FUNCTIONS AND PROPERTIES **VBA**

GUIDE FOR THE USER

VisualChart
Real Time Financial Information & Trading Software

Functions and Properties VBA

Alert

Description:
This function is used in indicators programming in order to trigger alerts. When certain conditions defined by the user are fulfilled a warning message shows up on screen.

Syntax:
.Alert(Description)

Parameters:

Name	Default	Description
Description	"Indicator Alert"	Text showing up when the alert is triggered.

The property **Indicator Alert** must be activated in the indicator properties editor so that the alert message shows up on screen.

Example:

.Alert("Warning, Average Crossover")

The following message will show up on screen "Warning, Average Crossover"

Angle

Description:
This function returns the value of the angle between the horizontal line and the regression line formed by the prices StartPrice andEndPrice that related the quotes with the time variable.

Syntax:
.Angle (StartBar, EndBar, StartPrice, EndPrice,Identifier)

Parameters:

Name	**Default**	**Description**
StarBar	-	Number of the bar where the straight line starts.
EndBar	-	Number of the bar where the straight line ends up.
StarPrice	-	Start price for the straight line
EndPrice	-	End price for the straight line
Identifier	-	Primary data source. The system interacts with data that is the name taken by the chart data series on which the strategy is applied. If there is more than one chart inserted in the same window they will be coded as Data2, Data3, Data4, etc.

Example:

Let´s assume that we are willing to know, at any stage, the value in radians between the current closing price and the closing price 30 bars ago. In this case, we should define first the start variable.

Dim AngleInRadians As Double

The value returned by the call to this property will be assigned to this variable:

AngleInRadians=.Angle(Bar-30, Bar, .Close (30), .Close (0))

AvgBarsInTrade

Description:
This function returns the average number of bars during which a trade is opened. This value will increase by while new bars are generated and its value will depend of the bar where this property is called.

Syntax:
.AvgBarsInTrade

Parameters:

Name	Default	Description
-	-	-

Example:

Let´s assume that we want to know the average number of bars that our trades last. First we must define a variable:

Dim TradeAvbars as Double

We caculate the average number of bars by trade by asking the following questions:

If.NumberOfTrades> 1 then

With this question we make sure that, at least, two trades have been generated.

The value returned by the property will be assigned to the variable

Then, we can stock the number of trades we have made under another variable previously defined:

.NumberOfTrades

We would calculate again the average number of bars by trades only when

.NumberOfTrades> NumCurrentTrades

AvgLosingTrade

Description:
This function returns the average results the average results of the losing trades. This value will change as new bars are generated and it will also depend on the bar thwre this property is called.

Syntax:
.AvgLosingTrade(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which the information will show up, **ByPoints** or**Percentage**

Example:

.AvgLosingTrade(ByPoints)
Assigns to the previously defined variable the average loses of all the losing trades. (In points)

AvgTrade

Description:
This function returns the average results of the trades. This value will change by while new bars are generated and its value will depend on the bar on which this property is called.

Syntax:
.AvgTrade(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which the information will show up, **ByPoints** or **Percentage**

Example:

.AvgTrade (Percentage)

This function assigns to the previously defined variable AvgProfit the average profit of all trades (in %).

AvgWinningTrade

Description:
This function returns the average result of the wining trades. This value will change by while new bars are generated and its value will depend on the bar on which this property is called.

Syntax:
.AvgWinningTrade(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which the information will show up, **ByPoints** or **Percentage**

Example:

.AvgWinnigTrade(ByPoints)

Asigns to the previously defined varaibleAvgProfitWinners the average profit of all the positive trades (in points).

BestSeries

Description:
This function returns the best value reached by the total profit. This will change by while new bars are generated and its value will depend on the bar on which this property has been called.

Syntax:
.BestSeries(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which the information will show up, **ByPoints** or **Percentage**

Example:

.BestSeries(Percentage)

Returns in %, the best value of the total profit reached when applying this property.

Buy

Description:
This function is used to send buy orders in stocks, futures, cfd´s etc...
Syntax:
.Buy Type, Contracts, Price, Label

Parameters:

Name	Default	Description
Type	AtClose	Type of order to be launched (AtClose, AtMarket, AtLimit, AtStop).
Contracts	1	Number of contract/stocks. The numerical specifications on contracts can be replaced by variables or by any function previously described.
Price	-	Buy price. This parameter must only be indicated for AtStop and AtLimit orders. The value can be expressed by using a number, a variable, a function or a mix of a variable and function.
Label	-	Label of the order in text format.

Example:

.Buy AtStop, 1, .High +10, "C1"

Sends a buy order AtStop and for one contract. The stop price is the top of the bar plus 10 points and the indentificant label of the order “C1”.

CalcDate

Description:
Sums a certain amounts of days to a certain day and returns as a result the resulting date under **military format(AAAAMMDD).**

Syntax:
.CalcDate(Date, Days)

Parameters:

Name	Default	Description
Date	-	Date (AAAAMMDD) to which the corresponding amount of days will be added.
Days	-	Amount of days to be added..

Example:

.CalcDate(2010110,5)

It sums 5 days to the date 10/01/2010, that under military format is 20100110and as a consequence returns 20100115, that under date format is 15/01/2010.

CalcTime

Description:
Sums an amount of minutes to a certain time and returns the result in military format (HHMM).

Syntax:
.CalcTime(Time, Minutes)

Parameters:

Name	Default	Description
Time	-	Time(HHMM) to which a certain amount of minutes will be added.
Minutes	-	Amount of minutes to be added.

Example:

.CalcTime(1000,30)

Sums30 minutes to 10:00 am (militaryformat 1000), and thus returns 1030, that under military format is 10:30 am.

Close

Description:
This function returns the value of the close of a certain bar.

Syntax:
.Close(BarsAgo, Identifier)

Parameters:

Name	Default	Description
BarsAgo	0	Number of bars backwards. The Defaultvalue refers to the current bar. In this parameter we can indicate a posible numerical value contained under the variable or type the value directly. It can also be specified as a function replacing the numerical value. This parameter only allows positive values.
Identifier	Data	Data source on which the function is applied. The system acts on Data that is the Name taken by the data series on which the strategy is applied. If there is more than one chart in the same window they will be named as Data1, Data2, Data3,etc.

Example:

.Close(3,Data1)

Returns the data three bars backwards of the Data Source codified as Data1.

ConfigStk

Description:
Enables to set the initial properties of the statistics that we are willing to obtain.

Syntax:
.ConfigStk(Sing, Unit, Filt, CompType, Compression, BeginDate, EndDate)

Parameters:

Name	Default	Description
Sing	ssNets	Results Filter depending on whether the results are winers (ssWins) orlosers (ssLoss).
Unit	suMoney	Data representation in cash (suMoney), in points (suPnts) orin percentage (ssPorc).
Filt	sfAll	Results filter by positions;in fact, depending if they are longs (sfLong) or shorts (sfShort).
CompType	sctTrades	Representation of the data grouped by trades (sctTraders), days (sctDays), weeks (sctWeeks), months (sctMoths) or years (sctYears).
Compresión	1	Compression Unit.
BeginDate	1	Start Date for the data range during which we want to extract the data.
EndDate	-	Final date for the data range during which we want to extract the data.

Example:

.ConfigStk ssLoss, suMoney, sfShort, sctDays,1, cDate(”09/08/2009”), cDate(“09/08/2010”)

This function sets the properties of a system statistics to obtain information in cash for the losses of a single day (only the ones where we have trades short), from August 9th 2009 toSeptember 9th 2010.

CurrentBar

Description:
This function returns the ordinal number of the bar on which the calculations are being run (current bar). Considering the first bar of the data series as bar number 0, the number will be increase in one unity every each evaluated bar.

When we work with two data series and one has more historical data than the other the calculations will start when one of the bars of the first series coincides in time with one of the bar of the second sereis. This bar will be considered as first bar (CurrentBar=1).

Syntax:
.CurrentBar

Parameters:

Name	**Default**	**Description**
-	-	-

Example:

.CurrentBar

If the calculations were being made in the seventh bar of the data series, this function will return for example the value 6.

CurrentContract

Description:
This function is used to know the amount of contracts or shares that are bought or sold in the current opened position.

Syntax:
.CurrentContract

Parameters:

Name	**Default**	**Description**
-	-	-

Example:

.CurrentContractif there is no position opened the function will return 0.

CurrentEntries

Description:
This function is used in order to know the number of different entries for an opened position.A position can have different entries in function of the order labelo r the modality of matching the orders.

Syntax:
.CurrentEntries

Parameters:

Name	Default	Description
-	-	-

Example:

.CurrentEntries

The function returns the number of entries at the momento of applying the order. If they are no functions it will return the value 0.

Date

Description:
All the bars of the chart have an associated date, even if they are intraday bars. The function returns the date on which the bar was produced. The returned format is military format so if the bar was produces on August 10th; the function will return the result 20000810.

Syntax:
.Date (BarsAgo,Data)

Parameters:

Name	Default	Description
BarsAgo	0	Number of bars backwards. TheDefaultvalue refers to the current bar. We can indicate for this parameter a numerical value included in a variable or directly type this value. This parameter only allows positive values.
Data	Data	Data source on which the function is going to be applied. The system acts upon Data which is the Name taken by the data series on the chart on which the strategy is applied. If there is more than one chart in the same window the will be known as Data1, Data2, Data3, etc.

Example:

.Date (3,Data)

The function returns the date corresponding to the third bar backwards from the current one of the data series known asData.

▣DateSubstract

Description:
Returns the difference in days between two dates**.**

Syntax:
.DateSubstract(Left, Right)

Parameters:

Name	Default	Description
Left	-	Minuendo (date in military format to which we substract)
Right	-	Sustraendo (date in military format to which we substract)

Example:

.DateSubstract(20100110, 20100120) The function returns 10.

▣ExitLong

Description:
This function is used when we are willing to close a long position but not to take, at the same time, a short positions. For example, if we have bought 500 shares, and the conditions to exit this trade are fulfilled, we will use this function.

Syntax:
.ExitLong Type, Contracts, Price, Label

Parameters:

Name	Default	Description
Type	AtClose	Type of order to be launched (AtClose, AtMarket, AtLimit, AtStop).
Contracts	1	Number of contracts /stocks. The numerical specifications on contract numbers can be replaced by variable or any other function previously defined.
Price	---	Buy Price. This parameter is only used for AtStop and AtLimit orders. We can replace this number by a function a variable or a mixed of both the function and the variable.
Label	---	Order label in text format.

