



WHS FutureStation

Express Functions

www.whselfinvest.com

Contents

Introduction	5
Symbol information	8
SymbolName()	8
PointValue()	9
TickSize()	10
TickValue().....	11
PrevDayHigh() / Low() / Open() / Close().....	12
PrevDayVol()	13
Time information	14
DayOfWeek().....	14
Duration().....	15
IsNewDay().....	16
NumericToTime()	17
Alerting & Informing	18
MessageBox().....	18
PlaySound()	19
ShowTip()	20
Highlight() / HighlightRGB()	21
HighlightAT() / HighlightRGBAt()	23
NumericToString().....	24
PriceToString().....	25
TimeToString().....	26
IsBarCompleted()	27
Loops & Arrays	28
For ... to / For ...ownto	28
While ... do	30
SetArrayTo() / SetArraySize() / GetArraySize()	31
Indexing & Efficient programming.....	33
CurrentBarIndex() / FinalBarIndex()	33
IndexOfHighest() / IndexOfLowest()	34
CalculateAtEveryTick()	35
IsFirstBar / IsFinalBar and associated functions	37

IsFinalBar() / IsFirstBar()	37
MovingAverage()	39
ExpMovingAverage()	40
RSI()	41
StdDev()	42
Mathematical functions	43
AbsValue()	43
Sign()	44
floor() / ceiling()	45
Sum()	46
Atr()	47
AtrAbs()	48
Sine()	49
Cosine()	50
Tangent()	51
ArcTangent()	52
Exp()	53
Log()	54
Power()	55
SquareRoot()	56
Highest()	57
Lowest()	58
IsZero()	59
IsNonZero()	60
Max()	61
Min()	62
Round()	63
RoundMultiple()	64
NormalCDF()	65
NormalPDF()	66
Charting functions	67
Plotline()	67
Plot()	68
Plotband()	69

Plotcrossinglines()	70
Plotbars()	71
Plotcandles()	72
GetPriceFormat()	73
SetYscaleFormat()	74
Importing series	75
Importing a series from a platform built-in indicator	75
Importing a series from another express indicator	76
Importing a price series from a symbol	77
Creating customized stops and targets	78
SetIntraPeriodUpdate()	78
EntryPrice()	79
EntryPriceOriginal()	80
BarsSinceEntry()	81
IsIntradayEntry()	82
MarketPosition()	83
MinPriceEntryBar() / MaxPriceEntryBar()	84
SetLongTrigger() / SetShortTrigger()	85
SetStopPrice()	87
SetTargetPrice()	88
Predefined interpretation tools	89
CrossesAbove() / CrossesBelow()	89
CrossesAboveThreshold() / CrossesBelowThreshold()	90
Triggerline()	91
Swing()	92
Bands()	93
TwoThresholds()	94

Introduction

This manual offers an overview of the available functions in the Express Module, intended as an educational tool to aid a step by step learning process. The functions are logically classified by category and grouped together to reflect a close relation to one another.

For every function, described in this manual, an example is provided to illustrate its utility and execution. The examples can be copy-pasted from this manual into the Express editor (see the appendix to the introduction for details on p. 7). Please note that the examples are solely offered for evaluation purposes, and must not be considered an adequate strategy for trading.

Before starting programming in WHS FutureStation Nano using the Express Module, it is important to familiarize oneself with the platform's most basic principles. These principles are well documented and explained in our platform manuals, video clips, and many forums, which can all be accessed from our client website.

Our video clips, in particular, deserve special recommendation. There are 75 video clips illustrating general platform capabilities and 42 video clips describing programming procedures. The programming video clips are distinctly separated into a theoretical explanation on the one hand and a practical example on the other.

Below is a short summary of the categories covered in this manual:

Symbol Information / Time Information / Alerting & Informing

These three function groups are very closely linked as Symbol Information and Time Information contain information that can be visually/auditory presented using Alerting & Informing functions.

Not only will the user experience freedom from his platform, as he will be duly notified by an indicator or signal, so too will the user experience freedom of creativity, thanks to an almost endless number of possible applications and customizations (colors, messages, sounds, etc.).

Loops & Arrays

This section covers the most basic and fundamental principles of programming in Express. It is an absolute must-read, and should therefore not be neglected.

Array functions, on the other hand, are intended for more experienced users with very specific needs.

Indexing & Efficient Programming

The indexing functions allow the user to identify any bar in a range in order to execute a set of instructions on that particular bar.

A very nice feature that falls under this category is CalculateAtEveryTick(), which allows the user to save a tremendous amount of computing capacity by holding calculations until the end of a period, rather than re-calculating the entire series at every tick.

IsFirstBar / IsFinalBar and Associated Functions

These functions should be used to concentrate all the instructions that do not need to be repeated at every bar. For example, this is the reason why IsFirstBar/IsFinalBar is used in conjunction with MovingAverage and ExpMovingAverage.

Mathematical Functions

This section is a straight-forward recapitulation of mathematical tools incorporated in Express, which will likely bring back fond memories of one's high-school days and teachers.

Charting Functions

This is a very interesting function area, allowing the user to express his creativity in customizing chart representation, such as background, chart color and type.

Importing Series

Answers the common request of applying one series to another, e.g. applying an RSI to a Moving Average.

Creating Customized Stops and Targets

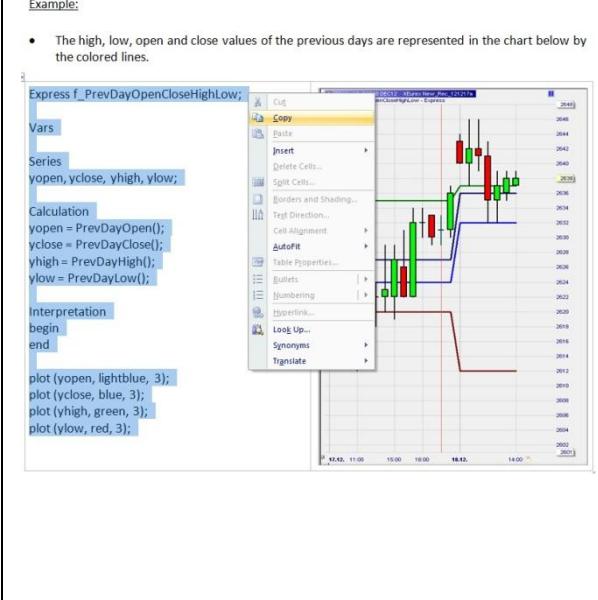
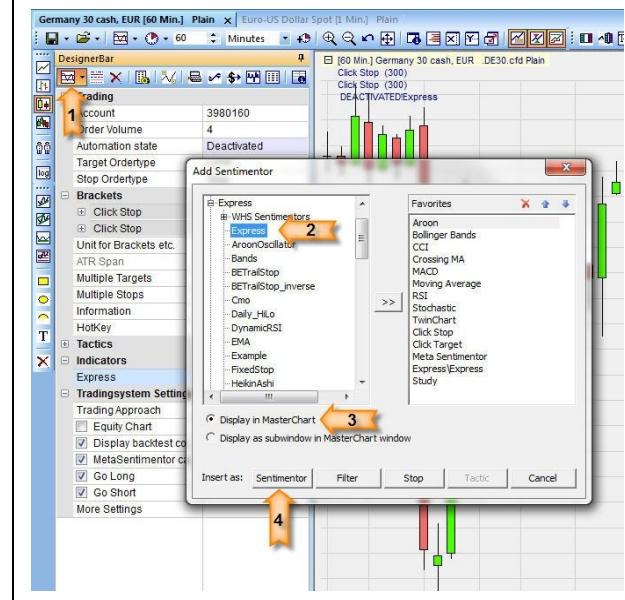
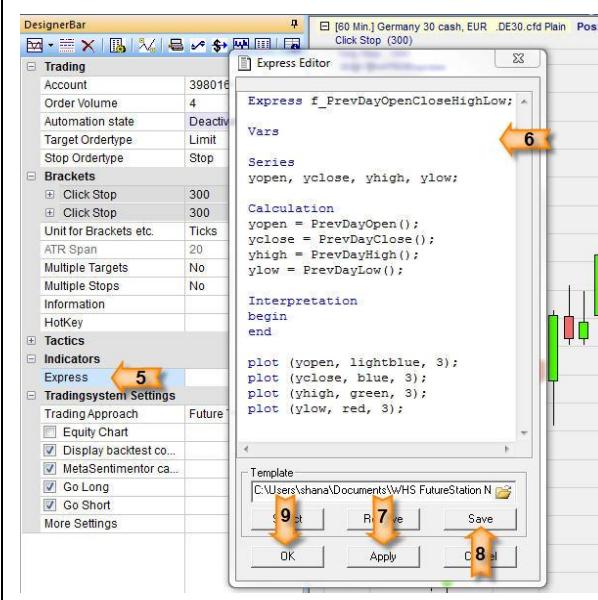
Another area where trading experience meets creativity. This function group allows the user to refine stops and targets to perfect a personalized strategy.

Predefined Interpretation Tools

This section covers an explanation of the predefined swing tools, designed to spare the user from lengthy programs.

Appendix to the Introduction

How to copy an express code from the manual and add it the express editor for display.

<p><u>Example:</u></p> <ul style="list-style-type: none"> The high, low, open and close values of the previous days are represented in the chart below by the colored lines.  <pre> Express f_PrevDayOpenCloseHighLow; Vars Series yopen, yclose, yhigh, ylow; Calculation yopen = PrevDayOpen(); yclose = PrevDayClose(); yhigh = PrevDayHigh(); ylow = PrevDayLow(); Interpretation begin end plot (yopen, lightblue, 3); plot (yclose, blue, 3); plot (yhigh, green, 3); plot (ylow, red, 3); </pre>	
<p>Select the text from the manual's example (Ctrl + A), right-click, and copy (or Ctrl + C)</p>	<ol style="list-style-type: none"> Click Add an Indicator Select Express and Express again Click Display in Master Chart or in a sub window Insert as Sentimentor
 <pre> DesignerBar Trading Account 3980160 Order Volume 4 Automation state Deactivated Stop Ordertype Stop Target Ordertype Limit Stop Ordertype Stop Brackets Click Stop 300 Click Stop 300 Unit for Brackets etc. Ticks ATR Span 20 Multiple Targets No Multiple Stops No Information HotKey Tactics Indicators Express TradingSystem Settings Trading Approach Future Equity Chart Display backtest co... MetaSentimentor ca... Go Long Go Short More Settings </pre> <pre> Express f_PrevDayOpenCloseHighLow; Vars Series yopen, yclose, yhigh, ylow; Calculation yopen = PrevDayOpen(); yclose = PrevDayClose(); yhigh = PrevDayHigh(); ylow = PrevDayLow(); Interpretation begin end plot (yopen, lightblue, 3); plot (yclose, blue, 3); plot (yhigh, green, 3); plot (ylow, red, 3); </pre>	
<ol style="list-style-type: none"> Double-click Express Ctrl + V to paste the selected text Click Apply Click Save Click OK 	<p>Mission Accomplished!!</p>

Symbol information

SymbolName()

Definition:

- Returns the name of the symbol used in a study.

Format:

- string SymbolName()

Example:

- Below we generate a message each time the condition (close > close[1]) and (close > close[2]) is verified. The message is made of two parts, a piece of text and the name of the symbol:

```
Express f_SymbolName
```

```
Calculation
```

```
if (close > close[1]) and  
  (close > close[2]) then MessageBox(  
    "Possible trend detected"  
  +  
    SymbolName());
```

```
Interpretation
```

```
begin  
end
```



PointValue()

Definition:

- Shows the monetary value of one point of a given instrument.

Format:

- `Float PointValue()`

Example:

- Displays the monetary values of a single point in DAX and MINI S&P futures: EUR 25 and USD 50 respectively.

```
Express f_PointValue
```

```
vars
```

```
Series a;
```

```
calculation
```

```
a = PointValue();
```

```
interpretation
```

```
begin
```

```
end
```

```
plot(a, blue, 2);
```



TickSize()

Definition:

- Returns the tick size of the instrument.

Format:

- `Float TickSize()`

Example:

- Indicator `a` displays the number of ticks per bar for a given instrument.

```
Express f_TickSize
```

```
Vars
```

```
Series
```

```
a;
```

```
Calculation
```

```
a = (h-l)/TickSize();
```

```
Interpretation
```

```
begin
```

```
end
```

```
plot (a, blue, 2);
```



TickCount()

Definition:

- Returns the monetary value of one tick for a given instrument.

Format:

- `Float TickValue ()`

Example:

- Indicator a displays the monetary values of the daily price variations of a given instrument.

```
Express f_TickValue;
```

```
Vars
```

```
Series
```

```
a;
```

```
Calculation
```

```
a = (h - l)/TickSize()*TickValue();
```

```
Interpretation
```

```
begin
```

```
end
```

```
plot(a, blue, 2);
```



PrevDayHigh() / Low() / Open() / Close()

Definition:

- The functions below
 - PrevDayHigh()
 - PrevDayLow()
 - PrevDayOpen()
 - PrevDayClose()return the high, low, open and close values of the previous day. In case the data is not available the value returned is void.

Format:

- `Float PrevDayHigh() ...`

Example:

- The high, low, open and close values of the previous days are represented in the chart below by the colored lines.

```
Express f_PrevDayOpenCloseHighLow;
```

Vars

Series

```
yopen, yclose, yhigh, ylow;
```

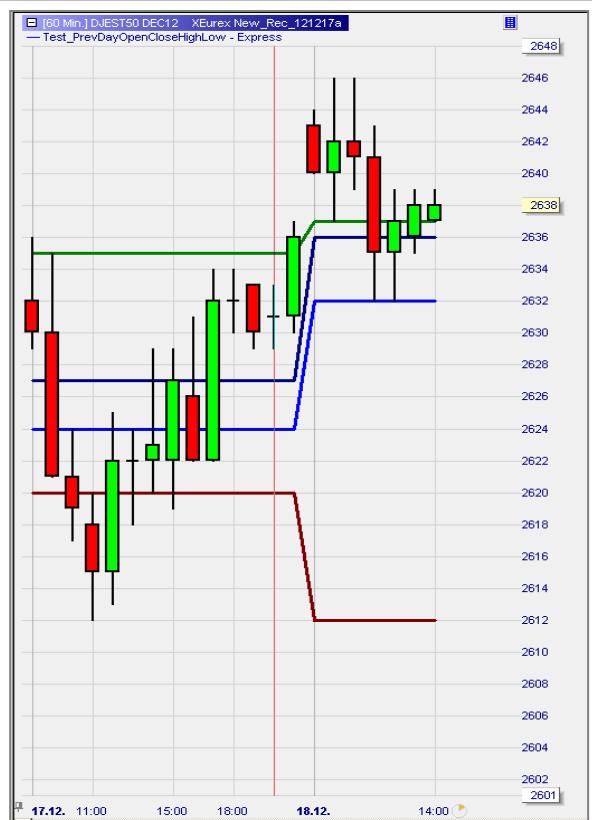
Calculation

```
yopen = PrevDayOpen();  
yclose = PrevDayClose();  
yhigh = PrevDayHigh();  
ylow = PrevDayLow();
```

Interpretation

```
begin  
end
```

```
plot (yopen, lightblue, 3);  
plot (yclose, blue, 3);  
plot (yhigh, green, 3);  
plot (ylow, red, 3);
```



PrevDayVol()

Definition:

- Returns the volume of the previous day or void in case the data is not available.

