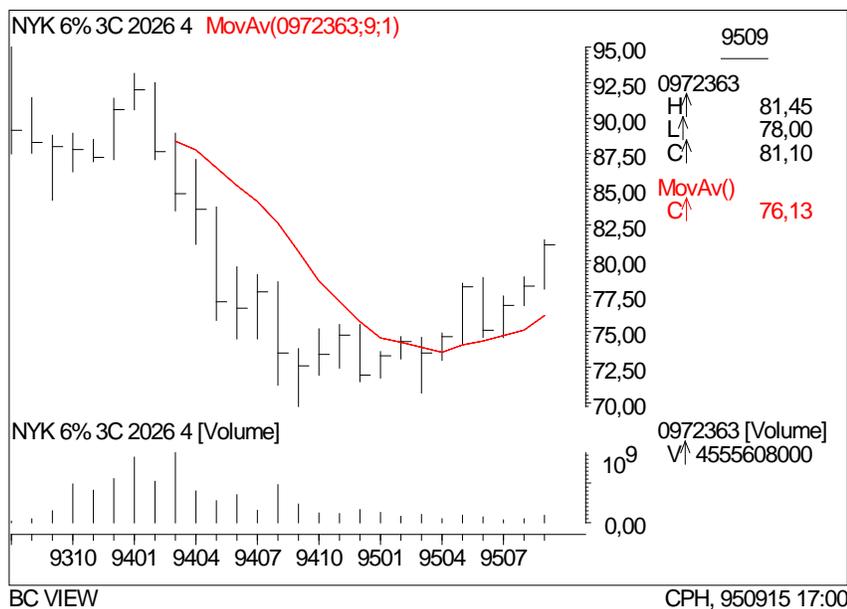|	
Open price->	Price that exists after first minute in the period
Low price->	Lowest price in the period
High price->	Highest price in the period
Close price->	Price that exists after last minute in the period
Yield->	Yield that exists after last minute in the period
Volume->	Sum of volumes in the period

Figure 3. View where the time axis shows months. The top element displays Low, High, and Close prices and a moving average calculated on the monthly close prices over the last nine months. The bottom element displays Volume.

Indicators and signals are calculated on data as we see them in a view. The data thereby must have been split in data types and in periods prior to calculation.

Indicators and signals are calculated period for period. This means that BC VIEW will start with the first period, calculate its value, move on to the next period, calculate its value and so on. E.g. the moving average of the view displayed in figure 3 first starts by calculating a value for the period 9403-9404, then calculates a value for the period 9404-9405, and so on until it finally calculates a value for 9509-9510.

Calculation of a value can be based on previous periods. Again if we look at the moving average in figure 3 it is calculated based on the current monthly close price and the previous 8 monthly close prices. Note that the moving average starts at period 9503-9504 because the first data exists for "NYK 6% 3C 2026 4" in period 9307-9308.

Let us calculate an indicator called MovAvLast2Periods. The indicator takes a set (graph or instrument) as parameter and calculates a moving average for the last 2 periods. The indicator can for instance be written as:

Result := (Instrument + Instrument[-1])/2

The indicator is calculated by adding the value of Instrument for the current period with the value of Instrument for the previous period and then dividing the result with 2.

Let us illustrate the period to period calculation by calculating MovAvLast2Periods on the close prices of an instrument (NYK 6% 3C 2026 4):

Period	High	Low	Close	Yield	Volume	MovAvLast2Periods-Close	HighLowAv
950901	78.60	78.00	78.35	8.75	53945000		(78.60+78.00)= 78.30
950904	78.80	78.20	78.55	8.72	138692000	(78.55+78.35)/2= 78.45	(78.80+78.20)= 78.50
950905	78.95	78.50	78.70	8.70	103105000	(78.70+78.55)/2= 78.625	(78.95+78.50)= 78.725
950906	79.30	78.75	79.10	8.64	711189000	(79.10+78.70)/2= 78.90	(79.30+78.75)= 79.025
950907	79.30	78.80	79.05	8.65	369596000	(79.05+79.10)/2= 79.075	(79.30+78.80)= 79.05
950908	79.45	78.75	79.15	8.63	494298000	(79.15+79.05)/2= 79.10	(79.45+78.75)= 79.10
950911	79.30	78.70	79.00	8.65	160563000	(79.00+79.15)/2= 79.075	(79.30+78.70)= 79.00
950912	79.65	79.10	79.40	8.60	120518000	(79.40+79.00)/2= 79.20	(79.65+79.10)= 79.375

| **950913** | 80.45 | 79.65 | 80.05 | 8.50 | 827299000 | **(80.05+79.40)/2= 79.725** | **(80.45+79.65)= 80.05** |

Table 1. Shows some quotes of "NYK 6% 3C 2026 4" in September 95, MovAvLast2Periods calculated on close prices, and HighLowAv calculated on high and low prices.

BC VIEW can calculate indicators on any data type. Let us say that you insert a graph that shows MovAvLast2Periods as a BarChart. Then BC VIEW will calculate MovAvLast2Periods for all the data types that are needed to draw the graph (High, Low, and Close). Thereby the high of the BarChart becomes the average of the high prices of the last 2 periods and so on. As for our example (Period = 04/09-95):

High: (78.80+78.60)/2= 78.70
Low: (78.20+78.00)/2= 78.10
Close: (78.55+78.35)/2= 78.45

When you create a graph, you specify for which data type(s) you want the graph to be shown. If the graph shows an indicator, the indicator may modify the data type or use more data types. The indicator cannot be said to show a data type if the indicator modifies it.

As an example of an indicator that both modifies the data type and uses more data types let us make an indicator to calculate the average between the high and the low price. It may look as follows:

Result := (High(Instrument) + Low(Instrument))/2

The indicator is calculated by (for each period) adding the high price of Instrument with the low price of Instrument and then dividing the result with 2. See Table 1.

Now let us look at signals. A signal can have 4 different returns namely LONG ⬆, EXIT LONG ⬆, SHORT ⬇, and EXIT SHORT ⬆. Each of these are calculated independently and returns either TRUE or FALSE. BC VIEW shows a signal in a view in one of 2 ways:

1.   As calculated ("Show All Signals"). Whenever LONG, EXIT LONG etc. **is** TRUE the returns are drawn.
2.   Only when traded. BC VIEW uses a simple model to simulate trades. The model moves between having bought (long), having sold (short), and being out of the market (out). Whenever LONG, EXIT LONG etc. **becomes** TRUE, BC VIEW will see if it needs to trade. If BC VIEW trades, the returns are drawn. Currently BC VIEW will not buy if already bought and sell if already sold. The following table describes what BC VIEW will do based on the returns which **become** TRUE. E.g. the table shows that if both LONG and SHORT **become** TRUE, BC VIEW will neither buy or sell (do nothing) The table:

NO	Exit Short	Short	Exit Long	Long	What BC VIEW does
0	FALSE	FALSE	FALSE	FALSE	Nothing
1	FALSE	FALSE	FALSE	TRUE	Buy
2	FALSE	FALSE	TRUE	FALSE	Go out if bought
3	FALSE	FALSE	TRUE	TRUE	Go out if bought
4	FALSE	TRUE	FALSE	FALSE	Sell
5	FALSE	TRUE	FALSE	TRUE	Nothing
6	FALSE	TRUE	TRUE	FALSE	Sell
7	FALSE	TRUE	TRUE	TRUE	Sell
8	TRUE	FALSE	FALSE	FALSE	Go out if sold
9	TRUE	FALSE	FALSE	TRUE	Buy
10	TRUE	FALSE	TRUE	FALSE	Go out
11	TRUE	FALSE	TRUE	TRUE	Go out
12	TRUE	TRUE	FALSE	FALSE	Go out if sold
13	TRUE	TRUE	FALSE	TRUE	Buy
14	TRUE	TRUE	TRUE	FALSE	Go out
15	TRUE	TRUE	TRUE	TRUE	Go out

A signal is split in a section for each return you want e.g. LONG and EXIT LONG. Each section holds a condition e.g. Instrument > 100. The condition determines the return of that section.