- ➢ If the total amount of contracts/shares (Contracts) and neither the label (Label,in case we are using several buy orders),are not specified the full position will be closed and we will exit the market.
- ➢ If the amount of contract/shares is not specified by the lable is, and if we are using several buy orders, alll the contracts/shares corresponding to this lable will be closed.
- ➢ If the amount of contracts and the label are specified, the amount of contract shares for the order which its label has been specified will be closed.
- ➢ If the number of contract/shares is specified but not the label, and presuming that several positions are opened,the amount of contracts/shares specified in the last order will be closed.

Examples:

.ExitLong AtStop, 1, .GetEntryPrice-10, "C1"

The position (1 contract), if the order is labeled as “C1”, will be closed with a sell order at stop at the entry price minus 10 points. In this case, we are talking about a stop loss order.

.ExitLong AtLimit, 1, .GetEntryPrice+30 .The positions (1 contract) will be closed with a sell limit order at the entry price plus 30 points. In this case, we are talking about a target profit order.

ExitShort

Description:
This function is used to close a short position without opening a long position.. For Example, is we have sold 5 futures contracts, and the necessary conditions are fulfilled to liquídate the position, we shall use this function.

Syntax:
.ExitShort Type, Contracts, Price, Label

Parameters:

Name	Default	Description
Type	AtClose	Type of order to be launched (AtClose, AtMarket, AtLimit, AtStop).
Contracts	1	Number of contracts/stocks. The numercial specifications on contracts can be replaced by variable or by any function previously defined.
Price	---	Buy price. This parameter must only be indicated for AtStop and AtLimit type orders. The value can be expressed by a number, a variable, a function or by a mix of variable and function.
Label	---	Label of the order in text format.

- If nor the number of contracts/stocks (contracts) neither the label (Label if we are using several sell orders), are specified, the full position will be closed and we will exit the market.
- If the number of stokcs is not specified but the lable is (assuming that we are using several sell orders),all the contract/stocks corresponding to the order with this label will be closed.
- If the number of contracts and the label are specified then this number of contracts/stocks of the order with the specified lable will be closed.
- If the number of contracts/stocks is specified but the label is not and if they are several opened positions, the number of contracts specified in the last order will be closed.

Examples:

.ExitShort AtStop, 1, .High +10, "ES1"

The position (1 contract),of the order labeled as “ES1”,will be closed with a buy stop order at the price of the high of the bar plus 10 points. In this case we are limiting loses with a protection stop.

FeedFields

Description:
This function determines the information fields to which we want to obtain Access in real time using the method with the same name.

Syntax:
.FeedFields(Field)

Parameters:

Name	Default	Description
Field	FFLast	**FFAskSize**Amount of offered contracts **FFBidSize**Amount of requested contracts. **FFBuy1**Price offered in the first buying position **FFBuyAgency**Buying Agency **FFBuyOrders**Selling Agency **FFDate**Date **FFDecimals**Decimales **FFDescription**Symbol description **FFDiff**Difference **FFDiff_P**Difference % **FFExpiry_Date**Expiry date **FFHigh**High **FFISIN**ISIN code

		FFLast Last price **FFLastVol**Last Volume **FFLow**Low **FFMinimumMov**Tick **FFNumTrades**Number of trades **FFOpen**Open **FFOpenInterest**Open Interest **FFPrevious**Previous **FFSell1**Price offered in the first selling position **FFSellAgency**Selling Agency **FFSellOrders**Buy orders **FFSubMarket**Submarket the symbol belongs to **FFTime**Time **FFVolume**Volume

Example:

If we want to obtain the last closed trade we should previously define an exit variable:

Dim UltNegocio as Double

The value returned by this property will be ascribed to this variable:

UltNegocio = .FeedFields(FFLast)

FilledOrders

Description:
The property FilledOrders enables to find out if, over a certain bar, buy or sell active orders associated to a certain label have been filled. If so, the function will return 1 if not 0.

Syntax:
.FilledOrders(Label, Side, BarsAgo)

Parameters:

Name	Default	Description
Label	-	Label associated to the order that we are willing to check.
Side	-	Position of the order to be checked. (0 buy position; 1 sell position)
BarsAgo	0	Bar from which we are filling to obtain the date. The Default value is the current.

Example 1:

.FilledOrders(“C1”, 0, 10)

If a buy order labeled as “C1” has been filled 10 bars backwards, the function will return 1.

Example 2:

In a system, we launch three buy orders at stop with the labels A, B and C.

In the following bar, we are willing to know how many orders have been executed.

to do so we proceed as follows:

We define an orders counter

Dim OrdersCounter As Integer

```
If .FilledOrders("A",0 , 0) =1 then
     OrdersCounter = 1
End If

If .FilledOrders("B",0 , 0) = 1 then
OrdersCounter = OrdersCounter +1
End If

If .FilledOrders("C",0 , 0) =1 then
     OrdersCounter = OrdersCounter +1
End If
```

Imagine that the three orders are "At Close" orders.

In the following bar, if we want to know how many orders have been filled, we will ask the question in relation to the previous bar, as the orders were ATCLOSE, and should have been executed iin the previous bar:

```
If .FilledOrders("A",0 , 1) =1 then
OrdersCounter = 1
End If

If .FilledOrders("B",0 , 1) = 1 then
OrdersCounter = OrdersCounter +1
End If

If .FilledOrders("C",0 , 1) =1 then
      OrdersCounter = OrdersCounter +1
End If
```

GetBackGroundColor

Description:
This function returns the background color of a certain bar.

Syntax:
.GetBackGrounColor (BarsAgo)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards. The value 0 refers to the current bar.

Example:

.GetBackGroundColor(0)   Returns the value of the window background color for the current bar.

GetBarColor

Description:
This function returns the color of the data line for a certain bar.

Syntax:
.GetBarColor (BarsAgo, Line)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards, 0 corresponds to the current bar.
Line	-	Line in the current bar whose color we are willing to obtain.

Example:

.GetBarColor(1,3)	Returns the color in the previous bar for the data line number 3.

GetBarsSinceEntry

Description:
This function is used to find out the number of bars completed since a certain position was opened (long or short).If no position has been opened it will return 0 as a result.

Syntax:
.GetBarsSinceEntry(EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards. The Default value refers to the current position.

Examples:

.GetBarsSinceEntry (0)	Returns the number of bars formed since the currrent position was opened.

GetBarsSinceExit

Description:
This function is used to know the number of bars completed since a certain position was opened.If we have not exit any position the function returns the value 0.

Syntax:
.GetBarsSinceExit(EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards. The default value refers to the current positions.

Examples:

.GetBarsSinceExit (1)	Returns the number of bars formed since the previous trade.

GetBarStyle

Description:
This function returns the style of the line used in a certain bar.

Syntax:
.GetBarStyle(BarsAgo, Line)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Line	-	Identifies the line to which the bar from which we are willing to obtain the style belongs.

Example:

.GetBarStyle (0,1)

It returns the style of the line in the current bar for line number 1. The values that can be returned by the function are:

- **IsSolid:**Full line.
- **IsDash:**Dashed line.
- **IsDot:**Dotte line.
- **IsDashDot:**Dashed line with point.
- **IsDashDotDot:**Dashed line with two points.

GetBarWidth

Description:
Returns the width of a certain line for the indicated bar.

Syntax:
.GetBarWidth(BarsAgo, Line)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Line		Identifies the line to which the bar from which we are willing to obtain the width belongs.

Example:

.GetBarWidth(0,1) The function returns the width (1,2,…) of line 1 inthe current bar.

GetConfigStk

Description:
This procedure returns the initial properties of the statistics in current use.

Syntax:
.GetConfigStk(Sing, Unit, Filt, CompType,Compressión, BeginDate, EndDate)

Parameters:

Name	Default	Description
Sing	-	Result filter depending of the trade results; winnners (ssWins) orlosers (ssLoss).
Unit	-	Representation of the figures in cash (suMoney), in points (suPnts) or in

		percentage (suPortc).
Filt	-	Result filter by position, long (sfLong) or short (sfShort).
CompType	-	Representation of the figures grouped by trades (sctTrades), days (sctDays),weeks (sctWeeks), months (scMoths) or years (sctYears).
Compressión	-	Compression units.
BeginDate	-	Start date for the statistics time interval.
EndDate	-	End date for the statistics time interval.

Example:
Lest´s suppose that we have defined the following start variables:

```
Dim Signo As StatisticSign
Dim Unidad As StatisticUnit
Dim Filter As StatisticFilt
Dim TComp As StatisticCompType
Dim Compression As Long
Dim DateStart As Date
Dim DateEnd As Date
```

When calling the propertyGetConfigStk:

```
.GetConfigStk(Sign, Unit, Filter, TComp, Compression, DateStart , DateEnd)
```

Each of the variables previously defined will be filled with the figures returned by GetConfigStk.

Following with the example given for the propertyConfigStk, when using GetConfigStk, we will get the following information as a result:

```
Sign = ssLoss = 2       Filter = sfShort = 2     Compression = 1            DateEnd = 09/08/2010
Unit = suMoney = 0      Tcomp = sctDays = 3      DateStart = 09/08/2009
```

GetDailyLosers

Description:
This function returns the amount of losing trades between two given dates. It will be always compared with the date of the trade start.

Syntax:
.GetDailyLosers(FromDate, ToDate)

Parameters:

Name	Default	Description
FromDate	-	Start date for the interval to be analyzed.
ToDate	-1	End date for the interval to be analyzed.If no value is specified or -1 is assigned the end of the interval will be the current .

Example:

If we want to know the number of losing trades between two dates 10/06/2010 y 10/09/2010,first, we define the variable to which we will assign the number of losing trades:

Dim LosingTrades As Long

We assign the value return by the function the following way:
Losing trades = .GetDailyLosers(20100610,20100910)

GetDailyWinners

Description:
This function returns the amount of winning trades between two given dates. It will be always compared with the date of the trade start.
Syntax:
.GetDailyWinners(FromDate, ToDate)

Parameters:

Name	Default	Description
FromDate	-	Start date for the interval to be analyzed.
ToDate	-1	End date for the interval to be analyzed. If no value is specified or -1 is assigned the end of the interval will be the current .

Example:

If we want to know the number of winning trades between two dates 10/06/2010 y 10/09/2010.
First, we define the variable to which we will assign the number of losing trades:
Dim LosingTrades As Long

We assign the value return by the function the following way::
WiningTrades = .GetDailyWinners(20100610,20100910)

GetEntryDate

Description:
This function is used to find out the date on which a position has been opened. The format of the date returned is military(**AAAAMMDD).**Therefore, if the position was opened on June 25th, the indicated function will return the numerical value 20100625.If no position has been started, it will return 0.

Syntax:
.GetEntryDate(EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards.The value Defaultrefers to the current position.

Examples:

.GetEntryDate(0) The function returns the date on which the current position was opened.