Format:

- `Float PrevDayVol ()`

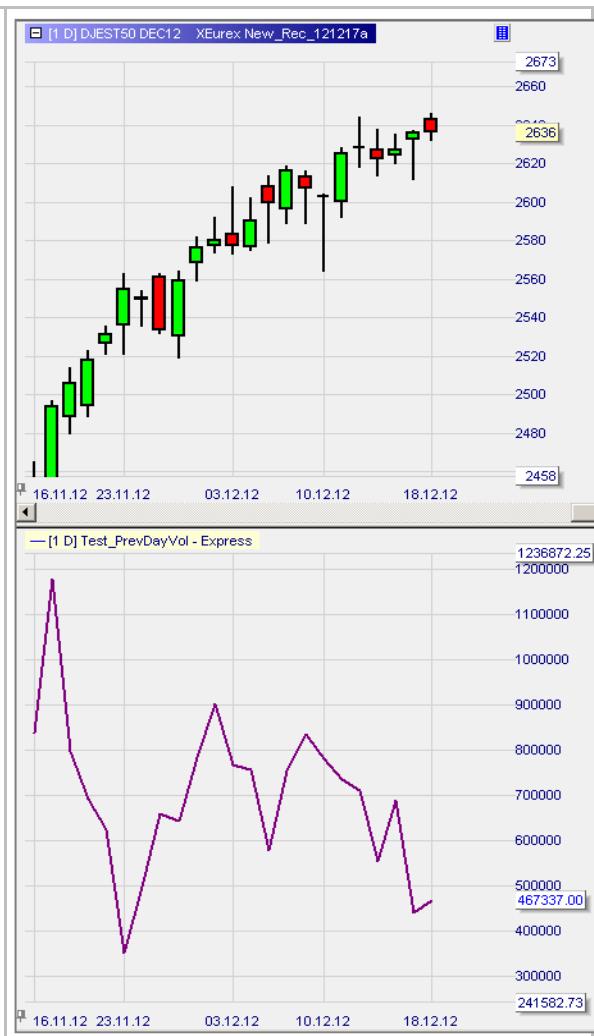
Example:

- The value of the volume of the previous days is displayed in the sub window:

```
Express f_PrevDayVol

Vars
Series
yvolume;
Calculation
yvolume = PrevDayVol();
Interpretation
begin
end

plot (yvolume, lightmagenta, 2);
```



Time information

DayOfWeek()

Definition:

- Returns the day of the week expressed as an integer between 1 and 5:
 - 0 = Sunday, 1 = Monday, 2 = Tuesdays, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday.

Format:

- int DayOfWeek (time time)

Example:

```
Express f_DayOfWeek
```

```
vars
```

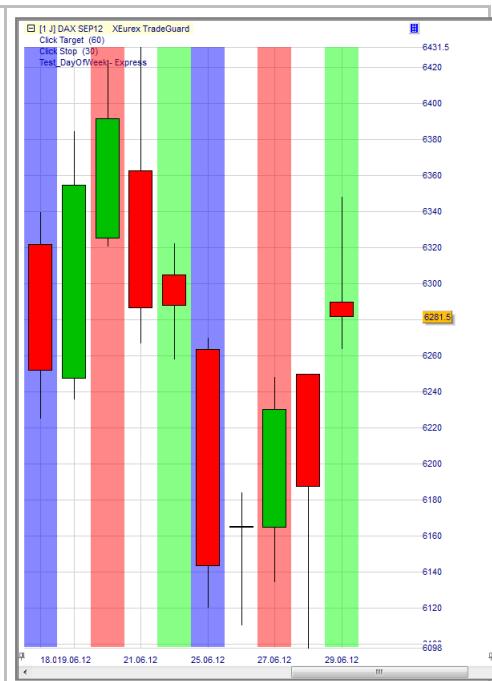
```
calculation
```

```
if DayOfWeek(date) = 1 then Highlight("slot","lightblue");
if DayOfWeek(date) = 3 then Highlight("slot","lightred");
if DayOfWeek(date) = 5 then Highlight("slot","lightgreen");
```

```
interpretation
```

```
begin
```

```
end
```



Duration()

Definition:

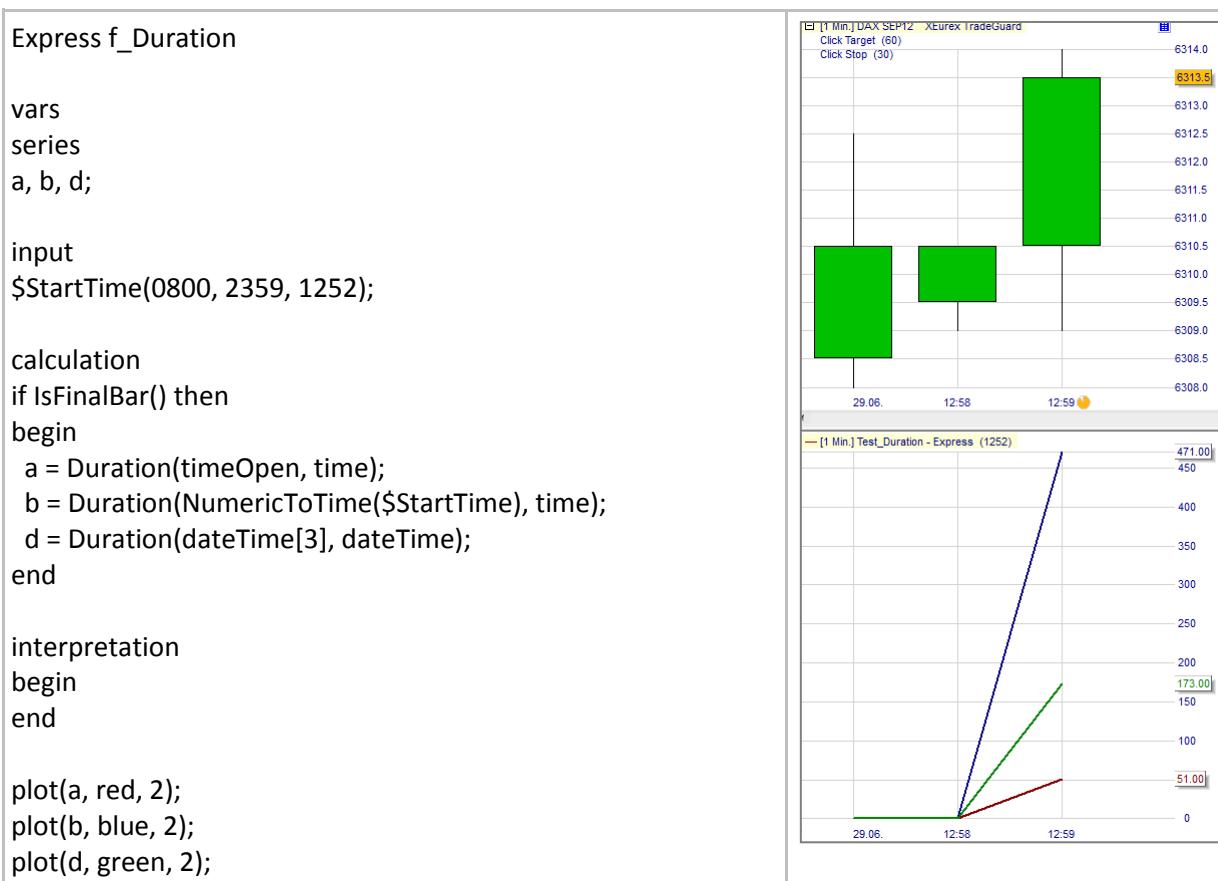
- Returns the number of seconds between time t1 and time t2¹.

Format:

- int Duration (time start, time end)

Example:

- Below we are looking at the last candle of a 1min chart on the Dax future:
 - Open time is 12:59:00.
 - Current time is 12:59:51.
- Our indicator displays three durations in seconds between different times:
 - Red: Duration between current and open times = 51.
 - Blue: Duration between current time and \$StartTime (= 12:52) = 471².
 - Green: Duration between current “date time” now and 3 candles ago: 173³.



¹ Express predefined time series are: date, dateOpen, time, timeOpen, dateTime, dateTimeOpen.

² 471 seconds = 7 x 60 + 0.85 x 60 seconds = 7 minutes + 51 seconds. Hence current time is: 12:52:00 + 00:07:51 = 12:59:51.

³ 173 seconds = 2 x 60 + 51 seconds.

IsNewDay()

Definition:

- Returns true at the first bar of each day.

Format:

- bool IsNewDay()

Example:

- Below the background of the first candle of each day is highlighted in magenta:

```
Express f_IsNewDay
```

```
calculation
```

```
if IsNewDay() then Highlight("slot", "magenta");
```

```
interpretation
```

```
begin
```

```
end
```



NumericToTime()

Definition:

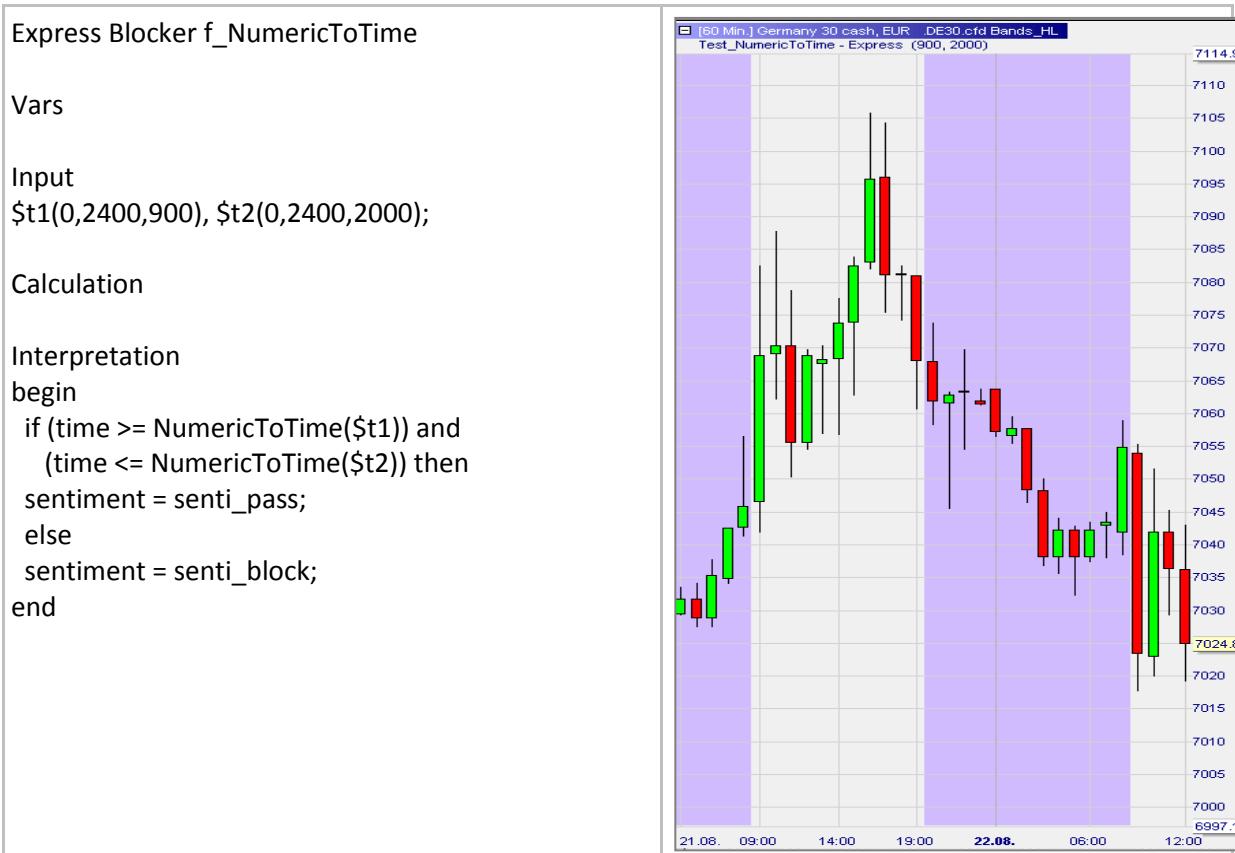
- Converts a 4 digits number into a HHMM time value (e.g. 1545 becomes 15:45).
 - If the first 2 digits are more than 23, HH is set to 23.
 - If the second 2 digits are more than 59, HH is set to 59.

Format:

- time NumericToTime (float value)

Example:

- Below we create a blocker which blocks trades outside the 09:00 – 20:00 interval as indicated by the darker background color:



Alerting & Informing

MessageBox()

Definition:

- Opens a pop-up window with a pre-defined message when a particular condition is met using a live data stream. Without live data there can be no messages.
 - A message box shows up once for every occurrence. For example, several message boxes can be prompted within the same bar.
 - To write 2 lines in a message use this instruction:
 - MessageBox("Line 1 \n Line 2");

Format:

- void MessageBox(string message)

Example:

- Below we are generating messages every time the price crosses the blue moving average. Messages may appear several times within the same bar.

Express f_MessageBox	
Vars	
Series	
myMA;	
Input	
\$span(1, 200, 30);	
Calculation	
If IsFirstBar() then MovingAverage(close, myMA, \$span);	
if CrossesAbove(close, myMA) then	
MessageBox("Price > Moving Average");	
if CrossesBelow(close, myMA) then	
MessageBox("Price < Moving Average");	
Interpretation	
begin	
end	
plot (myMA, blue, 3);	

PlaySound()

Definition:

- Plays a pre-defined sound when a particular condition is met using a live data stream. Without live data there can be no audible notifications.
- A sound is triggered once per occurrence and per bar. For example:
 - Several sounds can be prompted within the same bar.
 - To play a sound once at the end of a period use the following instruction:
if IsBarCompleted() and then PlaySound("gong");
- The sound files are available at the following folder: WHS FutureStation Nano\Wav
- If several PlaySound() functions are triggered at the same time only the first PlaySound() function in the code will be executed.

Format:

- void PlaySound(string sound)

Example:

- Here a sound is triggered every time three bullish bars occur consecutively.

```
Express f_PlaySound
vars
calculation
if (c > c[1]) and (c[1] > c[2]) and (c[2] > c[3])
then begin
    Highlightat(0, "slot","green");
    Highlightat(1, "slot","green");
    Highlightat(2, "slot","green");
    MessageBox("3 Bullish Bars - Possible
Trend");
    Playsound("corkpop");
end
Interpretation
begin
end
```

ShowTip()

Definition:

- Displays a message when the mouse is moved over a candle.
 - The message pops up over the candle after a couple of seconds.

Format:

- void ShowTip (string message)

Example:

- The tip “test1” was displayed because the price closed below 6480 points.

```
Express f_ShowTip
```

```
vars
```

```
calculation
```

```
if c < 6480 then Showtip("test1");  
if c > 6480 then Showtip("test2");
```

```
interpretation
```

```
begin
```

```
end
```



Highlight() / HighlightRGB()

Definition:

- Both functions enable the user to highlight the current bar using a specific marker and color.
- The available markers are: "ellipse", "upTriangle", "downTriangle", "slot", "bottomLine", "topLine", "textAbove" and "textBelow".
- The following colors are available for the Highlight() function: "red", "lightRed", "green", "lightGreen", "blue", "lightBlue", "magenta", "lightMagenta", "yellow", "lightYellow", "cyan", "lightCyan", "grey", "black", "white".
- For the HighlightRGB() function, an almost infinite number of colors can be selected using the RGB color scheme (RGB = Red-Green-Blue).
- Refer to the section on Plotting functions to learn how to create a RGB color code.