Let us make a simple signal that buys when the price have increased since the last period and goes out of the market when the price have decreased since the last period. It could look as follows:

LONG
  Instrument >Instrument[-1]
EXIT LONG
  Instrument < Instrument[-1]

The signal evaluates the LONG and the EXIT LONG sections independently. The LONG section evaluates Instrument > Instrument[-1]. Instrument > Instrument[-1] evaluates to TRUE if the price of Instrument is larger than it was in the last period. The EXIT LONG section evaluates Instrument < Instrument[-1]. Instrument < Instrument[-1] evaluates to TRUE if the price of Instrument is less than it was in the last period.

A signal can as indicators use more data types. To illustrate we could modify the previous example to buy when the high price increases and to go out of the market when the low price decreases:

LONG
  High(Instrument) >High(Instrument)[-1]
EXIT LONG
  Low(Instrument) < Low(Instrument)[-1]

Functions are calculated based on time as opposed to indicators and signals that are calculated based on periods. Where indicators and signals are calculated on the prices as they would be shown in a view - with a given data type and period - functions are calculated before prices have been split in periods and data type. Calculation of functions are solely based on the time of the prices.

Functions are calculated by calculating a value whenever there is a change in any of the instruments that are used in the function. This means that BC VIEW will start with the first available values, calculate the function, find and fetch the instrument(s) which first changes, calculate the function, find and fetch the instrument(s) that first changes, calculate the function, and so on.

To illustrate how functions are calculated we will look at an example that calculates Spread on two instruments:

Time	Realkredit 6% 2026	DS 7% St.lån 2024	Spread(DS 7% , RD 6%)
950914 09:00	80.45	84.95	84.95-80.45= 4.50
950914 11:00	80.65	85.35	85.35-80.65= 4.70
950914 12:00	80.80		85.35-80.80= 4.55
950914 13:00	80.70		85.35-80.70= 4.65
950914 14:00	80.80	85.20	85.20-80.80= 4.40
950914 15:00		85.30	85.30-80.80= 4.50
950914 16:00	80.95	85.40	85.40-80.95= 4.45

Table 2. Shows some quotes of "Realkredit 6% 2026" and "DS 7% St.lån 2024" for 14th of September 95 and the result for calling Spread.

In the example spread operates on prices stored by time. We see that prices exist for RD 6% at 09:00, 11:00, 12:00, 13:00, 14:00, and 16:00 and for DS 7% at 09:00, 11:00, 14:00, 15:00, and 16:00. Spread is calculated by subtracting the price that exists for RD 6% from the price that exists for DS 7% at given time. E.g. for 13:00 spread is calculated as the last known price of SD 7% namely the price at 11:00 subtracted by the price of RD 6% at 13:00: 85.35-80.70 = 4.65. Note that spread calculates a value each time there are a change in either RD 6% or SD 7%.
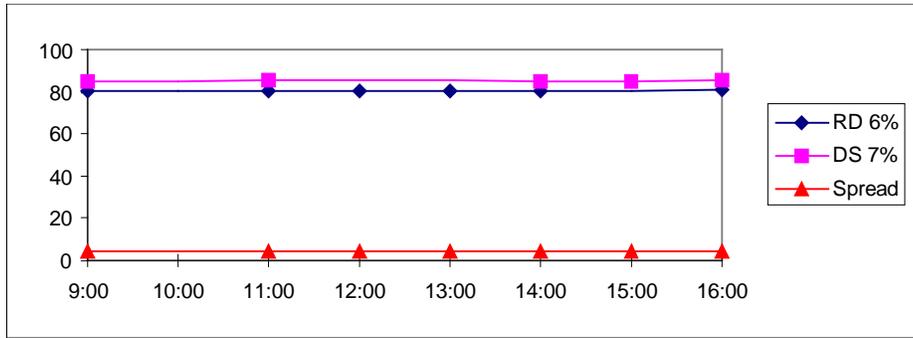
Figure 4. The figure shows the prices from table 2 to illustrate that spread calculates a value whenever there is a change in either RD 6% or DS 7%.

Now we have seen how a function (Spread) was calculated on prices stored by time. When prices are stored as Open, High, Low, Close, Yield, and Volume things are a little more complicated:
1. Prices. Functions will be calculated on Open, Close, and Time prices. Open prices are fixed on the time that the views (or instruments) exchange opens and Close prices are fixed just before the exchange closes.
2. Yield. Functions will be calculated on Yield. Yield is fixed just before the exchange closes.
3. Volume. Functions cannot be calculated based on Volume. Volume is a sum and BC VIEW cannot (as with prices) make the assumption that the last known volume can be used for further calculations

5   BC VIEW Language Description

BC VIEW contains a facility for writing your own functions, indicators, and signals. This can be useful when you want to try out your own ideas concerning the development of the financial market. These functions, indicators, and signals, which are denoted *interpreted*, are used just as the standard functions, indicators, and signals such as **MovAv**, **RSI**, **Rising** and **Falling**. They are written by the user by means of the *BC VIEW Language* (*BCL*) which is described in this chapter.

When writing an interpreted function or indicator , you specify how to compute the value, that the function or indicator returns. This computation can be done in several steps by computing intermediate results, storing these in variables and finally computing the desired result on the basis of these variables. All this is specified by means of the *BC VIEW Language* (*BCL*) except the input parameters which you must enter in the editor.

Interpreted signals are written in much the same way as interpreted functions and indicators, but instead of computing a return value you now specify one or more conditions for which signal types the signal itself should "return". That is, you specify a condition for one or more of: **LONG**, **SHORT**, **EXIT LONG** and **EXIT SHORT** - then, if a given condition is satisfied the corresponding signal type is returned by the interpreted signal return. The specification of signals resembles that of functions or indicators - the differences are described in the signal section.

5.1   Interpreted Functions

When BC VIEW calculates an interpreted function it is actually computing a single price of the function using the corresponding prices of each of the functions arguments. That is, when an interpreted function is calculated for a specific time the prices (values) of the arguments are taken for that time. The following examples will shed some light on this.

Example Spread

This function calculates the spread of two instruments - which is just the difference between the two. The function looks like this:

Interface:

   Spread(Security Instrument1; Security Instrument2)

Bcl:

> **Result := Instrument1 - Instrument2**

When you write a function you specify the *function interface* in the function editor. Here it is described in text: **Spread**, takes the two arguments, **Instrument1** and **Instrument2**. The type of the arguments are given as **Security** (Instrument).

The function directly calculates the return value of the function. Now, if **Spread** is to be calculated for a given time, the prices of **Instrument1** and **Instrument2** for that time are found, subtracted and returned as the result of **Spread**.

The line is actually an *assignment statement*. An assignment statement is interpreted as follows: the expression on the right-hand side of the assignment symbol (**:=**) is calculated and the result is stored in the variable on the left-hand side. In this example **Result** on the left-hand side specifies what the function return. **Result** is a *keyword*. A keyword is a word that is reserved by BCL, you cannot make a function, indicator, signal, or variable with the same name as a keyword.

Example **Cross**

This function calculates the cross between two instruments:

Interface:

> **Cross(Security Instrument1; Security Instrument2)**

Bcl:

> **Result := Instrument1/ Instrument2**

As in the preceding example, **Cross** is a function with two arguments, it returns the quotient between its first and its second.

If the functions in the examples above were to be displayed in a view they would be calculated for every single change in the parameters. The view then would show the calculations converted to its period and data type e.g. a barchart on a yearly basis.

It is also possible to write ordinary, mathematical functions in BCL.

Example Sqr

This illustrates how to write a mathematical function.

Interface:

> **Sqr(Number x)**

Bcl:

> **Result := x * x**

Sqr simply returns the square of its argument.

5.2 Interpreted Indicators

When the BC VIEW Language system calculates an interpreted indicator (or signal) it is computing a price for each period based on the price of the arguments for the same period.

Because indicators - as opposed to functions - operate on periods indicators can look at previous (and possibly future) periods. The next example will show how to access a previous period of an argument.

Example **Momentum**

Momentum is the ratio between the current period and the *n*th previous period. In BCL this can be expressed as follows:

Interface:

> **Momentum(Security Instrument; Integer n)**

Bcl:

> **Result := Instrument / Instrument[Date-n] * 100**

> The interface states that the **Momentum** indicator operates on two arguments: **Instrument** and the integer **n**.