.GetEntryDate(5) The function returns the date on which the fifth position ago was opened.

GetEntryPrice

Description:
This function is used to find out the price at which a position has been opened.
It returns the entry price of the position indicated in the parameter EntryAgo. If no position has been found, it will return 0.

Syntax:
.GetEntryPrice(EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards that we are willing to check.The value default refers to the current position.

Examples:

.GetEntryPrice(0)   The function returns the enry price of the current position.

.GetEntryPrice(5)   The function returns the enry price of the trade 5 trades ago.

▣GetEntryTime

Description:
This function is used to know the time at which a position has been opened. This information will be returned under 24 hours military format**(HHMM),**so thet, if the time is 5:35 pm it will be considered as the numerical format 1735.If no position has been started the function will return the value 0.

Syntax:
.GetEntryTime(EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of position backwards that we are willing to checkr. The valueDefault refers to the current position.

Examples:

.GetEntryTime(0) the function returns the entry time of the current position.

.GetEntryTime(4)   The function returns the entry time four positions backwards.

▣GetExitDate

Description:
This function is used to know the data on which a position has been closed. The format on which the date is returned is military format(**AAAAMMDD).**If the position was closed on June 25th, the indicated function will return the numerical value 20100625.If the position has not been closed yet the function will return 0.

Syntax:
.GetExitDate(EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards.The value default refers to the current positionúmero de posiciones hacia atrás.

Notes:
We must take into account that, at all effects, the last opened trade is considered to be closed in the current bar (bar on which the calculations are being made). Thus, if the value 0 is insicated in the parameter**EntryAgo**, this function will return the date of the current bar.

Therefore, if only the functions BuyandSell are used in the strategy, the value 1 applied to the parameter**EntryAgo**will return the date of the last closed operation before the current one. While the value 0 will always return the date of the current bar, indicating that there is currently a position opened.

But if we also use the functionsExitLong andExitShort, the case can occur where there is no position opened in the current bar. In these cases, the value 0 applied to the parameter**EntryAgo**will not return anything, as there is no position opened,while the value 1 will return the data when the last position was closed.

Example:

.GetExitDate(3) The function will return the exit date three trades backwards.

GetExitOrder

Description:
Specified if the nth order given over a certain bar for a Data (System type) is an exit order or not. If it has been an exit order, the function returns **True**.On the other hand if the order has been an exit order the cuntion will return **False**.

Syntax:
.GetExitOrder(Identifier, BarsAgo, NumOrder)

Parameters:

Name	Default	Description
Identifier	Data	Data series from which we will obtain the information.It must be a data associated to a system called via the methodGetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to obtain the data.The default value refers to the current bar.
NumOrder	-	Position of the order from which we are willing to obtain the information.

Example:

Let´s imagen that in the system "MiSistema" we proceeded the following way in a certain bar:

1) We have entered with an order .Buy labelled as "Buy _1"

2) The system prepares three new orders:

   a. .ExitLong atlimit 1, .GetEntryPrice + 50, "Buy_1"
   b. .ExitLong atstop, 1, .GetEntryPrice - 20, "Buy_1"
   c. .Sell atstop, 1, .Low – 100, "Sell_1"

We have also created a variale DataIdentifier type that refers to MySystem and called llamado MySystemData.

In this second develpment we define a Boolean variable enabling us to know i fan order from the system MySystem is an exit order or not.

We define the variable:

Dim Exit As Boolean

And we assign to it the result of the function GetExitOrder for the case of the current bar and referring to the second position:

Exit = .GetExitOrder(MySystemData, 0,1)

For the current bar, we were referring to initially, the variable Exit wil return True, as in this bar the second order launched from MySystem was an exit order.

GetExitPrice

Description:
This function is used to know the price at which a position has been closed. If no position has been closed it will return 0.

Syntax:
.GetExitPrice(EntryAgo)
Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards that we are willing to double check. The value default refers to the current position.

It is important to take into account the observations of the function GetExitDate.

Example:

.GetExitPrice(5) The function returns the exit price 5 trades ago.

GetExitTime

Description:

This function is used to know the time of a position closing. The time will be returned under 24 hours military format**(HHMM),**so that if the time is 5:35 pm we will consider it as the numerical value 1735. If no position has been closed, the function will return the value 0.

Syntax:
.GetExitTime(EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions baskwards that we are willing to check. The value por Defaultrefers to the current position.

Example:

.GetExitTime(3)   The function returns the exit time three trades backwards.

It is very important to take into account the observations of the functionGetExitDate.

GetHighest

Description:
This function is used to obtain the highest value of the last **n** bars.

Syntax:
.GetHighest(Identifier, TPrice, Length)

Parameters:

Name	Default	Description
Identifier	Data	Source Identifier. Any data source (high, low, close, indicators…). If there is more than one data source in the same window, they will be identified as Data1, Data2, Data3, etc.

TPrice	PriceClose	Field of the bar we want to refer to.To do so we must indicate in this field any of the following values: **PriceHigh**. **PriceLow** **PriceOpen** **PriceClose** **PriceVolume** If this function is calculated with an indicator as data source, we will use as parameter in **TPrice**the value**PriceClose**, referring to the close of the identifier data series. On the other hand if we indicated**PriceHigh** or any other it would not make any difference as it would always return the same value.
Length	1	Number of bars backwards to be considered.Any numerical type variable can be used to replace a number.

Examples:

.GetHighest(Data,PriceHigh, 10)

The function returns the numerical value of the highest high of the last 10 bars and for the indicated data series (Data).

.GetHighest(Data1,PriceVolume,100)

The function returns the numerical value of the highest volumen over the last 100 bars and for the indicated data series (Data1).

GetHighestBar

Description:
This function returns the number of the bar on which the highest data of a series is produced.

Syntax:
.GetHighestBar(Data,TPrice, Leght)

Parameters:

Name	Default	Description
Identifier	Data	Source Identifier. Any data source (high, low, close, indicators…). If there is more than one data source in the same window, they will be identified as Data1, Data2, Data3, etc.
Tprice	PriceClose	Field of the bar we want to refer to. To do so we must indicate in this field any of the following values: **PriceHigh**. **PriceLow** **PriceOpen** **PriceClose** **PriceVolume** If this function is calculated with an indicator as data source, we will use as parameter in **TPrice** the value **PriceClose**, referring to the close of the identifier data series. On the other hand if we indicated **PriceHigh** or any other it would not make any difference as it would always return the same value.
Leght	2	Number of bars backwards to be considered. Any numerical type variable can be used to replace a number.

Example:

.GetHighestBar(Data1, PriceLow, 20)

The function returns the bar number (backwards) where the higuest low of the last 20 bars of the indicated data series (Data1). is obtained

GetHistogramBand

Description:
This function returns the band line number ascribed to the line specified in the parameter with the same name.

Syntax:
.GetHistogramBand(Line)

Parameters:

Name	Default	Description
Line	-	Identifies the data line whose ban line number we are willing to know.

Example:

Imagine that we want to create an indicator and we are representing it the following way:

```
.SetIndicatorValue x, 1
.SetIndicatorValue 50, 2
.SetBarRepresentation 0,1,irHistogram
.SetHistogramBand 1, 2
```

If we define a start variable:

Dim MyBandLine

To which we can assign the value returned by the function:

MyBandLine = .GetHistogramBand(1)   The function will return the value 2 (the band line).

GetIndicatorIdentifier - GII

Description:
This function is used to create the data series corresponding to any indicator and to obtain an identifier of this series. To do so, we need to declare first a variable DataIdentifier type.

Once the variable is defined we will always assign to it the the value of the function**GetIndicatorIdentifier**in order to create the indicator data series and to obtain an identifier of the same indicator.
The identifier of the indicator must be obtained from the procedure OnInitCalculate.

Later on, in order to obtain the value of an indicator we must use the functionGetIndicatorValueand indicate in the paramter **Data**the variable on which we have saved the value of the correponding indicator.

The identifier obtained by this function can be use don any VBA function on which a Data is required (Data series on which the different functions are calculated).

Syntax:
.GetIndicatorIdentifier(Name, ParentDataIdentifier, ParamArray())

We can also use the short method**GII**:

.GII(Name, ParentDataIdentifier, ParamArray())

Parameters:

Name	Default	Description
Name	-	Indicator code.
ParentDataIdentifier	-	Identifier of the series on which the indicator is calculated. If we set this parameter as data, we will be calculating the indicator on the data or series on which the strategy is being calculated. If we are willing to obtain the identifier of an indicator being calculated on another indicator we shall indicate withing this parameter the identifier of the indicator we are willing to use as calculation.
ParamArray	-	First parameter specific of the indicator.It represents a group of parameters which their number is not specified as each of the indicators has a variable number. (a moving average has 2 while an RSI has 3). The order on which the parameters must be specified is the same as this indicators figure out on the dialog box showing up when we are ready to plot the indicator into a chart. If the parameter is **Price**type, it refers to one of the fields of the bar (Close, Open, etc.),as it would be the case of a moving average as its second parameter is the data source. In these cases we must specify the field of the bar on which we are going to calculate the indicator. We shall indicate any of the following constants equivalent to those fields: **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:** **PriceVolume:**
ParamArray	-	Second parameter specific of the indicator.
...	-	...
n-ésimo ParamArray	-	n-th parameter specific to the indicator.

Example:

.GetIndicatorIdentifier(RSI,Data,14,70, 30)

The source returns the indicator identifier RSI (14,70 and 30 are the indicator parameters).

GetIndicatorPos

Description:
This function obtains the trend of the indicator within a certain bar.

Syntax:
.GetIndicatorPos(BarsAgo, Line)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.
Line	-	Identifies the line, the bar we are willing to obtain the trend from belongs to.

Example:

.GetIndicatorPos(0,1)

Returns the trend, in the current bar, of the indicator line number 1.This trend can be:

- **ipBull**
- **ipBear**
- **ipNeutral**

GetIndicatorValue -GIV

This function is used to obtain the value of an indicator. To do so we must have previously determined the identifier of the indicator in the procedure InitCalculate via the functionGetIndicatorIdentifier.

Syntax:
.GetIndicatorValue(Identifier, BarsAgo, LineNumber)

We can also use the abbreviation**GIV**

.GIV(Identifier, BarsAgo,LineNumber)

Parameters:

Name	**Default**	**Description**
Identifier	-	Identifier of the indicator.
BarsAgo	0	Represents the number of bars backwards we are referring to in order to obtain the value of the indicator. If we are using a moving average, a value equal to 0 for this parameter, will forcé the function to return the value of the average in the current bar. A value equal to 1 will return the value of the average a bar ago and so forth and so on.
LineNumber	1	Line of the indicator to be obtained.Some indicators have more than one dat line. In these cases if we give to this parameter a value 1 it will return a value referring to the first data line, 2 for the second and so forth and so on.