Format:

- Void Highlight (string type, string color)
- Void HighlightRGB (string type, int red, int green, int blue)

Example 1:

- Highlights the 50th candle of the chart by coloring the background behind the bar in a lightgreen color.

```
Express f_Highlight
```

```
vars
```

```
calculation
```

```
if CurrentBarIndex() = 50 then  
Highlight("slot", "lightgreen");
```

```
interpretation
```

```
begin  
end
```



Example 2:

- Here we create a box using `highlight(textbelow,"black")` with a message whose information gets updated tick by tick. Note some very useful rules for creating content:
 - Pieces of texts must be surrounded with quotes: "Instrument: "
 - Numbers must be converted in a text format using `NumericToString()`.
 - String variable can be typed without quotes: `contract`
 - To go to the next line use the expression: "\n"
 - All pieces must be joined with a + sign.

```
express f_Highlight2

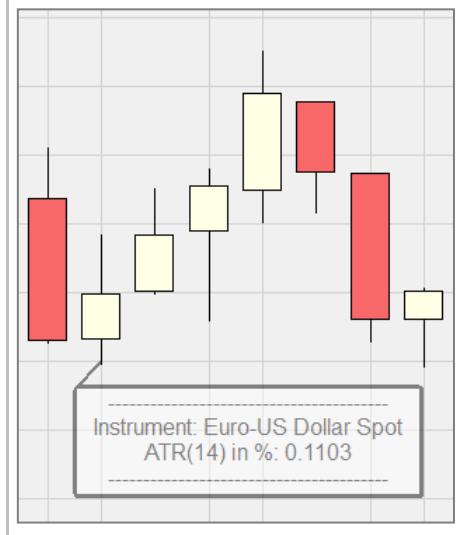
vars

input
$period(1,100,14);

string
contract;

calculation

if (CurrentBarIndex()= (FinalBarIndex() - 20)) then
begin
    contract = SymbolName();
    HighLight("textbelow:"
    + "-----"
    + "\n" + "Instrument: " + contract
    + "\n" + "ATR" + "(" + NumericToString($period,"") + ") in %: " +
    NumericToString(atr($period),"%2.2f")
    + "\n" + "-----"
    , "black");
end
interpretation
begin
end
```



HighlightAT() / HighlightRGBat()

Definition:

- Both functions enable the user to highlight a given bar using a specific marker and color. The given bar refers to a bar that is x bars away from the current bar. For example:
 - HighlightAT(0,"slot","blue"): the given bar is the current bar.
 - HighlightAT(10,"slot","blue"): the given bar is the 10th bar to the left of the current bar.
 - HighlightAT(-5,"slot","blue"): the given bar is the 5th bar to the right of the current bar.
- The available markers are: "ellipse", "upTriangle", "downTriangle", "slot", "bottomLine", "topLine", "textAbove" and "textBelow".
- The following colors are available for the HighlightAT() function: "red", "blue", "green", "yellow", "black", "grey", "white", "magenta", "lightred", lightblue", "lightgreen", "lightyellow", "black", "lightgrey", "white" and "lightmagenta".
- For the HighlightRGBAt() function, an almost infinite number of colors can be selected using the RGB color scheme (RGB = Red-Green-Blue).
- Refer to the section on Plotting functions to learn how to create a RGB color code.

Format:

- Void HighlightAt (string type, string color)
- Void HighlightRGBAt (int offset, string type, int red, int green, int blue)

Example:

- Four applications of the HighlightAT function are displayed:

```
Express f_HighlightAT
```

```
Calculation
```

```
If IsFirstBar() then
    Highlightat(-3,"slot","magenta");

If IsFinalBar() then
begin
    Highlightat(0,"slot","lightred");
    Highlightat(1,"slot","blue");
    Highlightat(5,"ellipse","lightgreen");
end
```

```
Interpretation
begin
end
```



NumericToString()

Definition:

- Applies to the elements of a series a given format and convert them into a string.
 - NumericToString applies the "%g" format by default.
 - Supports all formats used for the C-function "printf()". The most important are:
 - "%f" decimal floating point
 - "%6.2f" rounds to two decimals
 - "%g" discards trailing zeroes
 - "%e" scientific notation

Format:

- string NumericToString (float value, string format)

Example:

- Below we display the format of the close price to be shown using the ShowTip function.
 - For example by selecting type 4 we show the price in scientific notation:

```
Express f_NumericToString

vars

input
$type(0,4,0);

calculation

// Rounds to three decimals
if $type = 1 then
ShowTip(NumericToString(c, "%6.3f"));
else
// Decimal floating point
if $type = 2 then
ShowTip(NumericToString(c, "%f"));
else
// Discards trailing zeroes
if $type = 3 then
ShowTip(NumericToString(c, "%g"));
else
// Scientific notation
if $type = 4 then
ShowTip(NumericToString(c, "%e"));

interpretation
begin
end
```



PriceToString()

Definition:

- Rounds the price to which it applies to the nearest price in respect to the defined ticksize and precision of the analyzed symbol and converts it into a string. Takes fractional notations into account.

Format:

- string PriceToString (float value)

Example:

- Below we display a piece of information through a ShowTip.
 - It includes a title “Average price” followed at the next line by the average price value.
 - The format is a rational number rounded to the nearest ticksize and displayed with one decimal.

Example:

```
Express f_PriceToString
```

```
calculation
```

```
ShowTip("Average price: \n" + PriceToString((o + h + l + 2*c)/5));
```

```
interpretation
```

```
begin
```

```
end
```



TimeToString()

Definition:

- Returns a time variable using one of the formats below:

%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month as decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01 – 12)
%M	Minute as decimal number (00 – 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00 – 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z, %Z	Time-zone name or abbreviation; no characters if time zone is unknown
%%	Percent sign

Format:

- string TimeToString (time timeVal, string format)

Example:

- Below we create a tip which shows the time when we move the mouse over a candle:

Express f_TimeToString

Calculation

```
showtip(TimeToString(time, "%H:%M:%S"));
```

```
interpretation  
begin  
end
```



IsBarCompleted()

Definition:

- Returns true once the current period is completed.

Format:

- bool IsBarCompleted()

Example 1:

- If IsBarCompleted() and (volume > 1000) then PlaySound("gong");

Example 2:

- Below we launch a message every time the bar is completed:
 - The window containing the message will accumulate in the chart.

Express f_IsBarCompleted

Calculation

```
If IsBarCompleted() then  
MessageBox("A new period has begun");
```

Interpretation

```
begin  
end
```



Loops & Arrays

For ... to / For ... downto

Definition:

- Enables the execution of a loop in a program (see Syntax below).
 - A loop executes a given block of codes for a given number of times.

Syntax:

- ```
for <variable> = <start value> to <numerical expression>
begin
 <statement>;
 <statement>;
 ...
 <statement>;
end
```

### Example 1:

- Below we use For ... loop to calculate the average of the last 10 bars, bar by bar. Please note:
  - The loop is run at each bar to add up the last 10 close prices. We make i vary between 0 and 9 and we execute sum = sum + close[i].
    - i = 0 is the current bar, i = 1 is the first bar on the left, i = 2 is the second bar on the left, .. etc.

```
Express f_ForLoop_1
```

```
Vars
```

```
Numeric i;
Series avgsum, sum;
```

```
Calculation
```

```
Calculateateverytick(false);
for i = 0 to 9
begin
 sum = sum + close[i];
end
avgsum = (sum/10);
```

```
Interpretation
```

```
begin
end
```

```
plot(avgsum, magenta, 3);
```



## Example 2:

- Below we create a MACD indicator using a loop to calculate the difference between two exponential moving averages. Please note:
  - Here we do all the calculations at the first bar. Exponential moving averages are easily computed using an express internal function.
  - The MACD is computed using a loop. We make i vary between 0 and FinalBarIndex.
    - We do not use 'begin ... end' because there is only one expression.
    - Being at the first bar: i = 0 is the first bar, -1 is the second bar on the right, -2 is the third bar on the right, ... etc.
    - Note: We could also have used the following instruction:  
For i = 0 downto (- FinalBarIndex())

Express f\_ForLoop\_2

Vars

Series

ema1, ema2, macd, emacd, FBI;

numeric

i;

Calculation

CalculateAtEveryTick(false);

if IsFirstBar() then

begin

ExpMovingAverage(close, ema1, 12);

ExpMovingAverage(close, ema2, 26);

for i = 0 to FinalBarIndex() macd[-i] = ema1[-i] - ema2[-i];

ExpMovingAverage(MACD, eMACD, 9);

end

Interpretation

begin

end

plot(eMACD, blue, 3);

plot(MACD, red, 3);



## While ... do

### Definition:

- Enables a portion of a program to be executed several times consecutively as long as a given condition remains true (see Syntax below).

### Syntax:

```
• while <boolean expression>
begin
 <statement>;
 <statement>;
 ...
 <statement>;
end
```

### Example:

- Below we count how many times the price closes consecutively higher (green line) and lower (red line) and record the results bar by bar.

```
express f_WhileDo;

vars
series longPeriods, shortPeriods;
numeric counter;

calculation
counter = 1;
longPeriods = 0;
while (c[counter-1] >= c[counter]) and (c[counter] <> void)
begin
 longPeriods = longPeriods + 1;
 counter = counter +1;
end
counter = 1;
shortPeriods = 0;
while (c[counter-1] <= c[counter]) and (c[counter] <> void)
begin
 shortPeriods = shortPeriods + 1;
 counter = counter +1;
end

interpretation
begin
end

plot(longPeriods, green, 2);
plot(shortPeriods, red, 2);
```



## **SetArrayTo() / SetArraySize() / GetArraySize()**

### Definitions:

- An **array** is a set of a given number of elements used to store numbers.
  - $a[0] = 12, a[1] = 45, a[2] = -1, a[3] = -4/5$  are the 4 elements of an array called a.
  - To define array abc containing 4 elements we write under section Vars:  
**Array abc[3];**
- Elements of an array are set by default to zero. To affect 500 to all elements we write:
  - **SetArrayTo(abc, 500);**
- Array abc may be resized to 250 elements by writing:
  - **SetArraySize(abc, 250);**
- To get the size of array abc we write:
  - **GetArraySize(abc)**
- To attribute a number to a given element of array abc we write:
  - First element:  $abc[0] = 10;$
  - Second element:  $abc[1] = -7;$
  - Element i:  $abc[i-1] = -5;$

### Format:

- **void SetArrayTo (array arr, float value)**
- **void SetArraySize (array arr, int size)**
- **int GetArraySize (array arr)**

### Example:

- The indicator below displays the elements of an array.
  - At the final bar we display the elements of the array in a blue series ShowResults.
  - We modify the elements and display the new elements in a red series ShowResults2.

```

Express f_Array;

vars
input
$AbcSize(1, 100, 10), $AbcValue(1, 100, 15);

array
abc[0];

series
ShowResults, ShowResults2;

numeric
i;

calculation

if IsFinalBar() then
begin
SetArraySize(abc, $AbcSize);
SetArrayTo(abc, $AbcValue);
for i=0 to GetArraySize(abc)-1
begin
ShowResults[i] = abc[i];
abc[i] = i * power(-1,i);
ShowResults2[i] = abc[i];
end
end

interpretation
begin
end

plot(ShowResults, blue, 2);
plot(ShowResults2, red, 2);

```



## Indexing & Efficient programming

### CurrentBarIndex() / FinalBarIndex()

#### Definition:

- For a given chart FinalBarIndex() indicates the number of bars displayed on the chart minus one.  
For example:
  - The index of the first bar on the left of a chart is 0
  - The index of the last or final bar on the right of a chart is 99
  - If a chart displays 1000 bars, FinalBarIndex() = 999
- CurrentBarIndex() indicates the index number of a bar.  
For example:
  - When considering the first bar CurrentBarIndex() = 0
  - When considering the second bar CurrentBarIndex() = 1
  - When considering the 10th bar CurrentBarIndex() = 9
  - When considering the nth bar CurrentBarIndex() = n - 1

#### Format:

- int CurrentBarIndex()

#### Example:

- CurrentBarIndex() is assigned to series x and FinalBarIndex() is assigned to series y. These two series are plotted underneath the main chart:
  - CurrentBarIndex() is the green straight line starting from 0 up to 148
  - FinalBarIndex() is the red horizontal line at 148
  - Presently there are 149 (= 148 + 1) bars in this chart

Express f\_CurrentBarIndex

vars

series x,y;

calculation

```
x = CurrentBarIndex();
y = FinalBarIndex();
```

interpretation

begin

end

```
plot (x, green, 2);
plot (y, lightred, 2);
```



## IndexOfHighest() / IndexOfLowest()

### Definition:

- Returns the index of the highest / lowest value for the elements series[0], ... series[span - 1].

### Format:

- int IndexOfHighest (series series, int span)

### Example:

- Below we first color the background of the last 10 candles in grey.
- We then identify the highest and the lowest bars of the last 10 candles by coloring in green or red the background of the chart. For that purpose we need to indicate the index of the bars to the function HighlightAT:

```
Express f_IndexOfHighest
```

```
vars
```

```
input $period(1,50,10);
```

```
calculation
```

```
if (CurrentBarIndex() > (FinalBarIndex() - $period))
then Highlight("slot","grey");
```

```
if IsFinalBar() then
```

```
begin
```

```
for i = 0 to ($period - 1)
```

```
begin
```

```
if ((FinalBarIndex() - i) =
IndexOfHighest(h,$period)) then
HighlightAT(i,"slot","lightgreen");
```

```
if ((FinalBarIndex() - i) =
IndexOfLowest(l,$period)) then
HighlightAT(i,"slot","lightred");
```

```
end
```

```
end
```

```
interpretation
```

```
begin
```

```
end
```



## CalculateAtEveryTick()

### Definition:

- This function can be used in two modes:
  - With **CalculateAtEveryTick(false)** an express code is calculated once at the close of the bar.
    - This instruction is used when the result of a program is only required at the close of the bar or when the user wants to limit the number of iterations and CPU usage.
  - With **CalculateAtEveryTick(true)** an express code is calculated at every tick. When this instruction is not present in a code the program is executed by default at every tick.
    - Users should realize that express codes are executed every time a new tick is received by the platform. As long as the number of bars and the number of iterations are limited this instruction can be neglected.

### Example 1:

- Here we calculate at every tick.
- The blue line plotted below passes through the close prices. As the close price of the last candle gets updated at each incoming tick, so too does the blue line: The blue line moves in line with the close price.

```
Express f_CalculateAtEveryTick

vars
series x;

calculation
x = c;

interpretation
begin
end

plot(x,blue,2);
```



### Example 2:

- Here we calculate only at the close of the bar.
- Contrary to above here the blue line gets updated only once at the close of each bar: The blue line is fixed, positioned on the previous close, while the close price keeps going down.

### Express f\_CalculateAtEveryTick

```
vars
series x;

calculation

CalculateAtEveryTick(false);
x = C;

interpretation
begin
end

plot(x,blue,2);
```



## IsFirstBar / IsFinalBar and associated functions

### IsFinalBar() / IsFirstBar()

#### Definition:

- Returns true if at the first bar.
  - IsFirstBar() is used with the If ... then expression to identify the first bar on the left of the chart.
- Returns true if at the final bar.
  - IsFinalBar() is used with the If ... then expression to identify the last bar on the right of the chart.
  - These are very useful functions which can be used to reduce the number of iterations in express codes. These functions ought to systematically be used in conjunction with the following express functions: MovingAverage, ExpMovingAverage, StdDev and RSI.

#### Format:

- bool IsFinalBar()
- bool IsFirstBar()

#### Example 1:

- Here we need to calculate a constant numeric variable which will be used at every successive bar. The constant K equals the opening price of the first bar. The result, series x, is the difference between the close price and the opening price of the first bar.