> Now, when calling the indicator **Momentum** an instrument must be given as the first argument, whereas the second argument must be an integer, e.g. a figure - say, 20. The indicator calculates the quotient between the current period of **Instrument** and the **n**th previous period - denoted by **Instrument[Date-n]** - see below. This ratio is finally normalized by multiplying with 100.

The notation used in the preceding example shows the use of **[...]** to get other periods than the current one of an argument. Above we wrote **Instrument[Date-n]** indicating the **n**th *previous* period - relative to the current period. This is deduced as follows: the keyword **Date** is equivalent to the current period, so subtracting **n** gives the mentioned period. So, e.g. **Instrument** and **Instrument[Date]** are equivalent. **Date** can be omitted when combined with **-n**, that is, **Instrument[-n]** is short for **Instrument[Date-n]**. Future periods - as relative to current - are also accessible through the use of **Instrument[Date +n]**, or just **Instrument[n]**.

There is 3 more ways to specify a desired period. These are **[FirstDate]**, **[LastDate]** and **['dd?dd?dd']** indicating respectively the first and last period in the current time interval, and the period given by a specific date. These will be denoted *absolute* periods as opposed to the relative periods above. In **['dd?dd?dd']** each of the **dd**'s are a figure giving the date, month and year of the desired period and the **?**'s are some special characters, typically **.** (dot), **-** or **/**. The order of date, month and year is given by the user defaults.

Here is another example of indexing arguments:

> **Example MovAv**

> This examples gives an interpreted version of the build-in function **MovAv** (the moving average function). The function takes two arguments: an instrument to be averaged and a number indicating the number of periods to look back in the averaging. The function looks like this:

Interface:

> **MovAv(Security Instrument; Number n)**

Bcl:

> **total := Sum(i:=0; n-1; Instrument[-i])**

> **Result := total / n**

> The fist line calculates an intermediate result: the sum of the **n** previous periods - including the current period. This is done by use of the special function **Sum** to be described in the next paragraph. The result is stored in the variable **total**, which is used in the second line to calculate the return value of **MovAv**.

5.2.1 The Sum, Min and Max functions

The general format of the **Sum** function is as follows:

> **Sum**(<variable> **:=** <start value>**,** <end value>**,** <expression>)

Sum starts by assigning the value of <start value> to <variable>, it then calculates <expression> and stores the result temporarily. Next it adds one to <variable>, checks if this new value exceeds <end value>, if not, **Sum** evaluates <expression> (with the new value of <variable>), and adds this result to the temporary stored result. Then, **Sum** again adds one to <variable>, checks <end value> and so on. The final result of **Sum** is the value of the temporary (added) result when <variable> has exceeded <end value>. So, **Sum** adds all the results of <expression> for all the different values of <variable> in the range from <start value> to <end value> - both inclusive.

There're two other standard functions behaving much like **Sum: Min** and **Max**. They have the same format as **Sum** and they also evaluate <expression> for every value of <variable>, but instead of adding the results, they find the smallest respectively the largest of all the results. If <start value> is greater than <end value>, **Sum** returns 0 (zero), whereas the values of **Min** and **Max** are undefined in these cases.

Below some more examples on the use of **Sum**, **Min** and **Max** are given.

Example RSI

This is the relative strength indicator. It is calculated by taking the sum of the positive contributions of the previous n periods - relative to the previous period - and divide this by the sum of the absolute contributions. In BCL this can be calculated as follows. Three new standard functions have been used: **Pos** and **Abs** return the positive respectively the absolute value of their argument; the **If** function is explained in the paragraph below.

Interface:

   RSI(Security Instrument; Number n)

Bcl:

   SumPos := Sum(i:=1; n; Pos(Instrument[-i+1] - Instrument[-i]))

   SumTotal := Sum(i:=1; n; Abs(Instrument[-i+1] - Instrument[-i]))

   Result := If(SumTotal:=0; 50; SumPos/SumTotal * 100)

The **If** standard function is a function of three arguments. It checks if the first argument - a condition - is TRUE or FALSE. If it is TRUE, **If** evaluates the second argument and returns its value, otherwise **If** evaluates the third argument and uses this as the result. In the preceding example **If** was used to prevent an error caused by a division by zero.

5.2.2   Calling indicators from indicators

Now, we look at a more complex example involving 3 indicators. It demonstrates the ability to call other *interpreted* indicators from an interpreted indicator - just as if they were build-in indicators.

Example K, D, SlowD

The **K**, **D** and **SlowD** indicators are used to smoothen the curves just as the moving average function (**MovAv**), but they are more advanced than **MovAv**. They can be calculated as follows:

Interface:

   K(Security Instrument; Number n)

Bcl:

   L:= Sum(i:=1; n; Instrument[-i]) / n

   R:= Max(i:=1; n; Instrument[-i]) - Min(i:=1; n; Instrument[-i])

   Result := If(R=0; 100; (Instrument-L) / R * 100)

Note that the same variable (**i**) in **Sum**, **Max** and **Min** can be used several times without conflicts. Note also, that **L** is actually the moving average of **Instrument**. Now the function **D**:

Interface:

   D(Security Instrument; Number n1; Number n2)

Bcl:

   Result := Sum(i:=1; n2; K(Instrument; n1)[-i]) / n2

The second line of **D** uses a reference ([**-i**]) of the function **K**; this is done just as if **K** was an argument to the indicator **D** except that the arguments to **K** must be given before the period reference.

Finally **SlowD** is calculated as follows.

Interface:

   SlowD(Security Instrument; Number n1; Number n2; Number n3)

Bcl:

> **Result := Sum(i:=1; n3; D(Instrument; n1; n2)[-i]) / n3**

Here, the period referencing of a function, **D**, is used also.

Example A special index

This example illustrates the use of absolute references. The indicator calculates an index of three instruments weighted by 0.23, 0.47 respectively 0.30 with basis January, 1st 1980.

Interface:

> **MyIndex(Security Instrument1; Security Instrument2; Security Instrument3)**

Bcl:

> **IndexCalc := (Instrument1 * 23 + Instrument2 * 47 + Instrument3 * 30)/100**

> **Result := IndexCalc / IndexCalc['1/1-1980'] * 100**

Note the use of the **IndexCalc** variable; in the last line: this variable is referenced by an absolute date. When evaluating this, the BC VIEW Language system makes the reference for each of the factors in the expression which was assigned to **IndexCalc**. So

> **IndexCalc['1/1-1980'] is the same as**

> **(Instrument1['1/1-1980']* 23 + Instrument2['1/1-1980']* 47 + Instrument3['1/1-1980']* 30)/100.**

It is time to a warning. When **MyIndex** is applied for low prices all calculations are based on low prices. This may not be what you want. To avoid this problem make **MyIndex** as a function. Functions can also use absolute date references. The code - when created as a function - looks the same but because functions are calculated whenever a change in the instruments occur **MyIndex -** when applied for low prices as a function - return low of the values calculated by the function.

Finally an indicator which involves a special variable.

Example Standard Deviation

This example calculates the standard deviation of an instrument - deviated over **n** periods.

Interface:

> **StdDev(Security Instrument; Number n)**

Bcl:

> **Ma := MovAv(Instrument; n)**

> **SqInstrument := Instrument * Instrument**

> **Result := Sqrt(MovAv(SqInstrument; n) - Ma * Ma)**

Sqrt is the standard, square root function.

The **MovAv** function call in the last line calculates the moving average of the square of **Instrument** which is stored in the variable **SqInstrument**, that is, the average of the squared periods of **Instrument**.

5.2.3   Data Types

The preceding examples have no (explicit) reference to data types but actually all indicators must be defined for a specific set of data types and the same accounts for the arguments to the indicators, these types are specified in the interface description. The possible data types are:

> **Open, High, Low, Open, Yield, Volume**

Any combination of these can be specified as the type of an indicator or as the type of an argument.

Example **PeriodAv**

Say you want to write an interpreted indicator that returns the average between **Open** and **Close** of the period.

Interface:

PeriodAv(Security Instrument)

Bcl:

Result := (Close(Instrument) + Open(Instrument)) / 2

PeriodAv is calculated as the average of **Open** and **Close** of the current period.. The **Open** and **Close** periods of **Instrument** is reached through the use of a so-called *data extract* function. A data extract function is actually a data type treated as a function, so **Open(Instrument)** returns the **Open** data type of **Instrument** and **Close(Instrument)** returns the **Close** data type of **Instrument** for the actual period.