Example:

.GetIndicatorValue(RSI, 0, 1)Returns the value of the first line of the indicator in the current bar.

GetLineName

Description:

This function returns the Nameof the indicated data line.

Syntax:

.GetLineName(Line)

Parameters:

Name	**Default**	**Description**
Line	1	Identifies the line from which we are willing to obtain the Name.

Example:

.GetLineName(2)

If we were programming an indicator and had assigned to line number 2 theName “UpperBand”, the function .GetLineName(2) will return “UpperBand”.

GetLowest

Description:
This function is used to obtain the lowest value of the last n bars of a data series.

Syntax:
.GetLowest (Identifier, TPrice, Length)

Parameters:

Name	Default	Description
Identifier	Data	Source identifier. Any data series (high, low, closes indicators …).If there is more than 1 data source inserted in the same window they will be noted as Data1, Data2, Data3, etc.
TPrice	PriceClose	Field of the bar to which we want to refer. To do so, we must indicate any of the following values: **PriceHigh:** **PriceLow:** **PriceOpen:**. **PriceClose:** **PriceVolume:** If this function is calculated using an indicator as data source we will use as parameter in **TPrice**the value **PriceClose**, that refers to the data series of the identifier. If we indicated **PriceHigh** or any other we still get the same value as return.
Length	1	Number of bars backwards to consider. Any numerical type variable can be used instead of the number.

Example:

.GetLowest (Data, PriceLow, 10) Returns the lowest price of the 10 latest barsof the series (Data).

GetLowestBar

Description:
Returns the number of the bar where the lowest value of a series is produced.

Syntax:
.GetLowestBar (Data,TPrice, Leght)

Parameters:

Name	Default	Description
Identifier	Data	Source identifier. Any data series (high, low, closes indicators …). If there is more than 1 data source inserted in the same window they will be noted as Data1, Data2, Data3, etc.
Tprice	PriceClose	Field of the bar to which we want to refer. To do so, we must indicate any of the following values: **PriceHigh:** **PriceLow:** **PriceOpen:**. **PriceClose:** **PriceVolume:** If this function is calculated using an indicator as data source we will use as parameter in **TPrice** the value **PriceClose**, that refers to the data series of the identifier. If we indicated **PriceHigh** or any other we still get the same value as return.
Leght	2	Number of bars backwards to consider.

Example:

.GetLowestBar(Data1, PriceVolume, 20)

The function returns the number of the bar (backwards) where te lowest volumen of the latest 20 bars of the indicated data series is produces (Data1).

GetMarketPosition

Description:
This function is used to know, while the system is being calculated, which is our position in the market, long, short or flat. This function is very useful if we are working with stop or limit orders as we do not know when they have been filled.

Syntax:
.GetMarketPosition (EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of trades backwards, by Default we are using the current position.

Example:

.GetMarketPosition(0)

The function returns the current position.1 for long, 0 for flat and -1 for short.

GetMaxContracts

Description:
This function is used to know the máximum amount of contratcts (longo r short) negotiated within a single position.

Syntax:
.GetMaxContracts (EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards, by Default we are using the current position.

Example:
.GetMaxContracts(3)The function returns the number of contracts bought or sold three trades ago.

GetMaxEntries

Description:
This function is used to know the máximum amount of different entries within a single position. A position can have different entries according to the label established in the order and the orders matching modality.

Syntax:
.GetMaxEntries (EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards, by Default we are using the current position

Example:

.GetMaxEntries(5) The function returns the highest number of entries 5 trades backwards.

GetNthHighest

Description:
This function is used to obtain the highest value of the last n bars and with a certain order.

Syntax:
.GetNthHighest (Identifier, Nth, TPrice, Length)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Source indentifier. Any data series (high, low, close, indicators …). If there is more than 1 data source in the window they will be codified as Data1, Data2, Data3, etc.
Nth	1	This is the ordinal representing the value that we are willing to obtain. If Nth is worth 1, we will obtain the first highest value of the last **n**bars of the series, if Nth is worth 2 we will obtain the second highest value of the series and so foth and so on.
TPrice	PriceClose	Field of the bar to which we are willing to refer. To do so we must indicate in this field any of the following values: **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:** **PriceVolume:** If w calculate this function with an indicator as data source, we will pass as parameter in **TPrice**the value**PriceClose**, that refers to the close of the indicator data series. If we´d indicated**PriceHigh** or any other it will still return the same value.
Length	50	Number of bars backwards to be considered.

Example:

.GetNthHighest(Data, 3, PriceHigh, 10)Returns the third highest high of the last 10 bars.

GetNthLowest

Description:
This function is used to obtain the lowest value of the last **n** bars and with a certain order.

Syntax:
.GetNthLowest(Identifier, Nth, TPrice, Length)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Source indentifier. Any data series (high, low, close, indicators …). If there is more than 1 data source in the window they will be codified as Data1, Data2, Data3, etc.
Nth	1	This is the ordinal representing the value that we are willing to obtain. If Nth is worth 1, we will obtain the first lowest value of the last **n** bars of the series, if Nth is worth 2 we will obtain the second lowest value of the series and so foth and so on.
TPrice	PriceClose	Field of the bar to which we are willing to refer. To do so we must indicate in this field any of the following values: **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:** **PriceVolume:** If w calculate this function with an indicator as data source, we will pass as parameter in **TPrice** the value **PriceClose**, that refers to the close of the indicator data series. If we´d indicated **PriceHigh** or any other it will still return

		the same value.
Length	50	Number of bars backwards to be considered.

Example:

.GetNthLowest (Data, 2, PriceLow, 10)   Returns the scond lowest low of the last 10 bars.

GetOrderCount

Description:
Returns the amount of active orders given in a single bar.

Syntax:
.GetOrderCount(Identifier, BarsAgo)

Parameters:

Name	Default	Description
Identifier	-	Data series from which we obtain the information. It must be a data associated to a system called with the methodGetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to extract the amount of orders. The default value refers to the current bar.

Example:

Assuming that th system to which we are referring from identifier in the current bar sends to the market the following orders:

- An order to change the position in to short as the system is long
- An profit exit order
- A stop loss order

We can assign to a variable previously defined the value returned by this function:

Dim OrdersNum as Integer

OrdersNum = .GetOrderCount(MiSistemaData, 0)

The function returns the value 3 in the current bar as this is the number of active orders on it.

GetOrderDate

Description:
This function returns the data ascribed to the nth active order of a system in a certain bar.

Syntax:
.GetOrderDate(Identifier, BarsAgo, NumberOrder)

Parameters:

Name	Default	Description
Identifier	Data	Data series from which we obtain the information. It must be a data associated to a system called with the method GetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to extract the order. The default value refers to the current bar.
NumberOrder	0	Number of reference order ascribed to the order from which we are willing to obtain the information. The value Defaultrefers to the first active order in a certain bar.

Example:

With the example used in the previous function.GetOrderCount, if we only want to know the date of the latest of the three orders, we will proceed as follows:

Dim FechaUlOrden as Date

FechaUlOrden = .GetOrderDate(MySystemData,0, 2)

The function referst to a DD/MM/YYYY date type. The date returned will bte the corresponding to the current bar (in the case of BarsAgo=0), or to the date of the bar we are referring to (in the case of BarsAgo >0)

The position we are referring to in this case is 2.As indicated in the section Parameters of this function, the value 0 is ascribed to the first order, the value 1 to the second and so foth and so on.

GetOrderLabel

Description:
This function returns the label assigned to the nth active order of a system in a certain bar.

Syntax:
.GetOrderLabel(Identifier, BarsAgo, NumberOrder)

Parameters:

Name	Default	Description
Identifier	Data	Data series from which we obtain the information. It must be a data associated to a system called with the method GetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to extract the label. The default value refers to the current bar.
NumberOrder	0	Number of reference order ascribed to the order from which we are willing to obtain the information. The value Default refers to the first active order in a certain bar.

Example:

Assuming that in the system "MySystem" we follow this procedure in a certain bar:

1) We have trades with an order .Buy labeled as "Buy_1"
2) El sistema prepara tres órdenes nuevas:
    a. .ExitLong atlimit 1, .GetEntryPrice + 50, "Buy_1"
    b. .ExitLong atstop, 1, .GetEntryPrice - 20, "Buy_1"
    c. .Sell atstop, 1, .Low – 100, "Sell_1"

Starting from this, if from a second system (of course we must have created a DataIdentifier using "MySistema"), we indicate the following:

Dim Label as String

Label = .GetOrderLabel(MySystemaData, 0, 1)

In label, we will obtain "Buy_1", which is the label assignated to the second order of "MySystemaData".

GetOrderPrice

Description:
This function returns the price of the nth active order of a system given in a certain bar.

Syntax:
.GetOrderPrice(Identifier, BarsAgo, NumberOrder)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Data series from which we obtain the information. It must be a data associated to a system called with the method GetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to extract the label. The default value refers to the current bar.
NumberOrder	0	Number of reference order ascribed to the order from which we are willing to obtain the information. The value Default refers to the first active order in a certain bar.

Example:

By keep going with the example used for the function.GetOrderCount, if, from the three orders in teh current bar, we are willing to know the price of the first of them, we can define a Double type variable, to which we will assign the value returned by the function.GetOrderPrice:

Dim Price as Double

Price = .GetOrderPrice(My SystemData, 0, 0)

GetOrderSide

Description:
This function returns the position (long or short) of the nth active order of a system given in a certain bar.

Syntax:
.GetOrderside(Identifier, BarsAgo, NumberOrder)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Data series from which we obtain the information. It must be a data associated to a system called with the method GetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to extract the symbol of the associated value. The default value refers to the current bar.
NumberOrder	0	Number of reference order ascribed to the order from which we are willing to obtain the information. The value Default refers to the first active order in a certain bar.

Example:

By keep going with the example for the function .GetOrderLabel, if, from the three orders in the current bar, we are willing to know the side of the second of them, we can define an OrderSide type variable (osBuy, osSell), to which we will assign the value returned by the function .GetOrderside:

Dim Pos as OrderSide

Pos = .GetOrderSide(MySystemData, 0, 1)

The function will return OsSell, as the second order is a sell stop order.

GetOrderSymbolCode

Description:
Returns the code(in Visual Chart format, p.e 010072MFXI) of the value associated to the nth order of a system given in a certain bar.

Syntax:
.GetOrderSymbolCode(Identifier, BarsAgo, NumberOrder)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Data series from which we obtain the information. It must be a data associated to a system called with the method GetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to extract the symbol of the associated value. The default value refers to the current bar.
NumberOrder	0	Number of reference order ascribed to the order from which we are willing to obtain the information. The value Default refers to the first active order in a certain bar.