#### Express f\_IsFirstBar\_example1

```
vars
series x, zero;
numeric k;

calculation

if IsFirstBar() then
begin
 k = o;
end

x = c - k;

interpretation
begin
end

plot(x, blue, 2);
plot(zero, red, 2);
```



## Example 2:

- Here we would like to calculate and plot the MACD line.
- We first have to calculate an exponential moving average (9) and an exponential moving average (26). Their difference resulting in the MACD line.
- Exponential moving average (9) and exponential moving average (26) are calculated at the first bar using the ExpMovingAverage function.
  - Why aren't they calculated at every bar? Because like this we eliminate unnecessary calculations:
    - The ExpMovingAverage function produces series ema based on series close. Let's imagine that our chart displays 1000 bars. Series ema is first obtained after the first execution of the ExpMovingAverage function at the first bar. If we permitted the execution of the ExpMovingAverage at every other bar we would just produce 999 times the same result as what we obtained at the first bar. Hence we impose a onetime only execution of the ExpMovingAverage function. We are actually being more efficient using fewer resources.

```
Express f_IsFirstBar_example2
```

```
vars
```

```
series ema1, ema2, macd, zero;
input $period1(1, 50, 9);
input $period2(1, 100, 26);
```

```
calculation
```

```
if IsFirstBar() then
begin
 ExpMovingAverage(c, ema1, $period1);
 ExpMovingAverage(c, ema2, $period2);
end
```

```
macd = ema1 - ema2;
```

```
interpretation
```

```
begin
end
```

```
plot(macd, blue, 2);
plot(zero, black, 1);
```



## MovingAverage()

### Definition:

- Returns the moving average of a series over a given period of time.
  - This function should only be used in combination with IsFirstBar() or IsFinalBar() in order to save a tremendous amount of computation (see IsFirstBar or IsFinalBar).

### Format:

- MovingAverage(series source, series target, int span)

### Example:

- Below we draw two moving averages MA1 and MA2.
- Note: Since series MA1 and MA2 are calculated at the first bar we do not execute the MovingAverage function at other bars (see IsFirstBar and IsFinalBar for more details).

```
Express f_MovingAverage
```

```
Vars
```

```
Series
```

```
MA1, MA2;
```

```
Input
```

```
$spanMA1(1, 200, 8), $spanMA2(1, 200, 30);
```

```
Calculation
```

```
If IsFirstBar() then begin
```

```
 MovingAverage(close, MA1, $spanMA1);
```

```
 MovingAverage(close, MA2, $spanMA2);
```

```
end
```

```
Interpretation
```

```
begin
```

```
end
```

```
plot (MA1, red, 3);
```

```
plot (MA2, green, 3);
```



## ExpMovingAverage()

### Definition:

- Returns the exponential moving average of a series over a given period of time.
  - This function should only be used in combination with IsFirstBar() or IsFinalBar() in order to save a tremendous amount of computation (see IsFirstBar and IsFinalBar).

### Format:

- ExpMovingAverage (series source, series target, int span)

### Example:

- Below we draw two exponential moving averages MA1 and MA2.
- Note: Since series MA1 and MA2 are calculated at the first bar we do not execute the ExpMovingAverage function at other bars (see IsFirstBar and IsFinalBar for more details).

```
Express f_ExpMovingAverage
```

```
Vars
```

```
Series
```

```
MA1, MA2;
```

```
Input
```

```
$spanMA1(1, 200, 8), $spanMA2(1, 200, 30);
```

```
Calculation
```

```
If IsFirstBar() then
```

```
begin
```

```
 ExpMovingAverage(close, MA1, $spanMA1);
```

```
 ExpMovingAverage(close, MA2, $spanMA2);
```

```
end
```

```
Interpretation
```

```
begin
```

```
end
```

```
plot (MA1, red, 2);
```

```
plot (MA2, green, 2);
```



## RSI()

### Definition:

- Returns the relative strength index of a series.
  - The function should only be used in combination with IsFirstBar() or IsFinalBar() in order to save a tremendous amount of computation (see IsFirstBar or IsFinalBar).

### Format:

- RSI(series source, series target, int span)

### Example:

- Below we draw a smoothed RSI.
- Note: Since series RSI and smoothedRSI are calculated at the first bar we do not execute the RSI and MovingAverage functions at other bars (see IsFirstBar and IsFinalBar for more details).

```
Express f_RSI
```

```
Vars
```

```
Series
```

```
myRSI, smoothedRSI;
```

```
Input
```

```
$a(1,100,14), $b(1,100,3);
```

```
Calculation
```

```
If IsFirstBar() then
```

```
begin
```

```
 RSI(close, myRSI, $a);
```

```
 MovingAverage(myRSI, smoothedRSI, $b);
```

```
end
```

```
Interpretation
```

```
begin
```

```
end
```

```
plot(myRSI, magenta, 2);
plot(smoothedRSI, blue, 2);
plotline(0, grey, 1);
plotline(20, grey, 1);
plotline(80, grey, 1);
plotline(100, grey, 1);
```



## StdDev()

### Definition:

- Returns the standard deviation of a series for a given period.
  - This function should only be used in combination with IsFirstBar() or IsFinalBar() in order to save a tremendous amount of computation (see IsFirstBar or IsFinalBar).

### Format:

- void StdDev (series source, series target, int span)

### Example:

- Below we draw the Bollinger Bands which are based on a 20 periods moving average plus and minus two times the standard deviation.
- Note: Since series RSI and smoothedRSI are calculated at the first bar we do not execute the RSI and MovingAverage functions at other bars (see IsFirstBar and IsFinalBar for more details).



## Mathematical functions

### AbsValue()

Definition:

- Returns the absolute value of a number.

Format:

- float AbsValue (float value)

Example 1:

- $\text{AbsValue}(5) = \text{AbsValue}(-5) = 5$

Example 2:

- Below we display the absolute value of the difference in points between the open and close prices of every candle.

Express f\_AbsValue

Vars

Series  
span;

Calculation  
span = AbsValue(open-close);

Interpretation  
begin  
end

plot (span, green, 3);



## Sign()

### Definition:

- Returns the sign of a given number.
  - The sign of a negative number is -1.
  - The sign of a positive number is 1.
  - The sign of zero is 0.

### Format:

- Sign(float value).

### Example:

- Below we draw a curve Y in red and another one in green which shows the opposite sign of the Y:

Express f\_Sign

Vars

Series

Y, OppositeSignY;

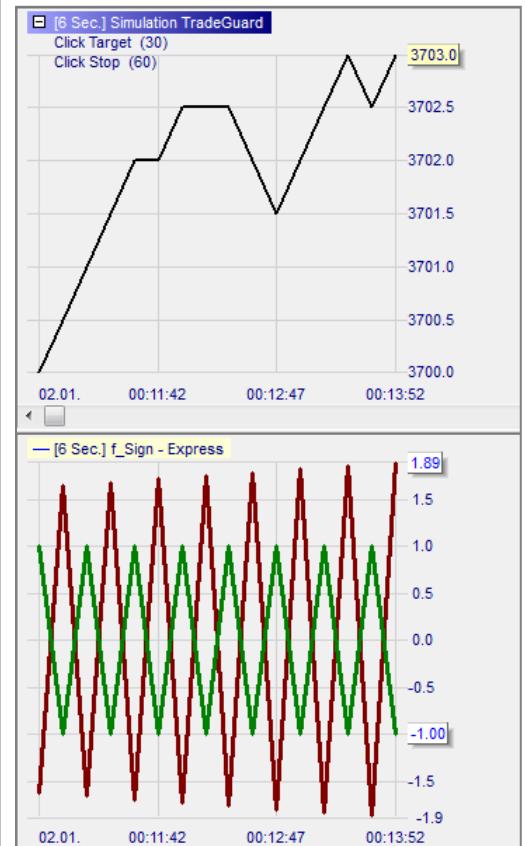
Calculation

```
Y = Power(-1.01, CurrentBarIndex());
OppositeSignY = -Sign(Y);
```

Interpretation

```
begin
end
```

```
plot(Y, red, 3);
plot(OppositeSignY, green, 3);
```



## **floor() / ceiling()**

### Definition:

- floor() returns the biggest integer which is less than or equal to a given number.
- ceiling() returns the smallest integer which is greater than or equal to a given number.

### Format:

- float Floor (float value)

### Example 1:

| x    | floor(x) | ceiling(x) |
|------|----------|------------|
| 2.3  | 2.0      | 3.0        |
| 5.7  | 5.0      | 6.0        |
| 0.0  | 0.0      | 0.0        |
| -0.5 | -1.0     | 0.0        |
| -1.5 | -2.0     | -1.0       |

### Example 2:

- Below we draw a green line of ceiling values and a red line of floor values.

Express f\_Floor\_Ceiling

Vars

Series

line\_floor, line\_ceiling;

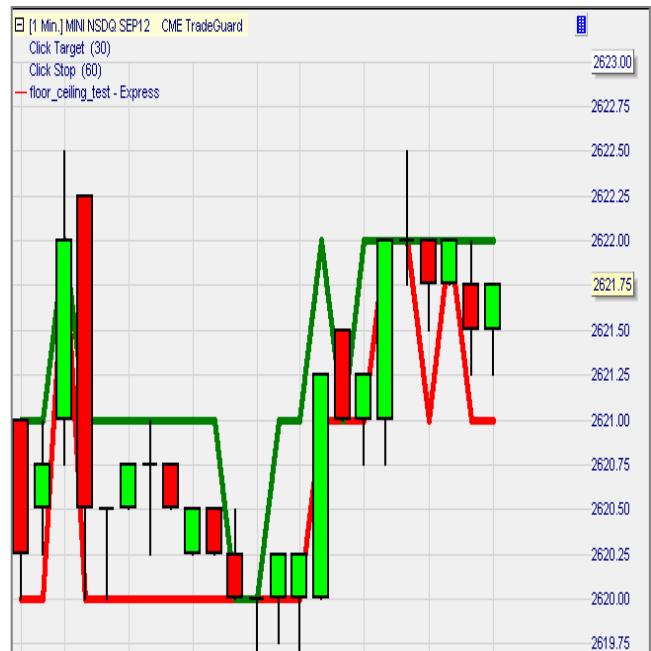
Calculation

```
line_floor = floor(c);
line_ceiling = ceiling(c);
```

Interpretation

```
begin
end
```

```
plot (line_floor, lightred, 3);
plot (line_ceiling, green, 3);
```



## Sum()

### Definition:

- Returns the sum of the elements series[0], ... series[span – 1].
  - Returns void if at least one element is void.

### Format:

- float Sum(series series, int span)

### Example:

- Below we draw the sum of the last five close prices:

```
Express f_Sum

Vars
Series
amount;

Input
$span(1, 100, 5);

Calculation

amount = sum(close, $span);

Interpretation
begin
end

plot (amount, green, 3);
```



## Atr()

### Definition:

- Measures the price volatility over a given period as a percentage value. It is an average of True Range over a given period. True Range being defined as follows:  
TrueRange = high - low;  
If (previous close <= high) then TrueRange = max(TrueRange, high - previous close);  
If (previous close >= low) then TrueRange = max(TrueRange, previous close - low);  
Atr(14) is TrueRange's moving average over 14 periods divided by the close price.

### Format:

- float Atr (int span)

### Example:

- The indicator below plots three different versions of the ATR: 1/ ATR in points, 2/ ATR in percent and 3/ ATR in ticks

```
Express f_Atr
```

```
Vars
```

```
input
```

```
$Period(1, 50, 14), $Type(1, 3, 1);
```

```
series
```

```
ATRpoints, ATRpercent, ATRticks, ATRline;
```

```
calculation
```

```
// ATR in points
```

```
if $Type = 1 then ATRline =
```

```
Round(Atr($Period)/100*close/TickSize(),0)*TickSize();
```

```
// ATR in percent
```

```
if $Type = 2 then ATRline = Atr($Period);
```

```
//ATR in ticks
```

```
if $Type = 3 then ATRline =
```

```
Round(Atr($Period)/100*close/TickSize(),0);
```

```
interpretation
```

```
begin
```

```
end
```

```
plot (ATRline,blue,2);
```



## AtrAbs()

### Definition:

- Measures the price volatility over a given period in points<sup>4</sup>. It is an average of True Range over a given period. True Range being defined as follows:  
TrueRange = high - low;  
If (previous close <= high) then TrueRange = max(TrueRange, high - previous close);  
If (previous close >= low) then TrueRange = max(TrueRange, previous close - low);  
AtrAbs(14) is TrueRange's moving average over 14 periods.

### Format:

- float AtrAbs (int span)

### Example:

```
Express f_AtrAbs
```

```
vars
```

```
input $Period(1, 50, 14);
series ATRline;
```

```
calculation
```

```
ATRline = AtrAbs($Period);
```

```
interpretation
```

```
begin
```

```
end
```

```
plot (ATRline,red,2);
```

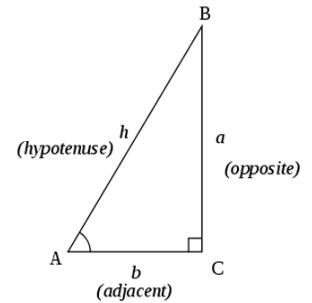


<sup>4</sup> AtrAbs() differs from Atr() in that it is expressed as an absolute value (= points) and not as a percentage value.

## Sine()

### Definition:

- Returns the sine value of a given angle.
  - Considering the triangle aside sine of angle AB-AC equals the ratio  $(a/h)$ :



### Format:

- float Sine(float value)

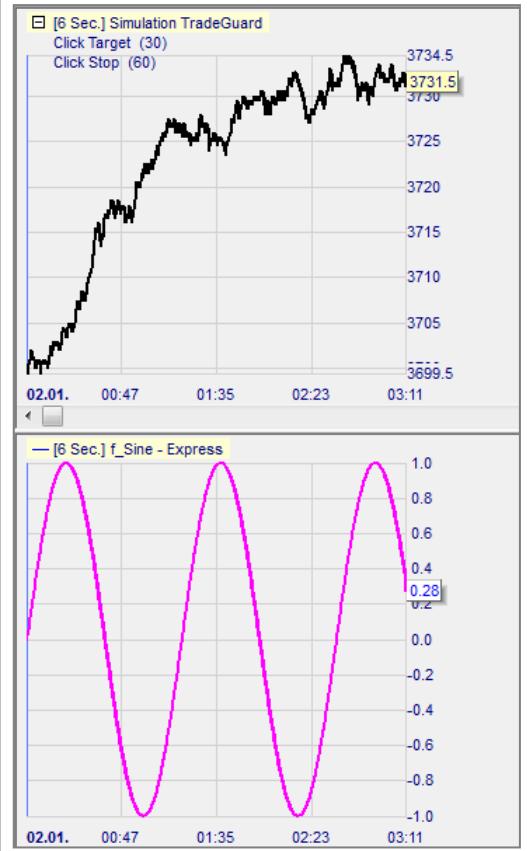
### Example 1:

- $\text{Sine}(0) = 0$
- $\text{Sine}(45) = 0.7071\dots$
- $\text{Sine}(90) = 1$

### Example 2:

- Below we draw the sine of the index of the bars:

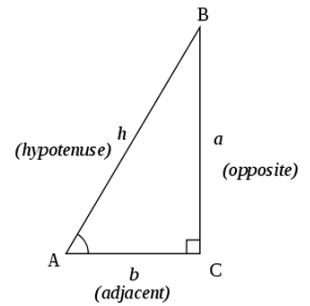
```
Express f_Sine
Vars
Series
val;
Calculation
val = sine(CurrentBarIndex());
Interpretation
begin
end
plot (val, magenta, 2);
```



## Cosine()

### Definition:

- Returns the cosine value of a given angle.
  - Considering the triangle aside cosine of angle AB-AC equals the ratio (b/h):



### Format:

- float Cosine (float value)

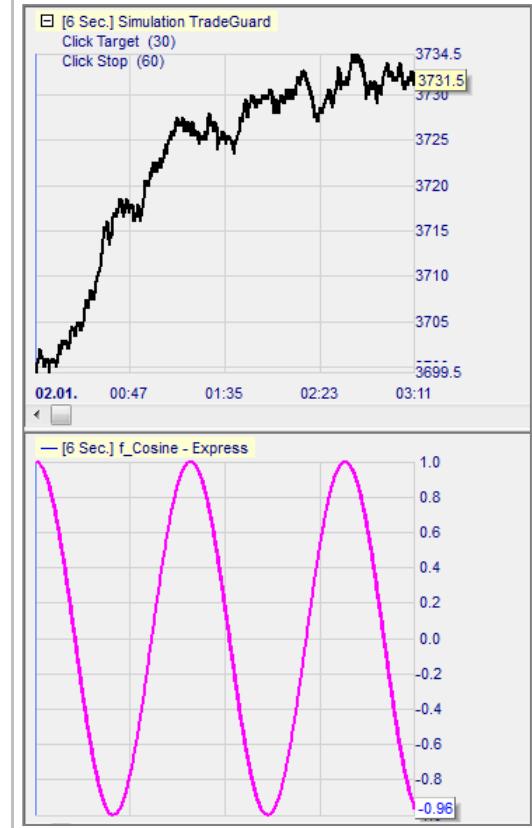
### Example 1:

- Cosine(0) = 1
- Cosine(45) = 0.7071...
- Cosine(90) = 0

### Example 2:

- Below we draw the cosine of the index of the bars:

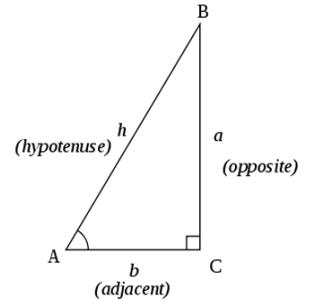
```
Express f_Cosine
Vars
Series
val;
Calculation
val = cosine(CurrentBarIndex());
Interpretation
begin
end
plot (val, magenta, 2);
```



## Tangent()

### Definition:

- Returns the tangent value of a given angle.
  - Considering the triangle aside tangent of angle AB-AC equals the ratio (a/b):



### Format:

- float Tangent (float value)

### Example 1:

- Tangent(0) = 0
- Tangent(45) = 1

### Example 2:

- Below we draw the tangent of the index of the bars:

Express f\_Tangent

Vars

Series  
value;

Calculation

```
value = tangent(FinalBarIndex() - CurrentBarIndex());
```

Interpretation

```
begin
end
```

```
plot (value, cyan, 3);
```



## ArcTangent()

### Definition:

- Is the reciprocal function of the Tangent() function:
  - $\text{ArcTangent}(\text{Tangent}(\text{angle})) = \text{angle}$
  - The returned angle is expressed as a number between  $-90^\circ$  and  $+90^\circ$ 
    - $\text{ArcTangent}(0) = 0$
    - $\text{ArcTangent}(1) = 45$

### Format:

- float ArcTangent (float value)

### Example:

- Below we display the angle defining the slope of the green moving average line.
  - To adjust the measure of the angle with what we see we enter two parameters:
    - Number of candles per centimeter
    - Number of points per centimeter
  - In the example below the angle goes from  $0^\circ$  (bottoming green line) to around  $50^\circ$  (steadily rising green line) before returning to  $0^\circ$  (flattening green line).

```
Express f_ArcTangent
```

```
vars
```

```
series
```

```
ma, angle;
```

```
input
```

```
$NumberOfpointsPerCM(0.1, 10.0, 2.9, 0.1, 1),
$NumberOfcandlesPerCM(0.1, 10.0, 4.0, 0.1, 1), $period(1, 50, 10);
```

```
calculation
```

```
if IsFirstBar() then
```

```
begin
```

```
CalculateAtEveryTick(false);
```

```
MovingAverage(c, ma, $period);
```

```
end
```

```
angle = ArcTangent((ma -
```

```
ma[1])/$NumberOfpointsPerCM*$NumberOfcandlesPerCM);
```

```
interpretation
```

```
begin
```

```
end
```

```
plot(angle,blue,2);
```



# Exp()

## Definition:

- Returns the exponential value of a given number.
  - It is the value of the constant  $e$  powered to a given number.
  - $e$  is a transcendental number that is the base of natural logarithms and is equal to approximately 2.718281828....

## Format:

- float Exp(float value)

## Example 1:

- $\text{Exp}(0) = 1$
- $\text{Exp}(1) = 2.718281828....$

## Example 2:

- Below we draw the exponential value of the close:

```
Express f_Exp
```

```
Vars
```

```
Series
val;
```

```
Calculation
```

```
val = exp(close);
```

```
Interpretation
```

```
begin
end
```

```
plot (val, red, 2);
```



## Log()

### Definition:

- Returns the logarithm value in base  $e$  of a given number.
  - $e$  is a transcendental number that is the base of natural logarithms and is equal to approximately 2.718281828....
  - $\text{Log}(\text{Exp}(\text{number})) = \text{number}$ .

### Format:

- float Log (float value)

### Example 1:

- $\text{Log}(0.1) = -2.30...$
- $\text{Log}(1) = 0$
- $\text{Log}(1000) = 6.91...$

### Example 2:

- Below we draw the logarithm in base  $e$  of the close:

```
Express f_Log
```

```
Vars
```

```
Series
val;
```

```
Calculation
```

```
val = log(close);
```

```
Interpretation
```

```
begin
end
```

```
plot (val, cyan, 3);
```



## Power()

### Definition:

- Returns the result of a number raised to a given power.

### Format:

- float Power (float value, float exponent)

### Example 1:

- Power(3, 0) = 1
- Power(3, 1) = 3
- Power(3, 2) = 9

### Example 2:

- Below we draw an exponential curve equal to the bar indices raised to power 2:

```
Express f_Power
```

```
Vars
```

```
series
```

```
Result;
```

```
input
```

```
$Exponent(1, 50, 2);
```

```
Calculation
```

```
Result = Power(CurrentBarIndex(), $Exponent);
```

```
Interpretation
```

```
begin
```

```
end
```

```
plot(Result, green, 2);
```



## SquareRoot()

### Definition:

- Returns the square root value of a given number.
  - Note: The number must be positive or null.

### Format:

- float SquareRoot (float value)

### Example 1:

| Number | SquareRoot(Number) |
|--------|--------------------|
| 1      | 1                  |
| 4      | 2                  |
| 9      | 3                  |
| 16     | 4                  |

### Example 2:

```
Express f_SquareRoot
```

```
Vars
```

```
Series
val;
```

```
Calculation
```

```
val = SquareRoot(CurrentBarIndex());
```

```
Interpretation
```

```
begin
end
```

```
plot (val, cyan, 2);
```



## Highest()

### Definition:

- Returns the highest value for the elements series[0], ..., series[span - 1].

### Format:

- `Float Highest (series series, int span)`

### Example:

- Below the green line TenBarHigh returns the highest value of the ten previous highs:

```
Express f_Highest
```

```
Vars
```

```
Series TenBarHigh;
```

```
Calculation
```

```
TenBarHigh = Highest(High, 10);
```

```
interpretation
```

```
begin
```

```
end
```

```
plot (TenBarHigh, green, 4);
```



## Lowest()

### Definition:

- Returns the lowest value for the elements series[0], … , series[span – 1].

### Format:

- float Lowest (series series, int span)

### Example:

- Below the red line TenBarLow returns the lowest value of the ten previous lows:

```
Express f_Lowest
```

```
Vars
```

```
Series TenBarLow;
```

```
Calculation
```

```
TenBarLow = Lowest(Low, 10);
```

```
interpretation
```

```
begin
```

```
end
```

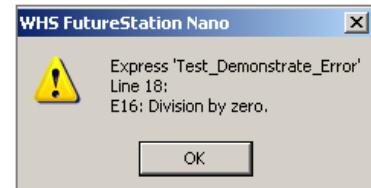
```
plot (TenBarLow, red, 4);
```



## IsZero()

### Definition:

- IsZero(number) returns true if AbsValue(number) <= 0.000 001
  - Consequence 1: This function treats as zero a non null number if it is smaller than 0.000 001.
  - Consequence 2: If one divides a number by a another non null number which is less than 0.000 001 the following message will appear:



### Format:

- bool IsZero(float value)

### Example:

- Below we draw the CurrentBarIndex line and highlight the background in light red if IsZero(CurrentBarIndex()/1 000 000 000) is true:
  - For the first 1001 bars the condition is true and the background is light red.
  - From the 1002 bar onwards the condition is false.

Express f\_IsZero

Vars

series

x, y;

numeric

a, b;

Calculation

```
if IsFirstBar() then a = 1/1000000000;

x = a*CurrentBarIndex();
y = CurrentBarIndex();

if IsZero(x) then Highlight("slot", "lightred");
```

Interpretation

begin

end

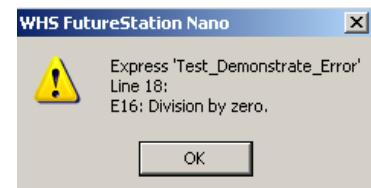
```
plot(y, blue, 2);
```



## IsNonZero()

### Definition:

- IsNonZero(number) returns true if AbsValue(number) > 0.000 001
  - Consequence 1: This function does not catch a non null number if it is smaller than 0.000 001.
  - Consequence 2: If one divides a number by a another non null number which is less than 0.000 001 the following message will appear:

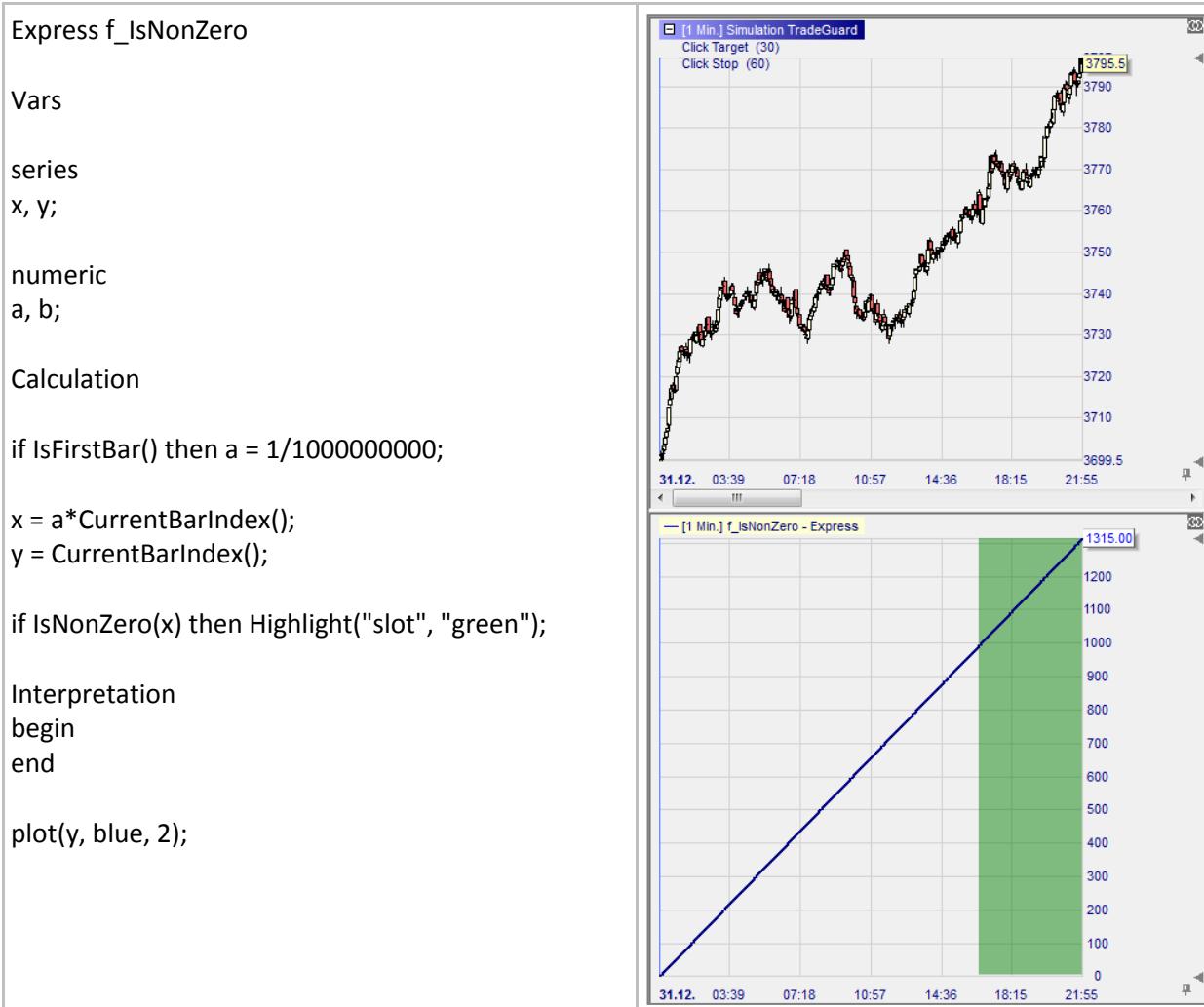


### Format:

- bool IsNonZero(float value)

### Example:

- Below we draw the CurrentBarIndex line and highlight the background in green if IsNonZero(CurrentBarIndex()/1 000 000 000) is true:
  - For the first 1001 bars the condition is false.
  - From the 1002 bar onwards the condition is true and the background is green.



## Max()

### Definition:

- Returns the greater of two numbers.

### Format:

- float Max(float value1, float value2)

### Example 1:

- Max(3, 8) returns 8.

### Example 2:

- Below we draw a line made of the maximum of the previous two close prices:

```
Express f_Max
```

```
Vars
```

```
series
```

```
MaxClose;
```

```
Calculation
```

```
MaxClose = Max(close[2], close[1]);
```

```
interpretation
```

```
begin
```

```
end
```

```
plot(MaxClose, green, 2);
```



## Min()

### Definition:

- Returns the lesser of two numbers.

### Format:

- float Min(float value1, float value2)

### Example 1:

- Min(3, 8) returns 3.

### Example 2:

- Below we draw a line made of the minimum of the previous two close prices:

```
Express f_Min
```

```
Vars
```

```
series
```

```
MinClose;
```

```
Calculation
```

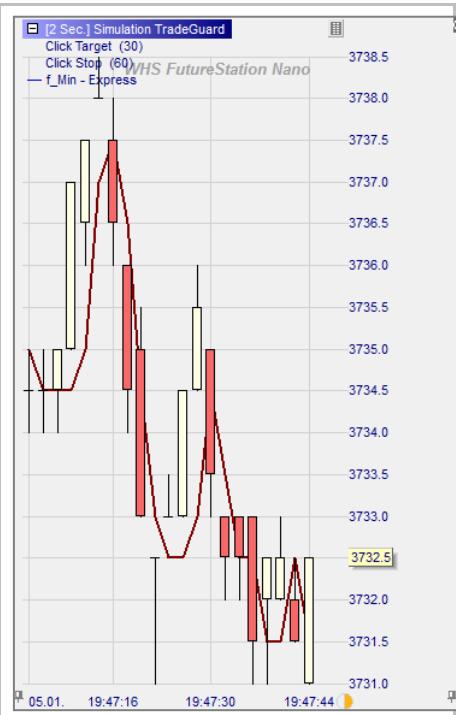
```
MinClose = Min(close[2], close[1]);
```

```
interpretation
```

```
begin
```

```
end
```

```
plot(MinClose, red, 2);
```



## Round()

### Definition:

- Returns the nearest rational number to a given number for a given number of decimals.
  - Midpoint (0.5) is rounded up to nearest rational number.