The data types are all *keywords* in the BCL.

5.2.4   Search Strategy

Whenever a non-standard function (as e.g. **Abs**) is used in BCL the function, indicator, or signal is found by its *name*, that is e.g., if you call the moving average indicator, **MovAv**, the BC VIEW Language looks through all its functions and indicators in the order specified below and the first function/indicator with the name **MovAv** is used. The order in which the search is done is as follows:

1.   User defined interpreted functions/ indicators
2.   Super User's interpreted functions/ indicators
3.   Build in functions/ Indicators

This fixed search strategy makes it possible for the user to overwrite functions/indicators/signals. E.g. if the user is not satisfied with the standard version of moving average he can specify his own indicator with the same name (**MovAv**). Whenever the system needs to evaluate **MovAv** it finds the user defined version and uses this. This procedure prevents having the user finding and renaming all occurences of **MovAv** in all interpreted functions.

5.3   Interpreted Signals

Interpreted signals are written in much the same way as interpreted indicators, but instead of returning a price the signal "returns" one of the four signal types LONG, SHORT, EXIT LONG and EXIT SHORT.
Here's a simple example of an interpreted signal:

Example **Moving Average Spread**

This signal signals SHORT if the moving average of a instrument is more than 4% under the current period, and it signals LONG if the moving average is more than 4% above the current period.

Interface:

MovAvSpread(Security Instrument; Number n)

Bcl:

Short

MovAv(Instrument; n) * 104/100 < Instrument

Long

MovAv(Instrument; n) * 96/100 > Instrument

The signal holds sections. Each section specifies one or more signal (return) types. First in a section you specify which signal types it returns. Secondly you give a condition (a Boolean expression). The condition determines if the signal

types are included in the return. Line 2 gives the condition for **MovAvSpread** to be Sʜᴏʀᴛ, that is, if the condition in line 2 is TRUE the signal "returns" Sʜᴏʀᴛ. Similarly for the signal type Lᴏɴɢ in line 4.

When writing an interpreted signal, one and only one section must be given for every signal type. In the preceding example **MovAvSpread** was defined for the two signal types Lᴏɴɢ and Sʜᴏʀᴛ.

Here are a few more examples of interpreted signals.

<u>**Example**</u> **Golden Cross Dead Cross**

This signal compares two moving averages. We go long when both a short and a long moving average are rissing and the short moving average are larger than the long moving average. We go short when both the short and the long moving average are falling and the short moving average are smaller than the long moving average.

Interface:

 GD_Cross(Security Instrument; Integer nShort; Integer nLong)

Bcl:

 ShortMovAv := MovAv(Instrument; nShort; 1)

 LongMovAv := MovAv(Instrument; nLong; 1)

 LONG

 Rising(ShortMovAv) AND Rising(LongMovAv) AND ShortMovAv>LongMovAv

 SHORT

 Falling(ShortMovAv) AND Falling(LongMovAv) AND ShortMovAv<LongMovAv

This signal first calculates the 2 moving averages called **ShortMovAv** and **LongMovAv**. Then the long section and the short section calculates each their boolean expression.

Rising is a boolean indicator which is TRUE if the current price is larger than the previous price:
 Instrument>Instrument[-1]

Falling is a boolean indicator which is TRUE if the current price is smaller than the previous price:
 Instrument<Instrument[-1]

<u>**Example**</u> **Golden Cross Dead Cross with reference instrument**

This signal is an extended version of the previous example. The idea is that we do not want to trade agaist the market. To know the market we look at a reference instrument. The reference instrument could be an index or such. E.g. if the instrument is a share it would be appropriate to use the local share index (e.g. *SP500* for US) as the reference instrument.

Interface:

 GD_DC2(Security Instrument; Integer nShort; Integer nLong; Security RefInstrument; Integer nRefInst)

Bcl:

 ShortMov :=MovAv(Instrument; nShort; 1)

 LongMov :=MovAv(Instrument; nLong; 1)

 ReffMov :=MovAv(RefInstrument; nRefInst; 1)

 LONG

 Rising(ShortMov) AND Rising(LongMov) AND ShortMov>LongMov AND Rising(ReffMov)

 SHORT

 Falling(ShortMov) AND Falling(LongMov) AND ShortMov<LongMov AND Falling(ReffMov)

<u>**Example**</u> **GD_CrossUSD_DEM**

This signal is an extended version of the basic Golden Cross Dead Cross example. The idea is to exit long if the german 10 years yield divided by the US 10 years yield have decreased more than 2% and to exit short if the german 10 years yield divided by the US 10 years yield have increased more than 2% - both compared to a moving average of same over the last 20 periods. The example also demonstrate that it is possible to reference instruments in BCL by their code.

Interface:

    GD_CrossUSD_DEM(Security Instrument; Integer nShort; Integer nLong)

Bcl:

    ShortMovAv := MovAv(Instrument; nShort; 1)

    LongMovAv := MovAv(Instrument; nLong; 1)

    Yield10Y_DemDivUsd := Yield("DEM10Y")/Yield("USD10Y")

    MovAv_Yield10Y := MovAv(Yield10Y_DemDivUsd; 20; 1);

    LONG

    Rising(ShortMovAv) AND Rising(LongMovAv) AND ShortMovAv>LongMovAv

    EXIT LONG

    Yield10Y_DemDivUsd <MovAv_Yield10Y*98/100

    SHORT

    Falling(ShortMovAv) AND Falling(LongMovAv) AND ShortMovAv<LongMovAv

    EXIT SHORT

    Yield10Y_DemDivUsd >MovAv_Yield10Y*102/100

The example shows a variable **Yield10Y_DemDivUsd**. The variable is calculated based on two instruments - **"DEM10Y"** and **"USD10Y"**. The instruments are specified directly in BCL and not - as we have seen previously - specified as arguments. You can also access instruments by their code in functions and indicators.

6   **Formal BCL Specification**

A programming language can formally be described by its syntax and its semantics. The syntax is the building blocks of the language and the semantics are the rules for connecting those building blocks. The syntax will be described using BNF (Backus Naur Normal Form) and the semantics will be described in English.

The return type of functions, indicators and the names and types of formal arguments are given in the interface description which is passed to the compiler before the actual compilation. Inside the functions and indicators the formal arguments are referred by the names given in this interface. The return type of signals are part of the BCL language.

BNF may at first glance look a little complicated but it is just an excellent tool for programming. You can use it to determine how to write your functions, indicators, or signals. Basically BNF describes a building block of the language as consisting of a number of other building blocks - down to characters etc.

The description of BCL use the following BNF symbols:
- **<>** *building block*. The < and the > is used in representation of building blocks. E.g. the description of BCL holds a building block called **variable**. It is represented in BNF as **<variable>**.
- **|** *option (or)*. The | is used to represent a choice. E.g. a digit is one of **0 | 1| 2| 3| 4 | 5 | 6 | 7 | 8 | 9**.
- **::=** *consists of*. In BNF **::=** describes a building block as some combination of building blocks. E.g. the description of BCL describes a data type as: **<data type> ::= High | Low | Open | Close | Volume | Yield**. This means that data type is one of High, Low, Open, Close, Volume, or Yield.

{} *zero or more times*. What is inside **{** and **}** can be repeated zero or more times. E.g. an identifier can be described as**: <identifier> ::= <letter> { <letter> | <digit> | _ }**. This means that an identifier consists of a letter that might be followed by a number of letters, digits, or _'s.

î *empty*. BNF uses **î** to tell if a building block can be missing. E.g. in BCL a variable might be followed by a date reference. Therefore a date reference is described as**: <date reference> ::= [<date expression>] | î**. This means that a date reference is either a [followed by a date expression followed by a] or it is not there.

In order to not complicate things unnecessary the syntax and the semantics explained will be those of indicators and signals. The syntax and semantics of functions are a subset of those of indicators - in functions you cannot use [] (except absolute) and data types. The semantics are not fully explained. In the following the mathematical term set is used for security.

6.1   Function / Indicator

An interpreted function or indicator looks as follows:

Syntax:
    { <assignment statement> }
    Result := <expression>

Semantics:
    The assignment to the keyword Result in the specification is an assignment to the function / indicator. The value of the function / indicator becomes the value of this assignment statement.