Example:

Assuming that we are working in the DAX futures with the system used for the function.GetOrderLabel. If we want to obtian the symbol associated to the third order in the current bar , we can define a variable and assign to it the code of the value. This would be the procedure:

Dim Symbol as String

Symbol=.GetOrderSymbolCode(MySiytemData,0,2)

The function will return "010015DX" which is the code of the Future Dax in Visual Chart format.

GetOrderType

Description:
This function returns the type of order (stop, limit, at close) of the nth active order of a system.

Syntax:
.GetOrderType(Identifier, BarsAgo, NumberOrder)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Data series from which we obtain the information. It must be a data associated to a system called with the method GetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to extract the type of order. The default value refers to the current bar.
NumberOrder	0	Number of reference order ascribed to the order from which we are willing to obtain the information. The value Default refers to the first active order in a certain bar.

Example:

Following with the example used for the function.GetOrderLabel, if we are willing to know the type of order associated to the first of them in the previous bar, we could define a TradeType variable (AtClose, AtLimit, AtStop, AtMarket) and assign to it the value returned by this variable:

Dim OrderType as TradeType

OrderType =.GetOrderType(MYSystemData,1,0)   The function will return AtLimit.

GetOrderVolume

Description:
This fucntion returns the amont of contracts /shares of the nthactive order of a system given in a certain bar.

Syntax:
.GetOrderVolume(Identifier, BarsAgo, NumberOrder)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Data series from which we obtain the information. It must be a data associated to a system called with the method GetSystemIdentifier.
BarsAgo	0	Bar from which we are willing to extract the type of order. The default value refers to the current bar.
NumberOrder	0	Number of reference order ascribed to the order from which we are willing to obtain the information. The value Default refers to the first active order in a certain bar.

Example:

Following with the example used for the function .GetOrderLabel, if we want to know the volumen of contracts/stocks negotiated in the second order of the current bar, we could defined a variable to assign to it the value returned by the function.

Dim VolumeOrden as Long

VolumeOrden =.GetOrderVolume(MySystemData,0,1)   the function will return 1.

GetPivotDow

Description:
This function is used to obtain the value of a pivot. A pivot is a peak in the quote, in this case we could consider it as a support.
Syntax:
.GetPivotDown(Identifier, Occur, TPrice, LeftCount, RightCount, Length)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Data series from which we will obtain the pivot (chart, indicators...).
Occur	1	Numerical value representing the number of the pivot backwards we are willing to obtain. If Occur is worth 1, we will be otaining the first pivot from the current bar, if it is worth 2 , the second one and so foth and so on.
TPrice	PriceClose	Field of the series from which we are willing to obtain the pivot. **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:** **PriceVolume:** If we calculate this function on an indicator we will pass as parameterin **TPrice**the value**PriceClose**, that refers to the close of the data series of the identifier. If we´d indicated **PriceHigh** or any other value if will not make any change as it will always return the same value.
LeftCount	-	Number of bars in the left side of the pivot.
RightCount	-	Number of bars in the right side of the pivot.
Length	50	Number of bars backwards to be considered while searching for the pivot.

Example:

.GetPivotDown(Data, 1, PriceLow, 2,4, 50)

The function will search in the latest 50 bars before the previous one, the value of the closer pivot (calculated on the lows), finding in the 2 bars on the left of the pivota and the 4 bars on the right, the value of the low superior to this one.

GetPivotUp

Description:
This function is used to obtain the value of a pivot. A pivot is a peak in the quote, in this case we could consider it as a resistance.

Syntax:
.GetPivotUp(Identifier, Occur, TPrice, LeftCount, RightCount, Length)

Parameters:

Name	Default	Description
Identifier	Data	Data series from which we will obtain the pivot (chart, indicators...).
Occur	1	Numerical value representing the number of the pivot backwards we are willing to obtain. If Occur is worth 1, we will be otaining the first pivot from the current bar, if it is worth 2 , the second one and so foth and so on.
TPrice	PriceClose	Field of the series from which we are willing to obtain the pivot. **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:** **PriceVolume:** If we calculate this function on an indicator we will pass as parameter in **TPrice** the value **PriceClose**, that refers to the close of the data series of the identifier. If we´d indicated **PriceHigh** or any other value if will not make any change as it will always return the same value.
LeftCount	-	Number of bars in the left side of the pivot.
RightCount	-	Number of bars in the right side of the pivot.
Length	50	Number of bars backwards to be considered while searching for the pivot.

Example:

.GetPivotUp(Data, 1, PriceHigh, 2,4, 50)

The function will search in the latest 50 bars before the previous one, the value of the closer pivot (calculated on the lows), finding in the 2 bars on the left of the pivota and the 4 bars on the right, the value of the low superior to this one.

GetPositionProfit

Description
This function is used to know the value (in points) of the profit obtained in a position. This functions considers the amount of contracts/stocks bought or sold. The value returned will be the difference between the close of the last bar and the entry point multiplied by the number of contracts sold.

Syntax:
.GetPositionProfit(EntryAgo)

Parameters:

Name	Default	Description
EntryAgo	0	Number of positions backwards.The value Default indicates the current position.

Example:

.GetPositionProfit(1)   Returns the profit obtained within the previous trade.

GetPrice

Description:
This function is used to know the price of a field of a bar belonging to a series.

Syntax:
.GetPrice(BarsAgo, Identifier)

Parameters:

Name	Default	Description
TPrice	PriceClose	Field of the bar from which we want to obtain the value. **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:** **PriceVolume:** **PriceOpInt:**
BarsAgo	0	Number of bars backwards
Identifier	Data	Data series to which we apply the function. If there is more than one element inserted they will be codified as Data1, Data2, Data3, etc.

Example:

.GetPrice(PriceHigh, 22, Data1) Returns the field High of a bars belonging to Data1 22 bars backwards.

GetStkLength

Description:
This function returns the total amount of values given for a certain type of statistical data.

Syntax:
.GetStkLength (Statistic)

Parameters:

Name	Default	Description
Statistic	-	Enables to select the statistical data from which we are willing to extract the value.

Example:

Inside a system, we want to know, at a certain stage,the lenght of the drawdawn, in fact, how many trades have been considered for the. We will proceed as follow:

Lenght_DD = .GetStkLength(svAvg_Drawdown)

Lenght_DD will return the amount of trades taken by the system to call the function and that have ben used to calculate the drawdawn.

GetStkValue

Description:
This function return the nth value of a certain statistical figure.

Syntax:
.GetStkValue(Statistic, Index)

Parameters:

Name	Default	Description
Statistic	-	Enables to select the statistical data from which we are willing to extract the value.

Index	-	Enables to select a certain positin within the data list, so that we can select the value given withing a certain positions.

Example:

We are working with a system and we want to know, at a certain stage, the drawdown of the system,so we could assign this value to a previously defined variable, the following way:

Actual_DD =.GetStkValue(svAvg_Drawdown)

In this case it will return the value of the Drawdown of this bar and at this moment.

GetStkValues

Description:
This function return the total group of values for a certain statistical data.

Syntax:
.GetStkValues(Statistic, aValues)

Parameters:

Name	Default	Description
Statistic	-	Enables to select the statistical data from which we are willing to extract the value.
aValues	-	Buffer where the values returned for the selected statistical data are stocked.

Example:

```
Dim Amount as long
Dim Values() as Double

Cantidad = .GetStkValues(svDrawdown,Values)
```

In this case, it returns the array **Values**for all the figures and in **Amount** the number of values written in the array.

GetSymbolIdentifier-GSI

Description:
This function is used only when we need to obtain and use data from a symbol which its chart is not on screen while inserting the corresponding system as , is this was the case, it would be easier to use two **DATAS**.

This function is also useful to refer to the values of a sysmbol determined within a Macro, as in this case the historical data are not available on screen.

In order to create a data source and to obtain its identifier we must previously have declared a DataIdentifier type variable.

Once the variable has been defined we will assign to it the value of the function**GetSymbolIdentifier**in order to obtain the indentifier of the symbol. The identifier of the symbol must be obtained from the procedure OnInitCalculate.

Later on, the obtained identifier cas be used with any VBA function requiring a Data (data series from which the different functions are calculated).

Syntax:
.GetSymbolIdentifier(Symbol, Compresion, Cr, FromDate, ToFinalDate)

We can also use its short name**GSI:**

.GSI(Symbol, Compresion, Cr, FromDate, ToFinalDate)

Parameters:

Name	**Default**	**Description**
Symbol	-	Code of the required symbol.
Compresion	-	Compression unit (2, 5 ,10...).
Cr	-	Compression type; they are four different types available: **CrMinutes:**To obtain a minutes chart. **CrDays:**To obtain a daily chart. **CrWeeks:**To obtain a weekly chart. **CrMonth:**To obtain a monthly chart.
FromDate	-	Start date if the data source with the required identifier
ToFinalDate	-	This is the end date of the historical data we are going to load. This data must always be superior to the current date so we recommend to always use 01/01/2037 to make sure that the data of the required source are always updated.

Example:

.GetSymbolIdentifier(010072MFXI, 5, CrMinutes, Date-30, Date)

The function used the last 30 days of the historical data of the l IBEX(MFXI) future in en 5 minutes.

GetSymbolInfo

Description:
This function is used to obtain a series of characteristics from a certain symbol and not only for a certain bar. These values remain constant all over the chart and are determined by the type of value**SymbolInfo**, that can take the following values:

SbiBarCompresion	Compression used for the bars
SbiCode	Code of the asset.
SbiCompresion	Compression being used (ticks, minutes, days,…)
SbiFirstSessionEnd	Closing time for the session of the asset (formatHHMM).
SbiFirstSession Start	Start time for the session of the asset (format HHMM).
SbiMarket	Market the product belongs to
SbiMinMov	Minimum movement (tick) of the product.
SbiName	Nameof the product.
SbiNumDec	Number of decimals considered in the scale of the product.
SbiPath	All the symbols registered in our computer are located in the folderVisualChart\RealServer\Data. With this figure we can extract the path from the folder DATA of a certain asset.
SbiPointValue	Value per point of the product.
SbiTimeD if	Time difference of the producto n which it is applicated (in seconds)
SbiVendor	Specifies the vendor of the asset.

Syntax:
.GetSymbolInfo(Info, Data, Day)

Parameters:

Name	**Default**	**Description**
Info	SbiName	Type of SymbolInfo that we are willing to check.
Symbol	Data	Data source from which we want to extract the information. If the data source is not specified, the chart on which the system is inserted will be taken.
Day	-	Day of the week from which we want to obtain the information. Some values can change depending on the day of the.(not compulsory)

Examples:

.GetSymbolInfo(SbiTimeDif)

If we are applying the function to e-mini S&P future (7 hours time difference), this function will return 25200(seconds).
.GetSymbolInfo(SbiMinMov)

If the function is being applied to the Future DX(010015DX), it will return 0.5 that is its mínimum movement.