### Format:

- float Round (float value, int precision)

### Example 1:

- Round(1.1550, 0) = 1
- Round(1.1550, 1) = 1.1
- Round(1.1550, 3) = 1.156

### Example 2:

- Below we draw a curve made of the rounded values to three decimals of EUR/USD prices:

```
Express f_Round

Vars
series
myClose;
Calculation
myClose = Round(close, 3);

Interpretation
begin
end

plot (myClose, green, 3);
```



## RoundMultiple()

### Definition:

- Returns a number which is the nearest multiple of a given base to a given number. For example:
  - If the base is 2.5 and the number is 3743.5 the result is 3742.5:
    - $3743.5 / 2.5 = 1497.4$  which is rounded to 1497.0
    - $1497 \times 2.5 = 3742.5$
  - If the base is 2.5 and the number is 3739.5 the result is 3740.0:
    - $3739.5 / 2.5 = 1495.8$  which is rounded to 1496.0
    - $1496 \times 2.5 = 3740.0$

### Format:

- float RoundMultiple (float value, float multiple)

### Example:

- Below we draw in green the line of the RoundMultiple applied to the close and based on a multiple of 2.5:

```
Express f_RoundMultiple
```

```
Vars
```

```
input
```

```
$Base(0.1, 5.0, 2.5, 0.1, 1);
```

```
Series
```

```
val;
```

```
Calculation
```

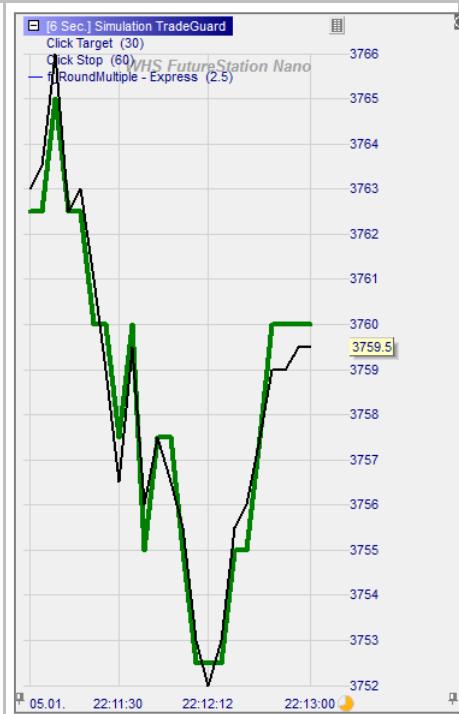
```
val = RoundMultiple (close, $Base);
```

```
Interpretation
```

```
begin
```

```
end
```

```
plot (val, green, 4);
```



## NormalCDF()

### Definition:

- Returns the value of the normal cumulative distribution function (CDF).
  - It is the cumulative distribution function with a mean of 0 and a standard deviation of 1.
    - It can be useful for computing probability-based pricing functions such as Black-Scholes.

### Format:

- NormalCDF (float value)

### Example:

- Below we draw the curve representing the normal CDF function (use parameters a and b to adjust it):

```
Express f_NormalCDF
```

```
vars
```

```
series
```

```
CBI, x;
```

```
input
```

```
$a(1, 500, 100), $b(1, 500, 10);
```

```
calculation
```

```
CBI = CurrentBarIndex();
x = NormalCDF((CBI - $b) / $a);
```

```
interpretation
```

```
begin
```

```
end
```

```
plot(x, red, 2);
```



## NormalPDF()

### Definition:

- Returns the value of the normal probability density function (PDF).
  - It is the probability density function with a mean of 0 and a standard deviation of 1.
    - It can be useful for computing probability-based pricing functions such as Black-Scholes.

### Format:

- NormalPDF (float value)

### Example:

- Below we draw the curve representing the normal PDF function (use parameters a and b to adjust it):

```
Express f_NormalPDF

vars

series
CBI, x;

input
$a(1,500,10), $b(1,500,150);

calculation

CBI = CurrentBarIndex();
x = NormalPDF((CBI - $b)/$a);

interpretation
begin
end

plot(x, blue, 2);
```



## Charting functions

### Plotline()

#### Definition:

- Draws a line at a given level.
  - Both predefined and RGB colors can be used.

#### Format:

- plotline (<constant or variable>, <colorname>, <pen width>);

#### Example:

- Below we draw a RSI:
  - The 100 and 0 horizontal lines are colored using a predefined color, grey.
  - The 70 and 30 horizontal lines are colored using RGB and defined using numeric variables.

```
Express f_plotline
```

```
Vars
```

```
Series
myRSI;
```

```
Numeric
upper(70), lower(30);
```

```
Calculation
```

```
If IsFirstBar() then RSI(close, myRSI, 14);
```

```
interpretation
```

```
begin
end
```

```
plot(myRSI, "lightblue", 2);
plotline(upper, 250, 100, 0, 2);
plotline(lower, 250, 100, 0, 2);
plotline(0, "grey", 2);
plotline(100, "grey", 2);
```



## Plot()

### Definition:

- Draws a simple curve.
  - Both predefined<sup>5</sup> and RGB colors (RGB= Red-Green-Blue) can be used.
  - In case of a typo in defining the color, the color is blue.
  - RGB colors are defined with three numbers. For example:



### Format:

- `plot(<series name>, "<color name>", <pen width>);`

### Example:

- Below we draw two simple moving averages:
  - The first one is based on the RGB code.
  - The second one on the predefined color "blue".

Express f\_plotRGB

Vars

Series

myMA, myMA2;

Calculation

If IsFirstBar() then

begin

  MovingAverage(close, myMA, 30);

  MovingAverage(close, myMA2, 10);

end

Interpretation

begin

end

plot(myMA, 230, 160, 79, 5);

plot(myMA2, "blue", 5);



<sup>5</sup> Predefined colors are black, white, red, green, blue, magenta, yellow, grey, lightred, lightgreen, lightblue.

## Plotband()

### Definition:

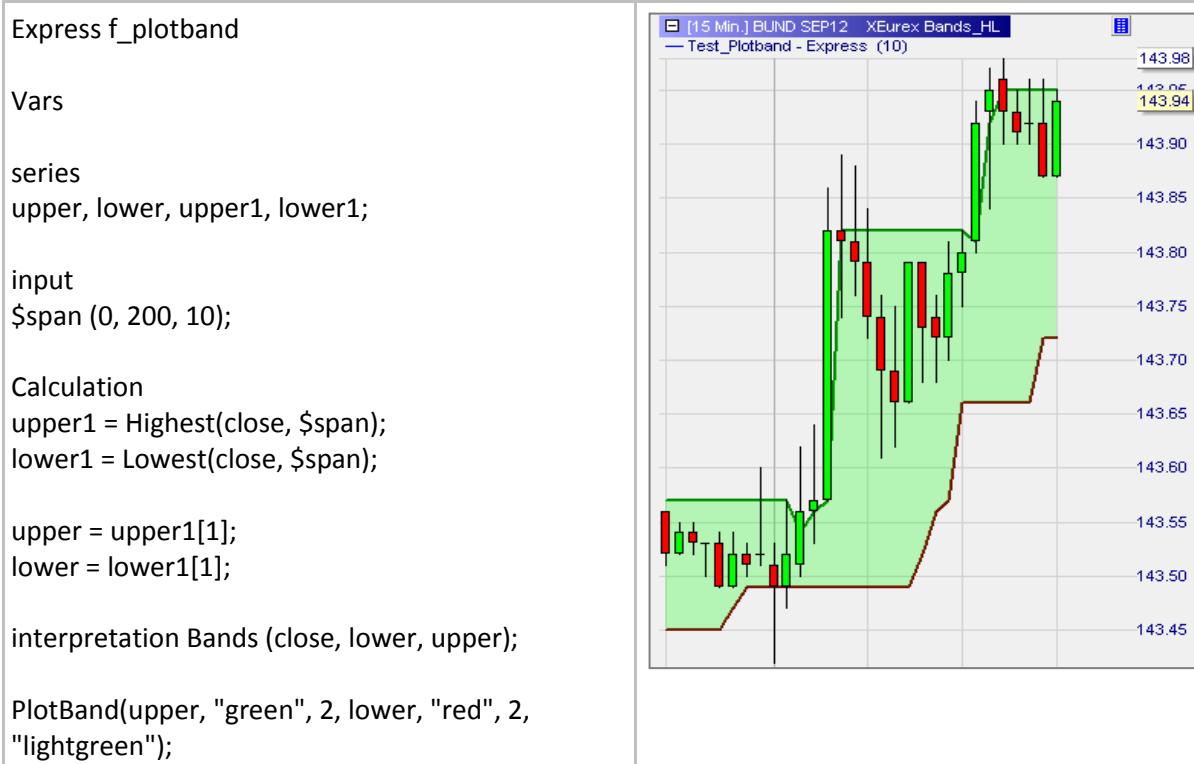
- Draws a band made of two curves and paints in one color the space in between.
  - Note: Only predefined colors can be used.

### Format:

- `plotband(<upper series name>, <color name>, <pen width>, <lower series name>, <color name>, <pen width>, <fill color>);`

### Example:

- Below we draw a band based on the highest highs and lowest lows of the last 10 bars:
  - The highest highs curve is in green and the lowest lows curve is in red.
  - The area between the two curves is in lightgreen.
  - Colors must be put in quotes: "blue".



## Plotcrossinglines()

### Definition:

- Draws a band made of two curves and paints in two colors the space in between.
  - Note: Only predefined colors can be used.

### Format:

- plotcrossinglines (<series1 name>, <color name>, <pen width>, <series2 name>, <color name>, <pen width>, <fill color series1 above series2>, <fill color series1 below series2>);

### Example:

- Below we draw a band based on the highest highs and lowest lows of the last 10 bars:
  - The highest highs curve is in green and the lowest lows curve is in red.
  - The area between the two curves is in lightgreen when the fast MA is above the slow MA and in lightred otherwise.
  - Colors must be put in quotes: "blue".

```
Express f_CrossingMovingAverages
```

```
Vars
```

```
Series
```

```
FastMA, SlowMA;
```

```
Input
```

```
$spanFMA(1, 200, 9), $spanSMA(1, 200, 30);
```

```
Calculation
```

```
If IsFirstBar() then
```

```
begin
```

```
 MovingAverage(close, SlowMA, $spanSMA);
```

```
 MovingAverage(close, FastMA, $spanFMA);
```

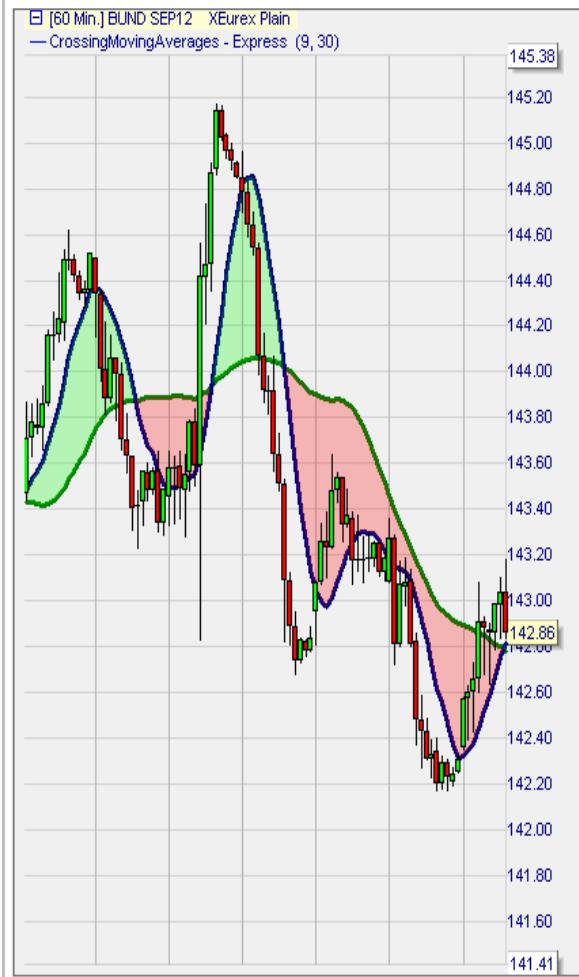
```
end
```

```
Interpretation
```

```
begin
```

```
end
```

```
plotcrossinglines(SlowMA, "green", 3, FastMA,
"blue", 3, "lightred", "lightgreen");
```



## Plotbars()

### Definition:

- Draws a bar chart.
  - Note: The colors used in the bar chart can only be changed in Extra / Colors.

### Format:

- plotbars(<open series>, <close series>, <high series>, <low series>);

### Example:

- Below we draw a bar chart based on the above candle chart:

```
Express f_PlotBarChart

vars
calculation
interpretation
begin
end

plotbars (open, close, high, low);
```



## Plotcandles()

### Definition:

- Draws a candle chart.
  - Note: The colors used in the bar chart can only be changed in Extra / Colors.

### Format:

- plotcandles(<open series>, <close series>, <high series>, <low series>);

### Example:

- Below we draw a candle chart based on the above bar chart:

```
Express f_PlotcandleChart

vars
calculation
interpretation
begin
end

plotcandles (open, close, high, low);
```



## GetPriceFormat()

### Definition:

- Applies the format of the MasterChart's y-axis into the sub-window.

### Format:

- string GetPriceFormat()

### Example:

- Below we are applying to our indicator's y-axis the same format as the MasterChart's.

```
Express f_GetPriceFormat_EFX
```

```
Vars
```

```
series
```

```
daxopen(studyDaxsep.open),
daxclose(studyDaxsep.close),
daxhigh(studyDaxsep.high),
daxlow(studyDaxsep.low);
```

```
Calculation
```

```
SetYscaleFormat (GetPriceFormat());
```

```
Interpretation
```

```
begin
end
```

```
plotcandles(daxopen, daxclose, daxhigh, daxlow);
```



## SetYscaleFormat()

### Definition:

- Defines the format of the y-axis in the sub window where indicators are plotted when they are not in the MasterChart.
  - Supports all formats used for the C-function “printf()”. The most important are:
    - “%f” decimal floating point
    - “%6.2f” rounds to two decimals
    - “%g” discards trailing zeroes
    - “%e” scientific notation

### Format:

- Void SetYscaleFormat (string format)

### Example:

- Below we set the format of the y-axis in the sub window as numbers with four decimals.

```
express f_SetYscaleFormat

vars

series
a;

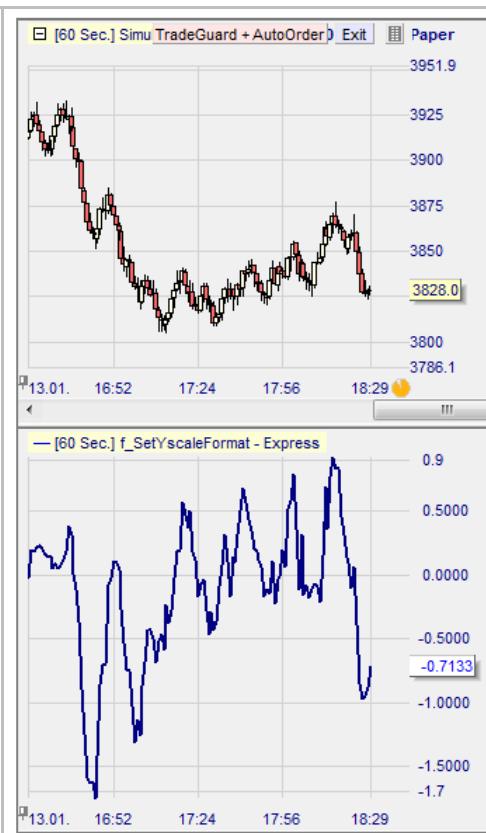
numeric ;
input ;

calculation

a = (c - c[10])/c[10]*100;
SetYscaleFormat("%9.4f");

interpretation
begin
end

plot(a,blue,2);
```

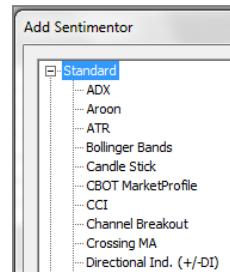


## Importing series

### Importing a series from a platform built-in indicator

#### Definition:

- Import a series from one of the standard indicator of the platform (see for an extract of the list of standard indicators on the right).

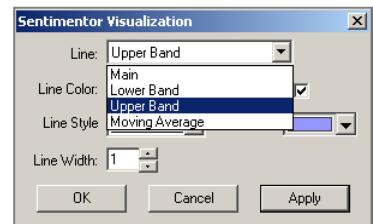


#### Syntax:

- series myseries (indicatorName.seriesName);
  - Important: When writing the indicatorName remove all space and non alphabetical characters, i.e. numbers, \_, -, +, /, ..., etc.
  - If there are two of the same indicator we need to indicate which one we want to import, i.e. series BBUB(BollingerBands.UpperBand2), the number 2 indicates that is the series that belongs to the second indicator.