6.2   Signal

The specification of signals resembles that of indicators except that the assignment to "Result" is substituted by one or more signal statements. The syntax is as follows:

Syntax:
    <signal body> ::= { <assignment statement> | <signal statement> }
    <signal statement> ::= <signal list> <boolean expression>
    <signal list> ::= <signal type> { , <signal type> }
    <signal type> ::= LONG | EXIT LONG | SHORT | EXIT SHORT

Semantics:
A given signal type is signaled if, and only if, the boolean expression in the signal statement containing the actual signal type evaluates to TRUE. A signal body cannot hold more than a single signal statement for each of the signal types specified in the description of the signal interface.

Assignment statements must preceed signal statements.

Note: If the signal is to be used in the optimize module it must hold at least a (EXIT) SHORT and a (EXIT) LONG signal.

6.3   Assignment Statement

The only statement in BCL is the assignment statement. It is as follows:

Syntax:
    <assignment statement> ::=
        <variable> := <expression> |
        <variable> [] := <expression> |
        <variable> := <boolean expression>

Semantics:

Variables in BCL can be sets (e.g. instruments that holds a lot of values) or simple (a single value) and they holds (the type) either values or boolean values (TRUE or FALSE). When assigning to a variable, the type of the variable and the type of the expression must be the same e.g. you cannot assign a boolean variable to a variable that can only hold a value.

6.4   Expression

This section contains the complete syntax of expressions.

Syntax:

```
<expression> ::= <term> { <adding operator> <term> }
<term> ::= <factor> { <multiplying operator> <factor> }
<factor> ::= <sign> <factor> |
    ( <expression> ) |
    <function / indicator call> <date reference> |
    <constant> |
    <instrument reference> <date reference> |
    <variable> <date reference> |
    <variable> [] |
    <variable>
<sign> ::= + | - | î
<adding operator> ::= + | -
<multiplying operator> ::= * | /
<function / indicator call> ::= <function / indicator> ( <actual argument list> ) |
    <loop> (<variable>:=<start value>; <end value>;<expression>) |
    If (<boolean expression>;<expression>;<expression>)
<loop> ::= Sum | Max | Min
<data extract> ::= High | Low | Open | Close | Volume | Yield
<math function> ::= Pos | Neg | Abs | Sqrt | Round | Log | Ln | Exp | Sin | Cos | Tan
<boolean indicator> ::= Rising | Falling
<build-in indicator>::= MovAv | Momentum | Rsi | Parabolic | Stddev
<build-in function> ::= Index | Spread | Cross
<function / indicator> ::= <function / indicator name> | <build-in>
<build-in> ::= <data extract> |
    <boolean indicator> |
    <math function> |
    <build-in indicator> |
    <build-in function>
<actual argument list> ::= <actual argument> { ; <actual argument> } | î
<actual argument> ::= <expression> | <boolean expression>
<date reference> ::= [ <date expression> ] | î
<date expression> ::= <relative date> | <absolute date>
<relative date> ::= Date | Date <adding operator> <variable> |
    Date <adding operator> <simple figure> |
    <sign> <variable> | <sign> <simple figure>
<absolute date> ::= FirstDate | LastDate |
    ' { <letter> | <digit> } '
<constant> ::= <figure>
<figure> ::= <simple figure> { <th sep> <simple figure> } |
    <simple figure> { <th sep> <simple figure> } <comma part> |
    <comma part>
<th sep> ::= , | .
<comma part> ::= <frac comma> <simple figure>
<frac comma> ::= . | ,
<simple figure> ::= <digit> { <digit> }
<instrument reference> ::= <instrument constant> <date reference>
<instrument constant> ::= "{ <letter> | <digit> } "
<boolean expression> ::= <boolean term> { or <boolean term> }
```

<boolean term> ::= <boolean factor> { and <boolean factor> }
<boolean factor> ::= Not <boolean factor> |
    (<boolean expression>) |
    <function / indicator call> |
    <boolean constant> |
    <variable> |
    <expression> <relational operator> <expression>
<relational operator> ::= = | <> | < | <= | >= | >
<boolean constant> ::= TRUE | FALSE
<variable> ::= <identifier>
<identifier> ::= <letter> { <letter> | <digit> | _ }
<letter> ::= A | B | C | ..... | Z | a | b | ... | z | <National letter>
 <digit> ::= 0 | 1 | 2 | ... | 9

Semantics:

The syntax implicitly defines the precedence rules for expressions: Signs have precedence over multiplication (*) and division (/) which precede over addition (+) and subtraction (-) - i.e. the common precedence rules for arithmetic applies. Similarly for boolean expressions - Not has precedence over And which precedes over Or.

<Indicator or function call>
The actual argument list must correspond to the formal argument list defined in the interface description or, for build-in functions, in the definitions given. The lists must correspond in number of arguments and in type (Boolean, Value or Set) to the individual arguments.

<date reference>
A date reference is either relative to the current date ('Date') absolute or missing. A missing reference is interpreted as a reference to the current date - as Set[Date]. A relative date seems a little subtle, but is effectively just specified by the optional Date followed by a sign and either a simple figure or a simple variable. The actual number always represents periods of the viewed basis (monthly, weekly, daily etc.). An absolute date is either FirstDate, LastDate (see below) or a specific date, e.g. '1/1-80'. The specific format of a date is determined by the user settings. If there is no period on the specified date the next following period is used. The two named absolute dates FirstDate and LastDate are:
    FirstDate - The date of the first period (of the viewed periods).
    LastDate - The date of the last viewed period.
Relative date references cannot be used in functions.

<data extract>
Date extracts cannot be used in functions.

<constant>
Numerical constants are interpreted in the normal way. The actual thousand separator (<th sep>) and fraction comma (<frac comma>) are determined by the user settings. A fraction comma followed by a simple figure is implicitly preceded by a 0 (e.g. .123 = 0.123).

<instrument reference>
A specific instrument is specified in an instrument reference. The specific format of the reference is given by the instrument code.

<boolean factor>
A function call in a boolean expression must be a boolean typed function, and a variable must be a boolean argument or another boolean variable.

6.4.1 Keywords
This is a list of the keywords in BCL:

Abs	AND	Close	Cos	Date	Exp	EXIT	FALSE
FirstDate	High	If	LastDate	Ln	Log	LONG	Low

Max	Min	Neg	NOT	Open	OR	Pos	Result
Round	SHORT	Sin	Sqrt	Sum	Tan	TRUE	Volume
Yield							

6.4.2   If-function

The if pseudo function is used to select a value or a set according to a boolean expression.

Syntax:

If (<boolean expression> ; <expression>; <expression>)

Semantics:

Returns the value of the first expression if the boolean expression evaluates to TRUE, otherwise the value of second expression is returned.

6.4.3   Loop functions

The three pseudo functions Sum, Min and Max are the so-called loop functions. They have a local loop-variable as a counter.

Syntax:

<loop> (<variable>:=<start value>; <end value>;<expression>) |
where <loop> ::= Sum | Min | Max

Semantics:

<Expression> is evaluated for <variable> equal to the values <start value>, <start value>+1,..., <start value>. The result of the function is equal to the sum, minimum or maximum of all the evaluations. If <end value> is less than <start value> the value of Sum is zero, whereas the values of Min and Max are undefined in this case.

6.4.4   Data extracts

These pseudo functions tell the interpreter the specific period type wanted from the argument. The argument is restricted to a specific instrument, an indicator call or an indicator argument.

Syntax:

<data extract> ::= High | Low | Open | Close | Volume | Yield

Semantics:

The <data extract> 's is called as their were functions with a single argument and returns the specified period type of the specified instrument etc. E.g. the High data extract returns the high value of the argument.

6.4.5   Mathematical Functions

The BC VIEW Language contains the following standard functions. All functions except **Round** takes a single number as argument and returns a number. **Round** takes two numbers as argument and returns an integer.