GetSymbolInfoEx

Description:
This property returns a structure with all the **SymbolInfo**data type related to the assigned data series.This property is used in order to avoid having to do more than one call to obtain all **SymbolInfo**figures.

Syntax:
.GetSymbolInfoEx(Symbol, Day)

Parameters:

Name	**Default**	**Description**
Symbol	Data	Data source from wich we are willing to extract the information. If not specified we wil considered the chart where the system is inserted as data.
Day	-	Optional parameter. Day of the week from which we want to obtain the information. Some values can change depending on the day of the week: **vbFriday:** **vbThursday:** **vbMonday:** **vbTuestday:** **vbSaturday:** **vbWednesday:** **vbSunday:**

GetSystemIdentifier-GSYSI

Description:
This function enables to obtain internally, the information of a certain system. This way, we can extract the information from this system without having to calculate it once and once again.

Syntax:
.GetSystemIdentifier(Name, ParentDataIdentifier, ParamArray)

We can also use the short method **GSYSI.**

.GSYSI(Name, ParentDataIdentifier, ParamArray)

Parameters:

Name	Default	Description
Name	-	Code of the system from which we are willing to extract the information
ParentDataIdentifier	Data	Data source from which we would be loading the system
ParamArray	-	Colection of entry parameters demanded by the system (optional).

Example:

We can assign to a DataIdentifier type variable the information of a certain system and use it later on with other functions, for example, por Example, .GetOrderCount, .GetOrderLabel etc.

Dim MySystemData as DataIdentifier

MySystemData =.GetSystemIdentifier(MySystem, Data, NumberofContracts, target, StopLoss,Start time, EndTime)

Dim GOC As Long

GOC = .GetOrderCount(MySystemData)

GetSwingHigh

Description:
This function is similar to GetPivotUp. It is normally used to obtain the value of a pivot. A pivot is a peak in the quote. The pivots can be calculated on any data series, symbol or indicators. If no pivot is found the function will return 0.

Syntax:
.GetSwinHigh(Identifier, Occur, Tprice, Strength, Length)

Parameters:

Name	Default	Description
Identifier	Data	Data series from which we will obtain the pivot (quotes, indicators...).
Occur	1	Numerical value representing the number of the pivot backwards we are willing to obtain.If Occur is worth 1, we will obtain the first pivot from the current bar, if it is worth 2 the second one and so foth and so on.
TPrice	PriceClose	Data series from which we are willing to obtain the pivot. **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:** **PriceVolume:** If this fonction is calculated on an indicator we will pass as paramter in **TPrice**the value**PriceClose**, that refers to the close of the indicator data series. If we´d indicated **PriceHigh** or any other price source it will still return the same value.
Strength	2	Number of bars to consider in both sides of the pivot.
Length	50	Number of bars backwards to be considered while searching for the pivot.

This function is very useful as it helps us to find support or resistances. WE can say that there is an up pivot when a value in a certain data series is superior to a number of previous and subsequent values specified in the parameter Strength.

Example:

.GetSwingHigh(Data, 1, PriceHigh, 2, 50)

In this case the function will search in the 50 bars preceeding the current one, the value of the closest pivot calculated within the highss,being 2 bars on each side of the pivot, the value of the high lower than this pivot.

GetSwingHighBar

Description:
This function returns the lenght,in number of bars,from the last bullish pivot. A bullish pivot is a price higher than the previous prices backwards and subsequent prices forwards.

Syntax:
.GetSwinHighBar(Identifier, Occur, Tprice, Strength, Length)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Series from which we wil obtain the value of the pivot (quotes, indicators...).
Occur	1	Numerical value rpresentig the number of the pivot backwards that we are willing to find. If Occur is worth 1, we will be obtaining the first pivot from the current bar, if it is worth two, the second one and so foth and so.
TPrice	PriceClose	Data series from which we are willing to obtain the pivot. **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:** **PriceVolume:**. If this fonction is calculated on an indicator we will pass as paramter in **TPrice** the value **PriceClose**, that refers to the close of the indicator data series. If we´d indicated **PriceHigh** or any other price source it will still return the same value.
Strength	5	Number of bars to consider in both sides of the pivot.
Length	50	Number of bars backwards to be considered while searching for the pivot. If this number of vars is overcome, the function will return.

Example:

.GetSwingHighBar(Data, 1, PriceHigh, 2, 50)

In this case, the function will search over the 50 bars preceeding the current one, for the value of the closes pivot (Calculated on the highs), being 2 the bar son each side of the pivot, the value of the high superior to it. If this pivot is 5 bars away from the current bar (where we are using the function, it will return 5. In the following bar it will return 6 and so foth and so on until a new pivot is found.

GetSwingLow

Description:
This function is similar to GetPivotDown. It is also used to obtain the value of a pivot. A pivot is a peak in the quote, in this case we will be able to consider it as a support as we are talking about an inverted peak. The pivots can be calculated on any data series, symbol or indicators. If no pivot is found the function will return 0.

Syntax:
.GetSwingLow(Identifier, Occur, Tprice, Strength, Length)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Series from which we wil obtain the value of the pivot (quotes, indicators...).
Occur	1	Numerical value rpresentig the number of the pivot backwards that we are willing to find. If Occur is worth 1, we will be obtaining the first pivot from the current bar, if it is worth two, the second one and so foth and so.

TPrice	PriceClose	Field of the bar from which we are willing to obtain the pivot **PriceHigh:** **PriceLow:** **PriceOpen:**. **PriceClose:** **PriceVolume:** If this fonction is calculated on an indicator we will pass as paramter in **TPrice** the value **PriceClose**, that refers to the close of the indicator data series. If we´d indicated **PriceHigh** or any other price source it will still return the same value.
Strength	2	Number of bars to consider in both sides of the pivot
Length	50	Number of bars backwards to consider while searching for the pivot

This function is very useful as it helps us to find support or resistances. WE can say that there is an up pivot when a value in a certain data series is superior to a number of previous and subsequent values specified in the parameter Strength.

Example:

.GetSwingLow(Data, 1, PriceLow, 2, 50)

In this case, the function will search throughout the 50 bars preceeding the current one, the value of the closest pivot (calculated on the lows), the value of the low superior to this low is coming in the two bar son both sides of the pivot.

GetSwingLowBar

Description:
This function returns the lenght, in number of bars, since the last bearish pivot occurred.A bearish pivot is a low lower than the prices on its right side and its left side.

Syntax:
.GetSwingLowBar(Identifier, Occur, Tprice, Strength, Length)

Parameters:

Name	**Default**	**Description**
Identifier	Data	Series from which we wil obtain the value of the pivot (quotes, indicators...).
Occur	1	Numerical value rpresentig the number of the pivot backwards that we are willing to find. If Occur is worth 1, we will be obtaining the first pivot from the current bar, if it is worth two, the second one and so foth and so.
TPrice	PriceClose	Field of the bar from which we are willing to obtain the pivot **PriceHigh:** **PriceLow:** **PriceOpen:**. **PriceClose:** **PriceVolume:** If this fonction is calculated on an indicator we will pass as paramter in **TPrice** the value **PriceClose** that refers to the close of the indicator data series. If we´d indicated **PriceHigh** or any other price source it will still return the same value.
Strength	2	Number of bars to consider in both sides of the pivot

Example:

.GetSwingLow(Data, 1, PriceLow, 2, 50)

In this case, the function will search over the 50 bars preceeding the current one the value of the closest pivot (calculated on the lows). The lowest value superior to the pivot bar one must occur in the two bars before and after the pivot one. If the pivot is found 5 bars before the current one on which the function is used, it will return a 5, is 6 bars then a 6 and so foth and so on untill a new pivot is found

GetTrueHigh

Description:
Returns the highest price between the high of a bar and the close of the previous bar.

Syntax:
.GetTrueHigh (Identifier,BarsAgo)

Parameters:

Name	Default	Description
Identifier	Data	Data series on which we apply the calculation
BarsAgo	10	Number of bars backwards that will serve as reference to apply the calculation

Example:

.GetTrueHigh (Data, 1)

Compares in the historical figures of the series Data, the high of the bar with the close of the previous bar, indicating the higher of both of them 2 (taking as reference to start the calculation the bar preceeding the current one (1)).

GetTrueLow

Description:
Returns the lowest price between the low of a bar and the close of the previous bar.

Syntax:
.GetTrueLow (Identifier, BarsAgo)

Parameters:

Name	Default	Description
Identifier	Data	Data series on which we apply the calculation
BarsAgo	10	Number of bars backwards that will serve as reference to apply the calculation

Example:

.GetTrueLow (Data1, 5)

Compares in the historical figures of the series Data1,the low of a bar with the close of the previous one, indicating the lower of both of them (taking as reference to start the calculation 5 bars backwards).

GetTrueRange

Description:
This function returns the difference between GetTrueHighand.GetTrueLowfor a data series.

Syntax:
.GetTrueRange (Identifier, BarsAgo)

Parameters:

Name	Default	Description
Identifier	Data	Data series on which we apply the calculation
BarsAgo	10	Bar number on which the function is applied

Example:

Let´s create an indicator extracting the following data

Dim Difference As Double

Difference = .GetTrueRangeCustom(Data, 0, 10700, 10400)

If we insert the indicator in the IBEX FUT. CONTINUOUS, chart the difference will worth 300 as it is the difference between 10700 and 10400. However, for the bars overcoming this margin the difference will change. For Example, if the close is worth 10809, the high will be then 10809, which is the real high and in this case the difference will be 409.

GetTrueRangeCustom

Description:
This function returns the difference between an established highest value and an established lowest value. If the highest value is lowest than the lowest value the returned value will be 0.
Syntax:
.GetTrueRangeCustom (Identifier, BarsAgo, High, Low)

Parameters:

Name	Default	Description
Identifier	Data	Data series from which we are willing to extract the information
BarsAgo	10	Number of the reference bar
High	-	Higuest value
Low	-	Lowest value

Example:

.GetTrueRange (Data, 2)

Returns the difference between the values returned by the function GetTrueHigh and GetTrueLow in the second bar backwards in the series Data.

GetVolatility

Description:
This functions returns the volatility between the current bar and the Nth bar backwards in terms of difference in points.