#### Example:

- Below we create two bands similar to the Bollinger Bands: Same moving average but bands 20% narrower.
  - We get the names of the series from the visualization window (see on the right).



```
Express f_ImportBollingerBands
```

```
Vars
```

```
series
```

```
BBMA(BollingerBands.MovingAverage), NBUB, NBLB,
BBUB(BollingerBands.UpperBand),
BBLB(BollingerBands.LowerBand);
```

```
Calculation
```

```
NBUB = BBMA + (BBUB - BBLB)*0.8*0.5;
NBLB = BBMA - (BBUB - BBLB)*0.8*0.5;
```

```
interpretation
```

```
begin
end
```

```
plot(NBUB, yellow, 2);
plot(NBLB, yellow, 2);
```



## Importing a series from another express indicator

### Definition:

- Import a series from an indicator that is coded in express.

### Syntax:

- series myseries (indicatorName**Express**.seriesName);
  - Note the addition of the term **Express** above.
  - Important: When writing the indicatorName remove all space and non alphabetical characters, i.e. numbers, \_, -, +, /, ..., etc. Please note that this restriction does not apply to the name of the series itself.

### Example:

- The first indicator below creates a moving average 6 of RSI(14) called MAR (2<sup>nd</sup> chart). The second indicator imports MAR and displays it back in color:

|                                                                                     |                                                                                                   |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| Express f_MA_RSI                                                                    | express f_MA_RSI_Color                                                                            |
| Vars                                                                                | vars                                                                                              |
| Series<br>R, MAR;                                                                   | series<br>MAR(fMARSIExpress.MAR),<br>MARup, MARdw;                                                |
| Calculation                                                                         | calculation                                                                                       |
| If IsFirstBar() then<br>begin<br>RSI(c, R, 14);<br>MovingAverage(R, MAR, 6);<br>end | if MAR >= MAR[1] then<br>begin<br>MARup = MAR;<br>MARup[1] = MAR[1];<br>end<br>else MARup = void; |
| Interpretation<br>begin<br>end                                                      | if MAR < MAR[1] then<br>begin<br>MARdw = MAR;<br>MARdw[1] = MAR[1];<br>end<br>else MARdw = void;  |
| plot(MAR, blue, 2);                                                                 | interpretation<br>begin<br>end<br><br>plot(MARup, lightgreen, 2);<br>plot(MARdw, lightred, 2);    |

## Importing a price series from a symbol

### Definition:

- Imports any of symbol 2's open, close, high, low and volume series into a symbol 1's study.

### Syntax:

- series myseries (**studySymbolName.seriesName**);
  - We use the indicator Study to embed symbol 2 in our study.
    - Symbol 2's chart can be displayed in a sub window or in the main chart.
  - Note the term Study above followed by symbol name. SeriesName can be open, close, high, low or volume.
  - Important: When writing the indicatorName remove all space and non alphabetical characters, i.e. numbers, \_, -, +, /, ..., etc.
  - If we import only one symbol we can drop some of the information contained in the expression "studySymbolName" (see the example below).

### Example:

- Below symbol 1 is the Dax future. The imported symbol 2 is the Mini Nasdaq future. The top chart refers to symbol 1. The middle chart refers to symbol 2. The bottom chart is a bar chart constructed with the imported price series from symbol 2.
  - Because we are importing only one symbol we can work with reduced expression like `studyMININSDQ.close` instead of `studyMININSDQMAR.close`.

```
express f_ImportAnotherSymbol

vars

series
oo(studyMININSDQMAR.open),
cc(studyMININSDQ.close),
ll(studyMINI.low),
hh(study.high);

numeric ;
input ;

calculation

interpretation
begin
end

plotbars(oo, cc, hh, ll);
```



## Creating customized stops and targets

### **SetIntraPeriodUpdate()**

*This function can only be used with stops.*

#### Definition:

- Enables the stop to be updated tick by tick at every bar.
  - If a position is opened without SetIntraPeriodUpdate, the stop is updated tick by tick only at the first bar (the entry bar). At subsequent bars the stop is updated once at the close of every bar.
  - Stops with SetIntraPeriodUpdate are ignored in backtesting.

#### Format:

- void SetIntraPeriodUpdate()

#### Example:

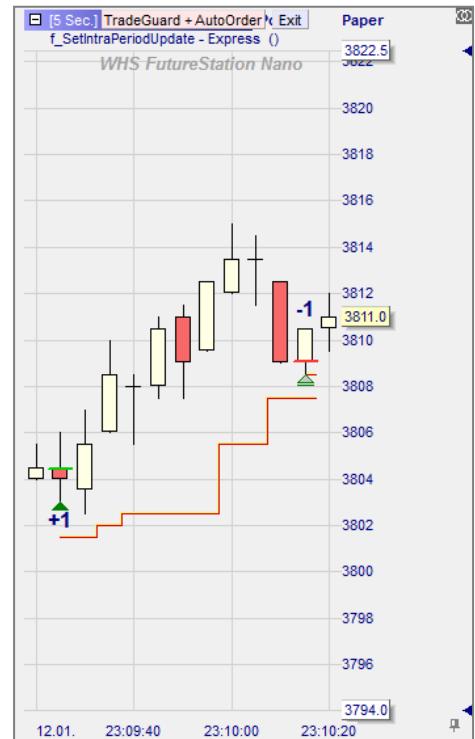
- Below we create a High/Low stop which is updated tick by tick so it is positioned at the lowest lows of the last 5 bars or at the highest highs of the latest 5 bars.

```
express Stop f_SetIntraPeriodUpdate

Calculation

SetIntraPeriodUpdate();

If Marketposition() = 1 then SetStopPrice(lowest(l, 5));
If Marketposition() = -1 then SetStopPrice(highest(h, 5));
```



## EntryPrice()

*This function can only be used with stops.*

### Definition:

- Returns two different values depending on whether TradeGuard<sup>6</sup> was activated and when:
  - TradeGuard deactivated: EntryPrice = Executed price<sup>7</sup>.
  - TradeGuard activated before opening of position: EntryPrice = Executed price<sup>3</sup>.
  - TradeGuard activated after opening of position: EntryPrice = TradeGuard activation price<sup>3</sup>.

### Format:

- float EntryPrice()

### Example:

- Below we create a fixed stop at 6 ticks (3 points) from the entry price. As the TradeGuard was activated when the position was opened entry price = executed price:
  - Executed price of 7732.5 as can be under Average Price.
  - On the chart the stop is 3 points (6 ticks) above 7732.5 as expected.

### Express Stop f\_EntryPrice

```
vars

numeric
StopLong, StopShort;

calculation

if BarsSinceEntry() = 0 then
begin
 StopLong = EntryPrice() - 6*TickSize();
 StopShort = EntryPrice() + 6*TickSize();
end
If MarketPosition() = 1 then SetStopPrice(StopLong);
else
if MarketPosition() = -1 then SetStopPrice(StopShort);
```



<sup>6</sup> With the activation of the Tradeguard stops are being placed automatically by the platform.

<sup>7</sup> We are referring to two prices. The first is the actual price at which we opened the position. It is called **Executed price**. The second is the price when we activated the TradeGuard. It is called **TradeGuard activation price**. These prices are different.

## EntryPriceOriginal()

*This function can only be used with stops.*

### Definition:

- Returns the executed price<sup>8</sup>.

### Format:

- float EntryPriceOriginal()

### Example:

- Below we create a fixed stop at 10 ticks (5 points) from the entry price original:
  - We opened the position at the executed price of 7729.0 as can be seen under Average Price and on the chart with the '-1 mark'.
  - We activated the TradeGuard afterwards and the platform positioned our stop 10 ticks (5 points) above as we wanted.

```
express Stop f_EntryPriceOriginal

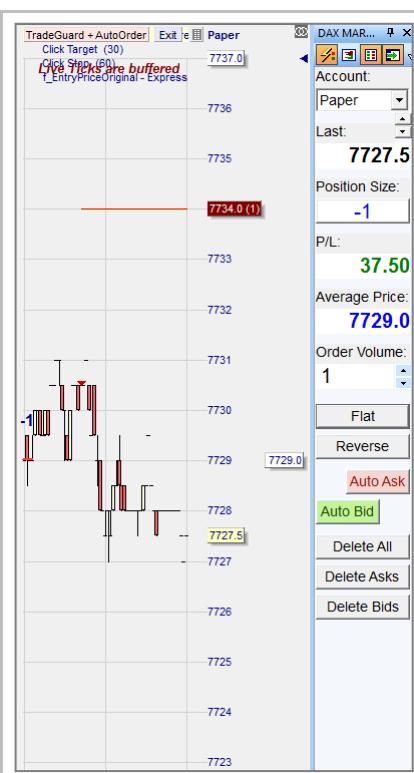
vars

numeric
StopLong, StopShort;

calculation

if BarsSinceEntry() = 0 then
begin
 StopLong = EntryPriceOriginal() - 10*TickSize();
 StopShort = EntryPriceOriginal() + 10*TickSize();
end

If MarketPosition() = 1 then SetStopPrice(StopLong);
else
if MarketPosition() = -1 then SetStopPrice(StopShort);
```



<sup>8</sup> We are referring to two prices. The first is the actual price at which we opened the position. It is called **Executed price**. The second is the price when we activated the TradeGuard. It is called **TradeGuard activation price**. These prices are different.

## BarsSinceEntry()

*This function can only be used with stops.*

### Definition:

- Returns the number of bars since the position was opened.
  - Returns 0 at the first bar, 1 at the second bar, 2 at the third bar, ...

### Format:

- int BarsSinceEntry()

### Example:

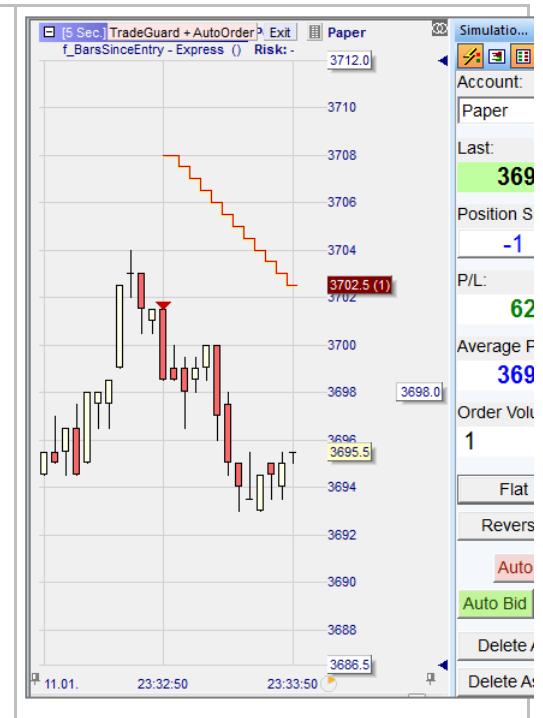
- Below we create a linear stop which starts 20 ticks away from our entry price before moving by one tick at the close of each bar. BarsSinceEntry() is used to define the bar of the entry and then to adapt the stop bar by bar.

```
express Stop f_BarsSinceEntry

vars
 numeric
 StpLong, StpShort;

calculation
 if BarsSinceEntry() = 0 then
 begin
 StpLong = EntryPriceOriginal() - 20*TickSize();
 StpShort = EntryPriceOriginal() + 20*TickSize();
 end
 else
 begin
 StpLong = max(StpLong, c - (20 +
 BarsSinceEntry())*TickSize());
 StpShort = min(StpShort, c + (20 -
 BarsSinceEntry())*TickSize());
 end

 if MarketPosition() = 1 then SetStopPrice(StpLong);
 if MarketPosition() = -1 then SetStopPrice(StpShort);
```



## IsIntradayEntry()

This function can only be used with stops.

### Definition:

- Returns true if the position has been opened in the current and not yet closed period.

### Format:

- bool IsIntradayEntry()

### Example:

- Below we create a linear stop which starts 10 ticks away from the original entry price before evolving one tick by one tick at the close of each bar. IsIntradayEntry is used to define the bar of the entry (in the same way as BarsSinceEntry() = 0 does it).

Express Stop f\_IsIntradayEntry

vars

numeric

last, StopL, StopS;

calculation

if MarketPosition() = 1 then

begin

  if IsIntradayEntry() then

    last = EntryPriceOriginal() - 10\*TickSize();

  else last = max(last, last + TickSize());

end

else if MarketPosition() = -1 then

begin

  if IsIntradayEntry() then

    last = EntryPriceOriginal() + 10\*TickSize();

  else last = min(last, last - TickSize());

end

SetStopPrice(last);



## MarketPosition()

*This function can only be used with stops.*

### Definition:

- Returns the current position: 1 = long, -1 = short.
  - Note: MarketPosition() = 0 or other number does not work.

### Format:

- int MarketPosition()

### Example:

- Below we create a fixed stop placed at 10 ticks from the original entry price. MarketPosition enables us to place a stop for each position long or short:

```
express Stop f_MarketPosition

vars

numeric
StopLong, StopShort;

calculation

if BarsSinceEntry() = 0 then
begin
 StopLong = c - 10*TickSize();
 StopShort = c + 10*TickSize();
end

if MarketPosition() = 1 then SetStopPrice(StopLong);
else
if MarketPosition() = -1 then SetStopPrice(StopShort);
```



## MinPriceEntryBar() / MaxPriceEntryBar()

*This function can only be used with stops.*

### Definition:

- **MinPriceEntryBar()**  
If a position is opened when TradeGuard is activated, MinPriceEntryBar returns the lowest price traded at the first bar (the bar of the entry) after the position is opened.  
If TradeGuard is activated when a position is already open, MinPriceEntryBar returns the lowest price traded at the first bar after the activation of TradeGuard.
- **MaxPriceEntryBar()**  
If a position is opened when TradeGuard is activated, MaxPriceEntryBar returns the highest price traded at the first bar (the bar of the entry) after the position is opened.  
If TradeGuard is activated when a position is already open, MaxPriceEntryBar returns the highest price traded at the first bar after the activation of TradeGuard.

### Format:

- float MaxPriceEntryBar(), float MinPriceEntryBar()

### Example:

- Below we create a stop which is initially 20 ticks away from the

Express Stop f\_MinMaxPriceEntryBar

Vars

numeric  
StopH, StopL;

Calculation

```
if BarsSinceEntry() = 0 then
begin
 StopH = MaxPriceEntryBar() + 20*TickSize();
 StopL = MinPriceEntryBar() - 20*TickSize();
end
else
begin
 StopH = min(StopH, Highest(h, 10));
 StopL = max(StopL, Lowest(l, 10));
end

If (MarketPosition() = -1) then SetStopPrice(StopH);
Else SetStopPrice(StopL);
```

The screenshot shows a candlestick chart for a financial instrument. The chart displays price action with various bars and a moving average line. On the right side, there is a detailed trade summary window titled 'f\_MinMaxPriceEntryBar - Express'. The summary includes:

- Click Target (20)
- Click Stop (20)
- f\_MinMaxPriceEntryBar - Express

Below this, the current market state is shown:

- Account: Paper
- Last: 3679.5
- Position Size: 1
- P/L: 112.50
- Average Price: 3675.0
- Order Volume: 1

Buttons for managing the trade include: Flat, Reverse, Auto Ask, Auto Bid, Delete All, and Delete Asks. The time axis at the bottom shows 08.01., 17:26:35, and 17:27:35.

## SetLongTrigger() / SetShortTrigger()

### Definition:

- Create conditional entry levels<sup>9</sup>.
  - Following a given signal, we define a price level at which a market, stop or limit order will be triggered to potentially open a position. If the order is not executed during the first period that follows the signal, it will be automatically cancelled and we will have to wait for another signal.