Function name	Returns
Abs(Number)	The absolute (the numerical) value of Number
Cos(Number)	The cosius of Number.
Exp(Number)	The exponential function of Number.
Ln(Number)	The natural logarithm of Number
Log(Number)	The logarithm with base 10
Neg(Number)	The Number itself, if it is less than zero, otherwise 0 is returned
Pos(Number)	The Number itself, if it is greater than zero, otherwise 0 is returned

Round(Number; Number2)	The multiplum of Number2 nearest to Number. E.g. Round(17; 7) = 14 (2 * 7).
Sin(Number)	The sinus of Number
Sqrt(Number)	The square-root. Value must be greater than or equal to 0.
Tan(Number)	The tangent of Number

7   List of functions

Function name / Arguments	B/I	Explanation
Cross Instrument1: Instrument Instrument2: Instrument	B	Calculates the ratio between two arguments. Usage: The function is normally used when you add new Currency Crosses to the list of instruments. E.g. USD/DEM can be computed on the basis of USD/CHF and DEM/CHF, where USD/CHF is the first argument and DEM/CHF the second.
CrossM100 p1: Instrument p2: Instrument	I	Cross multiplied by 100. See Cross.

Index
 	
IndexValue: Value where the index start often 100.
Instrument: Instrument.
Weight: The weight of the instrument when the index is calculated.
	B	Index is the only function/ indicator/ signal that takes a variable number of arguments. Index is called with an index value and a number of instrument, weight pairs: Index(<IndexValue>{,<Instrument>,<Weight>}).
		
		Index returns the normalized weighted sum of the instruments. The sum is normalized to IndexValue and weighted with the Value. It is calculated as:
		    IndexValue * Sum(Instrument's * Weight's)
		    ------------------ * ----------------
		    Index[FirstDate]* Sum(Weight's)
		
		Usage:
		    The function can be used to create your own index's as instruments or it can be used directly in a view.

Invert P: Instrument	I	Calculates 1/P.
Ratio Instrument1: Instrument Instrument2: Instrument	I	Calculates (Instrument1/Instrument2)* 100
Spread Instrument1: Instrument Instrument2: Instrument	B	Calculates the spread (difference) between two arguments. Usage: The function is normally used to calculate spread between 2 yields.
TECFunc	I	

8   List of indicators

Indicator name/	B/I	Explanation

Arguments	Data Set	
%CHANGE P: Instrument n: No. of periods used in calculation of %CHANGE	I OHLCYV	Calculate the relative change between today and n periods ago
ADXR P: Instrument n: No. of periods used in calculation of DMI	I OHLCYV	Calculates the average of DMI of current period and DMI 14 periods back. See DMI.
BOLLINGERdown P: Instrument n: No. of periods used in calculation of Moving Average x: Factor for stddev	I None	See BOLLINGERmid.
BOLLINGERmid P: Instrument n: No. of periods used in calculation of Moving Average	I None	Bollinger Bands. Use BOLLINGERmid with BOLLINGERdown and BOLLINGERup. The bollinger bands creates a band around a moving average calculated over n days. The factor x gives the width of the band. Trading bands are among the most widely used technical indicators in existence. Basically they are lines drawn at fixed intervals (usually a percentage) around a moving average. Bollinger Bands are bands that vary in distance from the moving average based on volatility. The upper band represents x number of standard deviations above the average, the lower band represents x number of standard deviations below the average. By using standard deviations rather than a fixed percentage, the bands adjust for volatility. During volatile periods the bands move further away from the average, while during market lulls the bands move closer to the average. Usage: The closer prices move to the upper band, the more overbought the market is; the closer prices move to the lower band, the more oversold the market is.
BOLLINGERup P: Instrument n: No. of periods used in calculation of Moving Average x: Factor for stddev	I None	See BOLLINGERmid.
CCI P1: Instrument n: No. of periods used in calculation of Moving Average	I None	Commodity Channel Index (CCI). CCI is designated to detect beginning and ending market trends. The computational procedure standardizes market prices much like a standard score in statistics. The Final index attempts to measure the deviation from normal or major changes in the markets trend according to the original author, 70% to 80% of all price fluctuations fall within +100 and -100 as measured by the index. A thorough discussion of

		CCI can be found in the October 1980 edition of Commodities magazine (now Futures). Usage: The trading rules for the CCI are as follows. Establish a long position when the CCI exceeds +100. Liquidate when the index drops below +100. For a short position, you use the -100 value as your reference point. Any values less than -100, e.g. -125 suggests a short position, while a rise to -85 tells you to liquidate your short position.
CCIsmoothed P1: Instrument n: No. of periods used in CCI n1: No. of periods used in calculation of Moving Average Type: Type for moving average	I None	CCIsmoothed calculates a moving average on CCI. See CCI.
CorrelationCoef P1: Instrument 1 P2: Instrument 2 n: Number of periods	I OHLCYV	Simple Linear Correlation Model. Usage: Correlation Coefficient is not a traditional technical analysis tool. It does not give the trader buy or sell signals; but it does tell you a great deal about the correlation between 2 instruments. The Correlation Coefficient is a value between 1 and -1. The value is a measurement for the 'strength' of the linear relationship between the 2 instruments. A value close to zero indicate a weak relation; values close to +1.0 indicate a strong 'possitive' correlation, and values close to -1.0 indicate a strong 'negative' correlation.
Dlm P: Instrument n: No. of periods	I None	DIRECTIONAL INDEX MINUS (Dlm) See the explanation for DMI
Dlp P: Instrument n: No. of periods	I None	Directional Index Plus (DIp) See the explanation for DMI
Divergensindicator P: Instrument CentralKurs: Weight:	I OHLCYV	
DMI P: Instrument n: No. of periods	I None	Directional Movement Index (DMI). The DMI is a trend following system. The average directional movement index determines the market trend. When used with the up and down directional indicator values, + DI(DIp) and - DI(DIm), the DMI is an exact trading system. The computations needed to generate the final figures for the DMI are not complex but are numerous and lengthy. If you need further explanation about the computations, please refer to J. Welles Wilder, Jr. book. Usage: The rules for using the DMI are simple. You establish a long position whenever the DIp crosses above the DMI. You reverse that position, liquidate the long position and establish a short position, when the DIm

		crosses above the DIp. In addition to the crossover rules, you must also follow the extreme point rule. When a crossover occurs, use the extreme price as the reverse point. For a short position, use the high made during the trading interval of the crossover. Conversely, reverse a long position using the low made during the trading interval of the crossover. You maintain the reverse point, the high or low, as your market entry or exit price even if the DIp and the DIm remain crossed for several trading intervals. This is supposed to keep you from getting whipsawed in the market. For some traders, the most significant use of the DMI-line(also called the ADX) is the turning point concept. First, the ADX must be above both DI lines. When the ADX turn slower, the market often reverses the current trend. The ADX serves as a warning for a market about to change direction. The main exception to this rule is a strong bullmarket during a blow-off stage. The ADX turns lower only to turn higher a few days later. According to the developer of the DMI, you should stop using any trend following system when the ADX is below both DI lines. The market is in a choppy sidewise range with no discernible trend.

EMS
 	
  P: Instrument
  Midt:
	I
	
	OHLC	

Falling Instrument	B Boolean	Falling returns true if the price of the current period is less than the price of the previous price.

Forward
 	
  P: Instrument
  P1: Instrument
  P2: Instrument
  n:
	I
	
	OHLCYV	

GainSince96 GainSince97 GainSince98 P: Instrument	I OHLCYV	Compute the relativ change since 1/1-1996 - 1/1-1997 -1/1-1998 etc.

IndexRela
 	
  P1: Instrument
  P2: Instrument
  n: No. of periods used in
    calculation of IndexRela
	I
	
	OHLCYV
		This Indicator rebase Instrument 2 (P2) with :
		Basedate: n periods ago
		Basevalue: The value for P1 n periods ago
		
		If this indicator e.g. is used in a view with a share (e.g. Microsoft) as base-graph and you Insert a new graph IndexRela(Microsoft; Dow Jones; 250), you will see Microsoft in real quotes and Dow Jones rebased with startvalue equal Microsoft's quote 250 periods ago.