Syntax:
.GetVolatility (Identifier, Tprice, Lenght)

Parameters:

Name	Default	Description
Identifier	Data	Data series on which the calculation is applied (quote, indicator..).
TPrice	PriceClose	Data field on which we are going to run the calculation. **PriceHigh:** **PriceLow:** **PriceOpen:** **PriceClose:**

		PriceVolume:
Lenght	1	Distance in relation with the current bar to calculate the volatility.

Example:

.GetVolatility(Data1,PriceLow, 5)

In this case, the function will return the difference between the low of the current bar and the low 5 bars backwards (from the data series Data 1).

GetWndBackGroundColor

Description:
This function returns the background color of an indicator window.

Syntax:
.GetWndBackGrounColor()

Parameters:

Name	Default	Description
-	-	-

GrossLoss

Description:
Returns th value of the net loses for the negative trades accumulated by our system until the current bar (on which the calculations are being run).To obtain it, we consider that the last opened trade concludes in the current bar.

Syntax:
.GrossLoss(Show As SttRepresentation)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to decide the format on which the result is goint to show up, **ByPoints** or**Percentage**.

Example:

.GrossLoss(Porcentual)   Will return the loss percentage (net) obtained by the system.

GrossProfit

Description:
This function returns the value of the net profits obtained by the system in the positive trades until the current bar (bar on which the calculations are being run). To obtain it we will consider that the last opened trade concludes in the current bar.

Syntax:
.GrossProfit(Show As SttRepresentation)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to decide the format on which the result is goint to show up, **ByPoints** or **Percentage**.

Example:

.GrossProfit(Puntos)   Will return in points the net profit obtained by the system.

High

Description:
This function returns the value of the bar high.

Syntax:
.High(BarsAgo, Identifier)

Parameters:

Name	**Default**	**Description**
BarsAgo	0	Number of bars backwardss.The value Defaultrefers to the current bar. In this parameter we can use any numerical value contained in a variable or enter the value directly. If can also be specified as function replacing the numerical.
Identifier	Data	Data source on which the function is applied. If they are several charts in the window they will be codified as Data1, Data2, Data3 etc.

Example:

.High(4, Data1) Will return the highest value of the las 4 bars (from the data source Data1).

LargestLosingTrade

Description:
This function returns the result of the worst operation, this value will change while new bars are generated by and its value will depend on the bar on which the property is called.

Syntax:
.LargestLosingTrade(Show)

Parameters:

Name	**Default**	**Description**
Show	Bypoints	Enables to decide the format on which the result is goint to show up, **ByPoints** or **Percentage**.

Example:

If we want to know, at a certain stage, the result of the worse operation (in points) done by our system we could define a variable:

Dim WorstTrade

An assign to it the value returned by the property:

WorstTrade =.LargestLosingTrade(ByPoints)

LargestWinningTrade

Description:
This function returns the result of the best trade made. This value will change while new bas are generated by and it value will depend on the bar on which the property is being called.

Syntax:
.LargestWinningTrade(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which the information will show up, **ByPoints**, or**Percentage**.

Example:

We are willing to know, at a certain stage, the result of the best trade in percentage ,made by our system so we could define a variable:

Dim BestTrade

And assign to it the value returned by the property:

BestTrade=.LargestWinningsTrade(Percentage)

Lc_Index

Description:
This function returns the ratio **Profits in long positions /Profits in short positions**.This value will change while new bars are generated by, and it value will dependo n the bar on which this property is called.

Syntax:
.Lc_Index(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which the information will show up, **ByPoints**, or **Percentage**.

Example:

We are willing to know, at a certain stage, the ratio **Profits in long positions /Profits in short positions**(percentage) of our system. We could define a variable:

Dim ratio_ganancia

And assign to it the value returned by the property:

ratio_profit=.Lc_Index(Percentage)

LimitOrder

Description:
This property enables to obtain the existing amount of orders in the bid and in the ask for certain prices levels in a certain bar. It only returns results in real time as they are not historical data available for the bid and ask.

Syntax:
.LimitOrder(Level, Side, BarsAgo, Identifier)

Parameters:

Name	**Default**	**Description**
Level	1	Indicates the nthbid and ask position whose amount we are willing to know.
Side	osBuy	Order type, osBuy (buy), osSell (sell).
BarsAgo	0	Only returns results in real time.
Identifier	Data	Data series from which we want to obtain the information

Example:

.LimitOrder(4, osSell, 0, Data1)

This way, we obtain the amount of titles/cotracts offered in the forth sell position of the data sourceData1.

LimitPrice

Description:
This property enables to obtain the price of the active orders in a bid and in the ask for a certain position and in a certain bar. It only returns results in real timeas they are not historical data available for the bid and ask.

Syntax:
.LimitePrice(Level, Side, BarsAgo, Identifier)

Parameters:

Name	**Default**	**Description**
Level	1	Indicates the nth bid and ask position whose price we are willing to know.
Side	osBuy	Position of the order , osBuy (buy), osSell (sell).
BarsAgo	0	Only returns results in real time.
Identifier	Data	Data from which we are extracting the information

Example:

.LimitOrder(4, osSell, 0, Data1)   Returns the price ofered in the forth sell positions of Data1.

LimitVol

Description:
The property LimitVol enables to obtain the total volume of the active orders in the bid and the ask, for a certain position and for a certain bar. This property only returns results in real time as there is not historical data kept for the bid and ask positions.

Syntax:
.LimitVol(Level, Side, BarsAgo, Identifier)

Parameters:

Name	**Default**	**Description**

Level	1	Indicates the nth bid and ask position whose volume we are willing to know.
Side	osBuy	Position of the order, osBuy (buy), osSell (sell).
BarsAgo	0	Only returns results in real time
Identifier	Data	Data from which we are willing to extract the information..

Example:

.LimitVol(2, osBuy, 0, Data)

We obtain this way the volume of contract/titles offered in the second buying position of the data source DATA.

Low

Description:
This function returns the lowest value of a bar.

Syntax:
.Low(BarsAgo, Identifier)

Parameters:

Name	Default	Description
BarsAgo	0	Number of bars backwards, the value Default refers to the current bar. This parameter can be used with any numerical value contained in a variable and also by entering it value. It can also be specified as cuntion replacing the numerical.
Identifier	Data	Data source on which the function is applied. If there is more than one chart inserted in the same window they will be codified as Data1, Data2, Data3 etc.

Example:

.Low(0, Data2)   Returns the low of the current bar (from the data source Data2).

MinutesToTime

Description:
This function is used to transforme minutes in standard time.

Syntax:
.MinutesToTime(Minutes)

Parameters:

Name	Default	Description
Minutes	-	Time in numerical format

Example:

.MinutesToTime(300)   Will return 5 (300 minutes are worthing 5 hours).

NetProfit

Description:
This function is used to obtain the value of the system´s net profit until the current bar (bar on which the calculation is being made). To obtain the total profit we will consider that the lasto penes trade concludes in the close of the current bar.

Syntax:
.NetProfit(Show As SttRepresentation)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to determine the format on which the information is going to show up, **ByPoints**, or**Percentage.**

Example:

.NetProfit(point)	Returns in points the net profit obtained by the system..

NumberOfLines

Description:
This functions returns the number of lines of the indicator on which it is applied. The indicator must have had previously assigned values to these lines so that the values returned by the function includes them (as the function only returns the values of the current lines).

Syntax:
.NumberOfLines

Parameters:

Name	Default	Description
-	-	-

Example:

Supposing tht we have created an indicator callled MyIndicator that runs the following calculation:

```
Value1 = .High + .Low + .Close / 3
Value2 = .High + .Low / 2
Value3 = .Open + .Close + .Close(1) / 3
```

And then we paint:

```
.SetIndicatorValue Valor1, 1
.SetIndicatorValue Valor2, 2
.SetIndicatorValue Valor3, 3
```

If, inside our code, we use the function:

```
NLines=.NumberOfLines
```

Nlines will be equal to 3 that is the number of painted lines.

NumberOfLosingTrades

Description:

This functions returns the number of losing trades made by our system until the current bar. This value will change while new bars are generated and it value will depend on the bar on which the property is called.

Syntax:
.NumberOfLosingTrades

Parameters:

Name	Default	Description

-	-	-

Example:

Imagine that we are willing to know, in a certain bar, the number of accumulated losing trades. We could define a variable:

Dim losers as long

losers = .NumberOfLosingTrades

In this case, it returns the number of losing trades accumulated at the momento since the momento when we called the property.

NumberOfTrades

Description:
Returns the numberof trades made at the moment. This value will change while new bars are generated by and it value will depend on the bar on which the property is called.

Syntax:
.NumberOfTrades

Parameters:

Name	Default	Description
-	-	-

Example:

Supposing that we are willing to define, in a certain bar, the number of accumulated trades, we could define a variable:

Dim totaltrades as long

totaltrades=.NumberOfTrades

Returns the number of trades accumulated when the property is called.

NumberOfWinningTrades

Description:
This function returns the number of winning trades. This value will change while new bars are generated by and it value will depend on the bar on which the property is called.
Syntax:
.NumberOfWinningTrades

Parameters:

Name	Default	Description
-	-	-

Example:

Supposing that we are willing to know, in a certain bar, the number of accumulated winning trades, we could define a variable:

Dim winers as long

Winers =.NumberOfWinningTrades

Returns the accumulated number of wining trades untill the momento when the property is called.

Open

Description:
This function returns the value of the bar´s open.

Syntax:
.Open(BarsAgo, Identifier)

Parameters:

Name	Default	Description
BarsAgo	0	Number of bars backwards. The value Defaultrefers to the current barl. In this parameter, we can enter a numerical value or indicate a numerical value contained in a variable. It can also be specified as a function replacing the numerical value. This parameter only allows positive values.
Identifier	Data	Primary data source. The system Works on Data which is the Name taken by the chart Data series on which the strategy is applied. If there are more charts inserted in the same window they will be codified as Data2, Data3, Data4, etc.

Example:

.Open(3, Data1)Returns the open value three bars backwards from the data source codified as Data1.

OpenDay

Description:
This function returns a value 1 if the bar to which we are referring is the one corresponding to the last trading session and 0 in the opposite case.

Syntax:
.OpenDay(Identifier, BarsAgo)

Parameters:

Name	Default	Description
Identifier	Data	Primary data source. The systems act in Datawhich that is the Nametaken by the data series on the chart on which the strategy is applied. If there is more than one data in the same window, they will be codified as Data2, Data3, Data4, etc. It can also be an indicator.
BarsAgo	-	Specifies the bar to obtain the data, the current bar is 0.

OpenInt

Description:
This function returns the value of the **OpenInterest in a bar** (in the case of the futures contract, where this information is provided).

Syntax:
.OpenInt(BarsAgo, Identifier)

Parameters:

Name	Default	Description
BarsAgo	0	Number of bars backwards. The valueDefaultrefers to the current bar. In this parameter we can indicated a numerical value contained in a variable or type a numerical value. It can also be specified as a function replacing the numerical value.