### Format:

- void SetLongTrigger (float value)

### Example 1:

- Below we generate automated signals based on the crossing of two moving averages. We are trying to enter at more favorable prices with a **conditional limit**<sup>10</sup> order placed at the mid price of the signal bar. Note below the set-up conditions and the description of what happens in blue:

**Express Stop f\_SetTrigger**

Vars

numeric  
bl, bs;

Calculation

```
bl = (h + l)/2;
bs = (h + l)/2;
SetLongTrigger(bl);
SetShortTrigger(bs);
```

Set-up of Order Default:

AutoOrder: Entry-Orders

Limit-Type: Limit

Stop-Type: Stop

Set-up of Evaluator Settings:

Signal execution

Sentiment-Enter Signal: Limit price next bar

Sentiment-Exit Signal: Close same bar

Execution of stop signal: Immediately

<sup>9</sup> Only apply to automated trades (AutoOrder must be activated) with AutoOrder Entry Orders set on “Limit = Limit” and “Stop = Stop” (To set up in Order Defaults) and Entry signals set on “confirmation price next bar” or “limit price next bar” (To set up in Evaluator Settings).

<sup>10</sup> If Evaluator Settings is set on “limit price next bar” and there are no functions like SetLongTrigger or SetShortTrigger in a program, the limit price will be set to the close of the period generating the signal. In case many indicators call this routine the strictest price is taken.

## Example 2:

- Below we generate automated signals based on the crossing of two moving averages. We are trying to enter at more favorable prices with a **conditional stop**<sup>11</sup> order placed 5 ticks away from the close price of the signal bar. Note below the set-up conditions and the description of what happens in blue:
- When a signal is created the platform places a stop order at the start of the next bar at the price level given by SetLongTrigger or SetShortTrigger.
- If the stop is hit before the next bar closes a position is opened, otherwise the stop order is cancelled.

|                                                                                                                                                                         |  |                                                                                                                                                                                                                                                                |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Express Stop f_SetTrigger</b> <pre> Vars numeric bl, bs;  Calculation bl = c + 5* TickSize(); bs = c - 5*TickSize(); SetLongTrigger(bl); SetShortTrigger(bs); </pre> |  | <b>Set-up of Order Default:</b> <p>AutoOrder: Entry-Orders</p> <p>Limit-Type: Limit</p> <p>Stop-Type: Stop</p>                                                                                                                                                 |
|                                                                                                                                                                         |  | <b>Set-up of Evaluator Settings:</b> <p>Signal execution</p> <p>Sentiment-Enter Signal: Confirmation price next bar <span style="color: red;">(Red Arrow)</span></p> <p>Sentiment-Exit Signal: Close same bar</p> <p>Execution of stop signal: Immediately</p> |

<sup>11</sup> If Evaluator Settings is set on “confirmation price next bar” and there are no functions like SetLongTrigger or SetShortTrigger in a program, the stop price will be set to the high/low of the period generating the signal. In case many indicators call this routine the strictest price is taken

## **SetStopPrice()**

*This function can only be used with stops.*

### Definition:

- Places a stop order at a given price level.

### Format:

- void SetStopPrice (float value)

### Example:

- Below we place a stop 2 ticks away from the lowest lows or the highest highs of the last 10 candles:
  - As we entered short two stops were positioned. The click stop at 4047 and the stop we programmed below which is 2 ticks above the line of the highest highs.

```
Express Stop f_SetStopPrice

vars

series
l, hh;

Calculation
l = Lowest(l, 10);
hh = Highest(h, 10);

If MarketPosition() = 1 then
SetStopPrice (l - 2*TickSize());
else
if MarketPosition() = -1 then
SetStopPrice (hh + 2*TickSize());
```

## SetTargetPrice()

This function can only be used with stops.

### Definition:

- Places a target order at a given price level.

### Format:

- void SetTargetPrice (float value)

### Example:

- Below we create a dynamic target which is depends on the range between the highest highs and lowest lows of the latest 5 bars.
  - Just after the entry the range was increasing so the target price was moved down at 3720.
  - Later the range contracted so our target remained at the same level.
  - Then as the range started again to expand while the market moved down our target moved down as well.

```
express Stop f_SetTargetPrice

vars

numeric
TargetL, TargetS, Range;

input
$n(1, 50, 5), $k(0.1, 2.0, 1.0, 0.1, 1);

calculation

Range = Highest(h, $n) - Lowest(l, $n);
if BarsSinceEntry() = 0 then
begin
 TargetL = c + $k*Range;
 TargetS = c - $k*Range;
end
else
begin
 TargetL = max(TargetL, c + $k*Range);
 TargetS = min(TargetS, c - $k*Range);
end

If MarketPosition() = 1 then SetTargetPrice(TargetL);
else
if MarketPosition() = -1 then SetTargetPrice(TargetS);
```



## Predefined interpretation tools

### CrossesAbove() / CrossesBelow()

#### Definition:

- Functions used to detect the crossing of two curves.
  - `CrossesAbove(curve, trigger) = true if (curve[1] <= trigger[1]) and (curve > trigger)`
  - `CrossesBelow(curve, trigger) = true if (curve[1] >= trigger[1]) and (curve < trigger)`

#### Format:

- `bool CrossesAbove (series curve, series trigger)`

#### Example:

- Below the indicator is a Bollinger Bands indicator (which has been imported).
- The `CrossesAbove` and `CrossesBelow` functions help define the interpretation:
  - A long signal is triggered when the close crosses above the upperband
  - A short signal is triggered when the close crosses below the lowerband



## CrossesAboveThreshold() / CrossesBelowThreshold()

### Definition:

- Functions used to detect the crossing of a given threshold by a curve.
  - `CrossesAboveThreshold(curve, threshold) = true if (curve[1] <= threshold) and (curve > threshold)`
  - `CrossesBelowThreshold(curve, threshold) = true if (curve[1] >= threshold) and (curve < threshold)`

### Format:

- `bool CrossesAboveThreshold (series curve, float threshold)`

### Example:

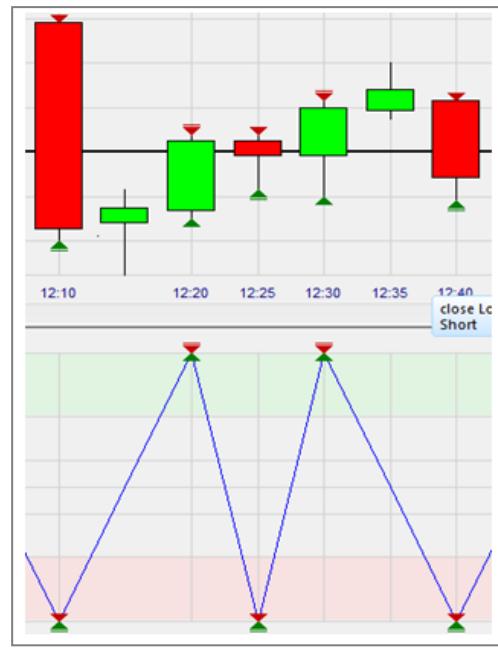
- Below the indicator is a horizontal line (the threshold at 6930 points).
- The `CrossesAboveThreshold` and `CrossesBelow` functions help define the interpretation:
  - A long signal is triggered if the close price crosses above 6930
  - A short signal is triggered if the close price crosses below 6930

Express f\_CrossesAboveBelowThreshold

Vars  
series;  
input \$span (0, 200, 10);

Calculation

interpretation  
begin  
if `CrossesAboveThreshold(close, 6930)` then sentiment = 100;  
if `CrossesBelowThreshold(close, 6930)` then sentiment = 0;  
end  
  
`plotline(6930, black, 2);`



## Triggerline()

### Definition:

- Interpretation scheme for indicators based on the crossing by a curve of another curve.
  - For example the close price crossing a moving average.

### Format:

- TriggerLine (series curve, series trigger)

### Example:

- Below the indicators are two moving averages.
- The interpretation is defined using the TriggerLine interpretation.
- By right clicking on the indicator and selecting Edit Interpretation one can display a window where the interpretation parameters can be modified.
  - Here we buy when the fast moving average crosses above the slow moving average (= 100) and we sell when the fast moving average crosses below the slow moving average (= 0).

```
Express f_Triggerline
```

Vars

Series  
FastMA, SlowMA;

Input

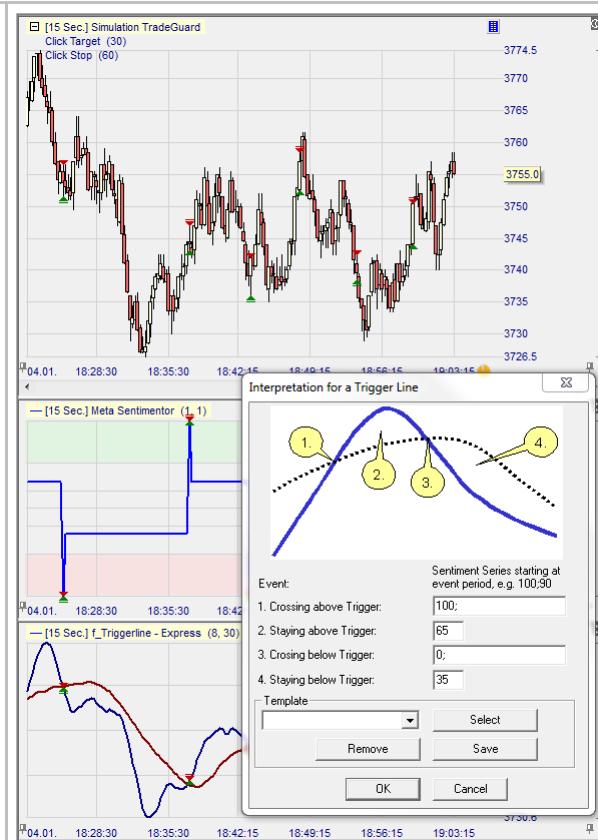
```
$spanFMA(1, 200, 8), $spanSMA(1, 200, 30);
```

Calculation

```
If IsFirstBar() then
begin
 MovingAverage(close, SlowMA, $spanSMA);
 MovingAverage(close, FastMA, $spanFMA);
end
```

```
Interpretation triggerline(FastMA, SlowMA);
```

```
plot (FastMA, blue, 2);
plot (SlowMA, red, 2);
```



## Swing()

### Definition:

- Interpretation scheme based on upwards and downwards swings in a curve.
  - For example the swings of the momentum indicators or other oscillators.

### Format:

- void Swing (series series, input spanLeft, input spanRight)

### Example:

- Below the indicator is a moving average of a RSI.
- The interpretation is defined using the Swing interpretation.
- By right clicking on the indicator and selecting Edit Interpretation one can display a window where the interpretation parameters can be modified.
  - Here we buy at the start of a new upswing (= 100).
    - A new upswing starts after a downswing as soon as there are two consecutive new highs (this parameter can be adjusted).
  - We sell at the start of a new downside (= 0).
    - A new downswing starts after an upswing as soon as there are two consecutive new lows (this parameter can be adjusted).

Express f\_Swing

Vars

Series  
myRSI, smoothedRSI;

Input  
\$spanLEFT(1,100,2), \$spanRIGHT(1,100,2),  
\$RSIspan(1, 100, 14), \$MAspan(1, 200, 10);

Calculation

If IsFirstBar() then  
begin  
  RSI(close, myRSI, \$RSIspan);  
  MovingAverage(myRSI, smoothedRSI, \$MAspan);  
end

Interpretation Swing(smoothedRSI, \$spanLEFT,  
\$spanRIGHT);

plot(smoothedRSI, red, 2);

## Bands()

### Definition:

- Interpretation scheme for indicators based on the crossing by a curve of two other curves forming a band.
  - For example the close price crossing these types of bands: Bollinger Bands, Donchian-Channel, Price-Channel, Commodity-Channel, ....

### Format:

- void Bands (series series, series lower, series upper)

### Example:

- Below the indicator is a channel based on the highest highs and lowest lows over 10 bars.
- The interpretation is defined using the Bands interpretation.
- By right clicking on the indicator and selecting Edit Interpretation one can display a window where the interpretation parameters can be modified.
  - Here we buy when the close crosses the upper curve (= 100) and we sell when the close crosses below the lower curve (= 0).

```
Express f_Bands

Vars

series
upper, lower, upper1, lower1;

input
$span (0, 200, 10);

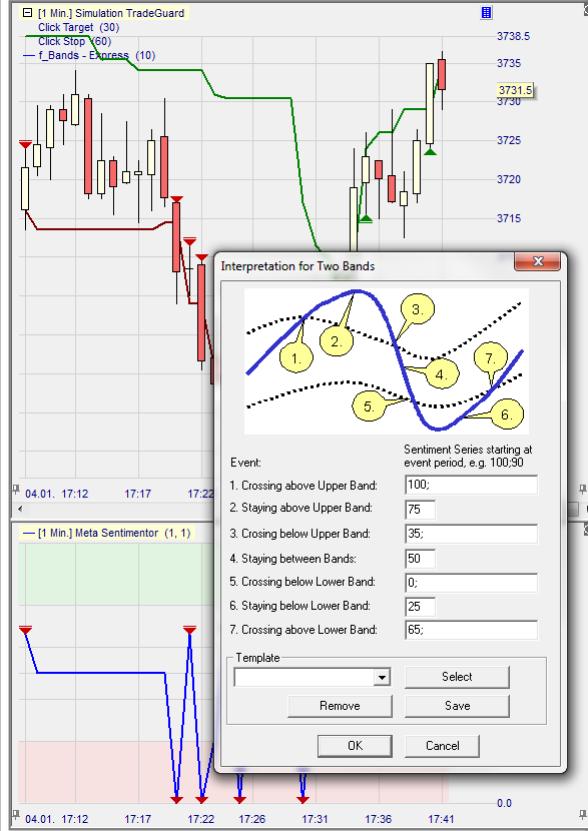
Calculation

upper1 = Highest(h, $span);
lower1 = Lowest(l, $span);

upper = upper1[1];
lower = lower1[1];

interpretation Bands (close, lower, upper);

plot (upper, green, 2);
plot (lower, red, 2);
```



The screenshot shows a trading interface with a candlestick chart for EURUSD. A green line represents the upper band and a red line represents the lower band. The chart shows several price crossings between these bands. A context menu is open over the bands, with a sub-menu titled 'Interpretation for Two Bands' visible. This sub-menu contains seven numbered items corresponding to the legend in the chart, with input fields for each item. The items are: 1. Crossing above Upper Band: 100; 2. Staying above Upper Band: 75; 3. Crossing below Upper Band: 35; 4. Staying between Bands: 50; 5. Crossing below Lower Band: 0; 6. Staying below Lower Band: 25; 7. Crossing above Lower Band: 65. Buttons for OK, Cancel, Remove, and Save are at the bottom.

## TwoThresholds()

### Definition:

- Interpretation scheme for indicators based on the crossing by a curve of two horizontal lines.
  - For example a RSI, a stochastic or other oscillators crossing two upper and lower lines.

### Format:

- void TwoThresholds (series series, input upThreshold, input downThreshold)

### Example:

- Below the indicator is a moving average over 10 bars of a RSI(14).
- The interpretation is defined using the TwoThresholds interpretation.
- By right-clicking on the indicator and selecting Edit Interpretation one can display a window where the interpretation parameters can be modified.
  - A long signal is triggered in case #7
  - A short signal is triggered in case #3

```
Express f_TwoThresholds

Vars

Series
myRSI, smoothedRSI;

Input
$RSIspan(1, 100, 14), $MAspan(1, 200, 10),
$upperzone(51, 99, 80), $lowerzone(1, 49, 20);

Calculation
If IsFirstBar() then
begin
 RSI(close, myRSI, $RSIspan);
 MovingAverage(myRSI, smoothedRSI, $MAspan);
end

Interpretation TwoThresholds(smoothedRSI,
$upperzone, $lowerzone);

plot(smoothedRSI, red, 2);
plotline(100, grey, 1);
plotline(80, grey, 1);
plotline(20, grey, 1);
plotline(0, grey, 1);
```