IndexRela97
 	
  P1: Instrument
  P2: Instrument
	I
	
	OHLCYV
		This Indicator rebase Instrument 2 (P2) with :
		Basedate: 961231
		Basevalue: The value for P1

		If this indicator e.g. is used in a view with a share (e.g. Microsoft) as base-graph and you Insert a new graph IndexRela97(Microsoft; Dow Jones), you will see Microsoft in real quotes and Dow Jones rebased with startvalue equal Microsoft's quote 961231.
MACD P: Instrument n1: Number of. periods in fast moving average. n2: Number of periods in slow moving average. n3: Number of periods in moving average on spread of the 2 moving averages	I OHLCYV	Moving Average Convergence Divergence (ACD). Gerald Appel's MACD is the difference between a fast exponential moving average (on n1) and a slow exponential moving average (on n2). During rising markets the fast moving average will rise more quickly than the slow moving average resulting in a rising differential line or a larger value. During falling markets the fast moving average will fall more quickly than the slow moving average line. The specified length of the fast moving average must be shorter than the specified length of the slow moving average, if not, the oscillator will invert (that is, buy signals will become sell signals and sell signals will become buy signals). Usage: Crossover of the fast moving average line over the slow moving average line is a buy signal. Crossover of the fast moving average under the slow moving average line is a sell signal. Reference: Gerald Appel, Signalert Corporation, 150 Great Neck Road, Great Neck, NY 11021
Momentum P: Instrument n: Number of periods to look back.	B OHLCYV	Momentum monitors the change in prices. It tells you whether prices are increasing at an increasing rate or decreasing at a decreasing rate. Momentum measures the price of to day relative to the price n'th periods ago. Momentum swings around a 100 line. If momentum is above the 100 line but is declining , prices are still increasing but at a decreasing rate. Usage: The normal trading rule is simple. Buy when the momentum line crosses from below the 100 line to above. Sell when the momentum line crosses from above the 100 line to below. Another possibility is to establish bands at each extreme of the momentum line. Initiate or change positions when the indicator enters either of those zones. You could modify that to enter a position only when the indicator reaches the overbought or over-sold zone and then exits that zone. You specify the length of the momentum indicator. You must determine a value suitable to your trading needs and methods. Some technicians argue the length of the momentum indicator should equal the normal price cycle. The best method is to experiment with different lengths until you find the length that works best for that particular instrument you are trading.
MovAv Instrument: Instrument n: Number of periods to calculate the moving average on. Type: The moving average type (how it is calculated). See explanation	B OHLCYV	Moving averages are one of the most commonly used technical tools. They follow the trend, smooth the normal fluctuations of the data, and clearly signal long and short positions. Usage: The trading rules etc. are numerious. Some use Golden cross/Death crosses, some use envelopes, some use High/Low moving averages, some use combinations of the three different types of moving averages etc. The normal moving average crossover buy/sell signals are (using 2 moving averages) as follows. Buy when a short increasing moving average cross

		from below to above an increasing longer term average. Conversely sell when a short decreasing moving average cross from above to below a decreasing longer term average
		The normal moving average crossover buy/sell signals are (using 3 moving averages) as follows. A buy signal is flashed when the short and intermediate term averages cross from below to above the longer term average. Conversely, a sell signal is issued when the short and intermediate term averages cross from above to below the longer term average. See also Oscillator and MACD
		
		Call:
		BC VIEW allows you to be very creative with the moving averages. You may select 4 different averages (the third argument in the formula):
		   1. Simple moving average.
		   2. Exponential moving average.
		   3. Weighted moving average.
		   4. Accumulated moving average.

MovAvDisp Instrument: Instrument n: Number of periods to calculate the moving average on. Type: The moving average type (how it is calculated). See explanation Displaced. Number of periods to displace the moving average.	I OHLCYV	Calculates a moving average that is displaced by a number of periods.
Mvolume P: Instrument n: No. of periods	I	Compute total volume (price * no. of traded units) the last n periods

OSC
 	
P1: Instrument
n1: Number of periods to calculate the first moving average on.
n2: Number of periods to calculate the second moving average on.
Snittype: The moving average type (how it is calculated). See explanation.
	I
	
	OHLCY
		Oscillator (OSC). An oscillator is the simple difference between 2 moving averages. Moving averages are one of the most commonly used technical tools. They follow the trend, smooth the normal fluctuations of the data, and clearly signal long and short positions.
		You may select up to four different averages:
		   1. Simple Moving average
		   2. Exponential Moving average
		   3. Weighted Moving average
		   4. Accumulated Moving average
		
		Usage:
		One trading rule is simular to the crossover system used in moving averages. In fact, the oscillator is another method of using two moving averages. Sell when the oscillator crosses the zero line from above to below. Buy when the oscillator crosses the zero line from below to above. Some traders buy the valleys and sell the peaks of the oscillator. For further explanation see explanation for Moving Averages (MovAv).

OSCline P1: Instrument n1: Number of periods for first moving average in oscillator. n2: Number of periods for second	I OCHLY	Line Oscillator (OSCline). OSCline compute a moving average on the oscillator. For further explanation - see explanation for Moving Averages and Oscillator.

moving average in moving average. n3: Number of periods used by the moving average calculated on the oscillator. Snittype: The moving average type (how it is calculated). See explanation on MovAv.		

Parabolic
 	
Instrument: Instrument
AccStart:
AccAdd: Acceleration factor
AccLimit:
	B
	
	None
		The Parabolic Time/Price system indicator. The Parabolic Time/Price system is a simple study to use. The study continuously computes 'stop and reverse' price points (SAR) and a position is entered when a rice penetrates the SAR. The universal formula for the SAR is:
		
		$SAR(t)=SAR(t-1)+(A * EP(trade)-SAR(t-1))$
		where
		   SAR(t) is the stop and reverse price for the current interval
		   SAR(t-1) is the stop and reverse price for the previous interval
		   A is the acceleration factor
		   EP(trade) is the extreme price for the trade
		The acceleration factor is a weighting factor. In Wilders work , the initial value for the acceleration factor is 0.02. The acceleration factor increases by a value of 0.02 each time the extreme price changes for the trade. You do not increment the acceleration factor if the extreme price fails to change. The value for the acceleration factor never exceeds 0.2 in Wilders methodology.
		
		Usage:
		Wilder has rules for the parabolic time/price study. They are: First, a position is entered whenever prices penetrate the SAR. When this occurs, the new SAR becomes the extreme price from the previous trade. Next, the acceleration factor only increases when a new extreme price is set. When Long , you only increment the acceleration factor when the market sets a new high, a new extreme price for the trade. Conversely, you increase the acceleration factor when new lows are made in a short position. If the extreme price does not change, you use the previous acceleration factor in the computations. Do not increase the acceleration factor. Lastly, the new SAR price must not enter the trading range for the current or previous interval. In a long position, the new SAR must not exceed the low for current trading interval nor may it exceed the low for the previous trading interval. When short, the new SAR never falls below the high of the current or previous trading interval. If this condition occurs, you use either the lowest value of the two lows for a long position or the highest value of the two highs for a short position for the new SAR price.
		If you need further details, please read Wilder's book, New Concepts in Technical Trading Systems.

Rising Instrument	B Boolean	Rising returns TRUE if the price of the current period is larger than the price of the previous period.
RSI Instrument: Instrument n: Number of periods to calculate RSI over.	B OHLCYV	The Relative Strength Indicator (RSI). The RSI is a J.Welles Wilder,Jr. trading tool. The main purpose of this study is to measure the market's strength and weakness. The RSI measures the average of 'UP' movements in the period relative to the sum of 'Up' and 'Down' movements in the period. The RSI always falls between a value of 100 and 0.

		Usage: You can use the RSI as an overbought/oversold indicator. A high RSI, above 70 (20 period RSI), suggests an over bought or weakening bull market. Conversely, a low RSI, below 30 (20 period RSI), implies an oversold market or dying bear market. Another use of the RSI is divergence. Market prices continue to move higher/lower while the RSI fails to move higher/lower during the same time period. Divergence may occur in a few trading intervals, but true divergence usually requires a lengthy time frame, perhaps as much as 20 to 60 trading intervals. Selling when the RSI as above 70 or buying when the RSI is below 30 can be an expensive trading system. A move to those levels is a signal that market conditions are ripe for a market top or bottom. A failure swing or divergence accompanies your best trading signals.

RSIma
 	
Instrument: Instrument
n: Number of periods to calculate RSI over.
Type: See explanation.
	I
	
	OHLCYV
		Relative Strength Indicator based on any type of moving average.
		The build-in RSI is based on simple moving averages. RSIma can be based on any moving averages. RSIma will, if moving average type:
		1. be based on simple linear moving averages which generates a RSIma equal to the build-in RSI.
		2. be based on exponential moving averages.
		3. be based on weighted moving averages.
		4. be based on accumulated moving averages which generates a RSI using the work sheet method.
		
		The equation for the Relative Strength Index, RSI, is:
		$$RS = \frac{\text{Sum of period to period increases}}{\text{Sum of absolute values of period to period changes}} * 100$$
		which often is written as:
		$$RSI = 100 - (100 / (1 + RS)), \text{ where}$$
		$$RS = \frac{\text{Average of period to period increases}}{\text{Average of period to period decreases}}$$
		The last formula has been developed in order to calculate RSI on daily basis using a work sheet. The work sheet method calculate a new value of an average based on the previous value of the average. This calculation of an average is also known as an accumulated moving average - type 4 of the build-in MovAv indicator.
		For information on the work method see 'New Concepts in Technical Trading Systems by Wilder page 64.
		We have implemented the RSIma using the build in MovAv indicator as such:
		    AvUp := MovAv(Pos(P-P[-1]);N; Type)
		    AvAllUpDown :=MovAv(Abs(P-P[-1]);N; Type)
		    Result :=If(AvAllUpDown<P * 0.0000000001 ;50;
		    (AvUp/AvAllUpDown)* 100)
		The 'If' handles the situation where no changes in the price exist (AvAllUpDown = 0). Because accumulated and exponential moving averages are not guaranteed to become equal to the price, when there are no further changes, we accept a value as being '0' when it is a lot smaller than the price. 'AvAllUpDown<P * 0.000000001' makes RSIma become 50 when the average of the changes are less than a billion to the price.

Spread_DMp_DMm P: Instrument n: No. of periods	I None	Calculates the spread between Dmp and DMm
StdDev Instrument: Instrument Number: Number of periods to calculate the standard deviation over.	B OHLCYV	Standard Deviation. Computes the standard deviation for an instrument over a given number of periods.
StDev Instrument: Instrument Number: Number of periods to calculate the standard deviation over.	I None	Standard Deviation calculated on the average of the high, low, and Close price.
Stochatic%D P1: Instrument n1: Number of periods to calculate the stochastic on. n2: Number of periods to use in moving average to smoothen Stochatic%K	I None	Stochastic%D. A moving average on Stochastic%K. See Stochatic%K.
Stochastic%K P1: Instrument n1: Number of periods to calculate the stochastic on.	I None	Stochatic%K. Dr. George C. Lane is the author of the stochastic indicator. His basic premise is as follows; During periods of price decreases, daily closes tend to accumulate near the extreme lows of the day. Periods of price increases tend to show closes accumulating near the extreme highs of the day. The stochastic study is an oscillator designed to indicate oversold and overbought market conditions. Some technical analysts prefer the slow stochastic rather than the normal stochastic. The slow stochastic is simply the normal stochastic smoothed via a moving average technique. The normal Stochastic generates 2 lines. They are the %K and %D (A moving average on Stochastic%K). The normal stochastic has overbought and oversold zones. Dr. Lane suggests using 80 as the overbought zone and 20 as the oversold zone. Others prefer 75 and 25. Dr. Lane also contends the most important signal as divergence between %D and the instrument. He explains divergence as the process where the slow stochastic %D line makes a series of lower highs while the security makes a series of higher highs. This signals an overbought market. An oversold market exhibits a series of lower lows while the %D makes a series of higher lows. When one of the above patterns appear, you should anticipate a market signal. You initiate a market position when the %K crosses the %D from the right-hand side. A right-hand crossover is when the %D bottoms or tops and moves higher or lower and the %K crosses the %D line. According to Dr. Lane, your most reliable trades occur with divergence and when the %D is between 10 and 15 for a buy signal and between 85 and 90 for a sell signal.

StochasticSlow%D P1: Instrument n1: Number of periods to calculate the stochastic on. n2: Number of periods to use in moving average to smoothen Stochatic%D and Stochatic%K Type: Type of moving average to smoothen Stochatic%D.	I None	StochasticSlow%D. A moving average on Stochastic%D. Stochatic%D is a moving average on Stochatic%K - therefore StochasticSlow%D in fact is a moving average on a moving average on Stochatic%K. See Stochatic%K.
Tillaeg P: Instrument P2: Instrument 2 P3: Instrument 3 n:	I OHLCYV	

TrueRange
 	
P1: Instrument
	I
	
	None
		True Range is defined as the greatest of the following:
		    The distance from today's high to today's low
		    The distance from yesterday's close to today's high
		    The distance from yesterday's close to today's low
		Usage:
		True range is e.g. used in computation of The Volatility Index
		and in the Directional Movement Index

Volatility P: Instrument n: Length of standard deviation on logarithmic basis. m: Number of n's per year	I OHLCYV	Volatility. Usually market volatility is reported on an annualized basis. Numbers are usually gathered each day at the close, and then combined with a specified number of days over a preceding period (10, 30 or 90 days). The volatility is measured in pct. Usage: Historical volatility is not a traditional technical analysis tool. It does not give the trader buy or sell signals. It does tell you a great deal about past price variability. Declining volatility implies an orderly market, i.e., price variation is relatively stable, not erratic. The opposite is true with rising volatility.
VolatilityIndex P1: Instrument n1: Length of Average on TrueRange.	I None	VolatilityIndex. The Volatility Study measures market volatility by plotting a smoothed average of true range. As the number increases, the market is more volatile. Usage: The concept of Average TrueRange can be used in determining how far from the market your protective stops need to be placed so as to avoid whipsaws.
VolumeFilter P: Instrument r: Volume that must have been traded in order to have VolumeFilter return a price.	I OHLCY	The indicator returns the last price where volume is larger than r.
Willams%R P1: Instrument n1: Number of periods to calculate Willams%R on.	I None	William's' Percent R (Williams%R). Noted author and commodity trader, Larry Williams, developed trading formula called the %R. In his original work, the method examined 10 trading days to determine the trading range. Once the 10 day trading range was determined, he calculated where today's closing price fell

		within that range. Williams%R is designed to detect overbought and oversold market conditions. The %R always falls between a value of 100 and 0. Usage: The trading rules are simple. You Sell when %R reaches 10% or lower and buy when it reaches 90% or higher. The %R works best in trending markets, either bull or bear trends. Likewise, it is not uncommon for divergence to occur between the %R and the market. It is just another hint of the market's condition.

9   List of signals

Indicator name/ Arguments	B/I Data Set	Explanation

10   Summery

	Function	Indicator	Signal
Calculated by time or by periods	Time. For each change in input a new value is calculated.	Periods. For all periods indicators calculate a value.	Periods. For all periods signals calculate a value.
Data Type	No change of data type. Cannot use daily high and low prices when calculated. Volume is not allowed.	Data type can be modified and more data types can be used.	More data types can be used.
Can call	Functions.	Functions and Indicators.	Functions and indicators.
Can be called by	Functions, indicators, signals, views, and instruments.	Indicators, signals, and views.	Views.
When to use	Functions are used to performs simple calculations where there are no need for specifying data type and to reference back. Functions are used when you do not want calculation on a given data type. Functions must be used if you want to create a calculated instrument.	Indicators are used to perform calculations that requires that data have been split in data types and periods.	Signals are used to calculate when to buy and when to sell in views.

Table 3. Summery.

Register

—A—

Abs;19
ADXR;21
assignment statement;9;16

—B—

Backus Naur Normal Form;15
BC VIEW Language Description;8
BNF;15
BOLLINGERdown;21
BOLLINGERmid;21
BOLLINGERup;21

—C—

calculating, functions;7
calculating, indicators;5
calculating, indicators and signals;5
calculating, signals;6
calculation of functions, indicators, and signals;4
CCI;21
CCIsmoothed;22
CorrelationCoef;22
Cos;19
cross;9
Cross;20
CrossM100;20

—D—

data extracts;13;19
data types;12
data types, converting;4
Divergensindicator;22
Dlm;22
Dlp;22
DMI;22

—E—

editing functions, indicators, or signals;3
EMS;23
EXIT LONG;6
EXIT SHORT;6
Exp;19
expression;17

—F—

Falling;23
Formal BCL Specification;15
Forward;23

—V—

—W—