		This parameter only allows positive values..
Identifier	Data	Primary data source. The systems act in Data which that is the Name taken by the data series on the chart on which the strategy is applied. If there is more than one data in the same window, they will be codified as Data2, Data3, Data4, etc. It can also be an indicator.

Example:

.OpenInt(5, Data)

Returns the OpenInteress 5 bars backwards (from the data series Data).

■PaintBar

Description:

By using this option we can paint the required fields from certain bars as open, high.low,close.

Syntax:
.PaintBar(Open, High, Low, Close, Color, LineNumber, Width, nBars)

Parameters:

Name	Default	Description
Open	-	Enables to use a function or variable.
High	-	Enables to use a function or variable.
Low	-	Enables to use a function or variable.
Close	-	Enables to use a function or variable.
Color	-	Color used to paint the bars by using the functionRGB.
LineNumber	-	Specifies the order number for paint suty in studies with different paint orders (ex, if we want to include withing a study a paint bar order (.PaintBar) and a paint line orders (.PaintSeries), we must set in the first order, line number 0 and in the second order line number 1.
Width	1	Width of the bar to be painted.
nBarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.

Example:

.PaintBar .Open, .High, .Low, .Close, RGB(0,0,225), 0, 1, 0

Paints in the current bar in blue by keeping its original colour.

■ PaintCandlestick

Description:

By using this option we can paint the required fields from certain candlesticks as open, high.low,close.

Syntax:
.PaintCandlestick(Open, High, Low, Close, Color, LineNumber, Width, nBars)

Parameters:

Name	Default	Description
Open	-	Enables to use a function or variable.
High	-	Enables to use a function or variable.
Low	-	Enables to use a function or variable.
Close	-	Enables to use a function or variable.
Color	-	Color used to paint the bars by using the functionRGB.
LineNumber	-	Specifies the order number for paint suty in studies with different paint orders (ex, if we want to include withing a study a paint bar order (.PaintBar) and a paint line orders (.PaintSeries), we must set in the first order, line number 0 and in the second order line number 1.
Width	1	Width of the bar to be painted.
nBarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.

Example:

.PaintCandlestick .Open, .High, .Low, .Close, RGB(32,151,25), 0, 1, 0

Paints in the current br a candlestick in green tone keeping the original width.

PaintMaxMin

Description:

This function is similar to PaintBar/PaintCandlestick. The difference is that, in the current function, we can only establish values for the high and the low of the bar we are willing to paint.

Syntax:
.PaintMaxMin(Top, Bottom, Color, LineNumber, Width, nBars)

Parameters:

Name	Default	Description
Top	-	High of the bar we are willing to pain. Enables to use a function or variable.
Bottom	-	Low of the bar we are willing to pain. Enables to use a function or variable.
Color	-	Color us the paint the bar. We use the function RGB.
LineNumber	-	Specifies the order number for paint suty in studies with different paint orders (ex, if we want to include withing a study a paint bar order (.PaintBar) and a paint line orders (.PaintSeries), we must set in the first order, line number 0 and in the second order line number 1.
Width	1	Width use to pain.
nBarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.

Example:

.PaintMaxMin .Low, .Low, RGB(0, 0, 255), 0, 8, 0

Paints the low of the bar with a circle (width 8) in blue color.

PaintSeries

Description:

This function is used to paint data lines in a window. This task can also be run by creating an indicator, but if we wanted to mix lines with options of painting bars or painting figures we could decide to create a study including these two types. We must remember that the studies can not be used in other kind of strategies as systems, indicators etc. Consequently if we want to use the data lines later on we should use an.

Syntax:
.PaintSeries(Price, Color, LineNumber, Width, nBars)

Parameters:

Name	Default	Description
Price	-	In this parameter we incicate the required value for the line in the current bar. Any numerical value or a numerical type variable is accepted in this parameter.
Color	-	Color used to paint.We use the function RGB.
LineNumber	-	Specifies the order number for paint suty in studies with different paint orders (ex, if we want to include withing a study a paint bar order (.PaintBar) and a paint line orders (.PaintSeries), we must set in the first order, line number 0 and in the second order line number 1.
Width	1	Width used to paint the candlestick.
nBarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.

Example:

.PaintSeries (.High + .Low) / 2, RGB(255, 0, 0), 0, 1, 0

In this case the function paints a green colored line representing the average point of the bar.

PercentProfitable

Description:
This function returns the reliability ration. This value will change while new bar are generated by and it value will depend on the bar on which the property is called.

Syntax:
.PercentProfitable

Parameters:

Name	Default	Description
-	-	-

Example:

We want to know, at a certain stage, the reliability ratio of our system. To do so we shall previously define a variable:

Dim Reliability as double

At a certain stage, we will assign to this variable the value of the property .PercentProfitable:

Fiabilidad =.PercentProfitable

ProfitFactor

Description:
This function returns the **Profit factor**. This value will change while new bars are generated, and its value will dependo n the bar on which this property is called.

Syntax:
.ProfitFactor(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which we want the information to show up, **ByPoints** or**Percentage**.

Example:

We want to know, at a certain moment, the profit factor (percentage) of our system. To do so, we shall previously define a variable:

Dim FProfit as double

At a certain stage, we will assign to this variable the value of this function:

FProfit =.ProfitFactor(percentage)

PRR

Description:
This function returns the ration**Adjusted Profit Factor**. This value will change while new bars are generated by and will depend on the bar on which the property is called.

Syntax:
.PRR(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which we want the information to show up, **ByPoints** or **Percentage**.

Example:

If we want to know, at a certain stage, the Adjusted profit factor (in points) of our system, we must first define a variable:

Dim AFact.Profas double

An assign to it, at a certain stage, the value of this function:

FGanAjus =.PRR(ByPoints)

RegressionAngle

Description:
This function returns the angle formed by the horizontal line and the regression line formed by the closing prices located between **StartBar**and**EndBar**.

Syntax:
.RegressionAngle(StarBar, EndBar, Data)

Parameters:

Name	Default	Description
StarBar	-	Start bar of the regression line.
EndBar	-	End bar of the regression line
Data	-	Data series whose data are we willing to extract, it can be an asset or an indicator.

RegressionSlope

Description:
This function returns the value of the slope of the regression line formed by the closing prices situated between**StartBar**and**EndBar**.

Syntax:
.RegressionSlope(StarBar, EndBar, Data)

Parameters:

Name	Default	Description
StarBar	-	Start bar of the regression line.
EndBar	-	End bar of the regression line
Data	-	Data series whose data are we willing to extract, it can be an asset or an indicator.

ReleaseDataIdentifier- RDI

Description:
This function enables to freed a **DataIndentifier**previously called via the methodGetSymbolIndentifier.

Syntax:
.ReleaseDataIdentifier(Identifier)

The short mode**RDI**can also be used.

.RDI(Identfier)

Parameters:

Name	Default	Description
Identifier	Data	Data to be freed.

Example:

First, we must create a data identifier that we are going to use too extract the minimimum movement of the Dax future December contract.

Dim NewData as DataIdentifer

NewData = .GetSymbolIdentifier("010015FDAXZ0", 1, crDays, CDate("01/01/2010"), CDate("01/01/2036"))

Then we extract the mínimum movement that we are willing to use:

Dim pip as Double

pip = .GetSymbolInfo(SbiMinMov, NuevoData)

Finally, as the new data is consuming ressources on the memory, we freed it as we are no longer going touse it. To do so, we use the function ReleaseDataIdentifier

.ReleaseDataIdentifier (NuevoData)

Sell

Description:

This function is used to sell futures contracts or to sell stocks at credit.It is important to know that the function is used to open short positions, not only to closet he long positions. Therefore, if we want to cancel a long without opening a new short position we must use the functionExitLong.

Syntax:

.Sell(Type, Contracts, Price, Label)

Parameters:

Name	Default	Description
Type	AtClose	Type of order we are willing to launch (AtClose, AtMarket, AtLimit, AtStop).
Contracts	1	Number of contracts/stocks. The numerical specifications on contracts can be replaced by variables or by any other function previously defined.
Price	-	Sell price. This parameter must only be indicated for AtStop and AtLimit orders. The value can be expressed via a number, a variable or a function, or a mix of both.
Label	-	Order label in text format.

Example:

.Sell(AtStop, 1, .Close-10, "V1")

In this case the function sends an AtStop sell order(one contract), the stop price set at the close of the bar minus 10 points and the label “V1”.

SetBackGroundColor

Description:
This function is used to paint the background of the window, for a certain bar, in the indicated color.

Syntax:
.SetBackGroundColor (BarsAgo, Color)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Color	-	Color on which the bakground is to be painted for the indicated bar(BarsAgo). Use of the function **RGB.**

Example:

.SetBackGroundColor(1, (RGB 255,0,0))   Will paint the background of the dinwo (in the previous bar)in red.

SetBarColor

Description:
This function assigns to the indicated bar of a certain indicator line, a certain color.

Syntax:
.SetBarColor (BarsAgo, Line, Color)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Line		Identifies the data line to which the bar on which the porperty will be established belongs.
Color	-	Color with which the indicated bar must be painted.It uses the function **RGB**.

Example:

.SetBarColor(0,1, RGB(255,0,0)) Will paint in red the current bar of line 1.

SetBarProperties

Description:
This function assigns to the indicated bar, of a certain line of the indicator, the color, width and type of line and also the representation used in the rest of the parameters.

Syntax:
.SetBarProperties (BarsAgo, Line, Color, Width, Style, Representation)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Line	-	Identifies the data line to which the bar on which the property is established belongs.
Color	-	Color to paint the indicated bar. Function**RGB**.
Width	-	Indicates the width to be applied to the bar (1,2,..).
Style	-	Style used for the representation: **lsSolid**continuous line **lsDash** non-continuous line **lsDot** dotted line **lsDashdot**dotted line with point **lsDashdotdot**dotted line with 2 pointss
Representation	-	Type or representation bo be used: **irBars** bars **irCandlestic**candlesticks **irDottedLine**dotted line **irFilledHistogram**filled histogramm **irHistogram** histogramm **irLineal** lineal **irParabolic** parabOLIC **irVolume**volumen

Example:

.SetBarProperties(0,1, RGB(255,0,0),2,lsSolid,irLineal)

In this case the function will paint in red the line 1 of the current bar (in format of continuous line and thickness 2).

SetBarRepresentation

Description:
This function sets the type of representation for a certain bar.

Syntax:
.SetBarRepresentation(BarsAgo, Line, Representation)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Line		Identifies the data line to which the bar on which the property is established belongs.
Representation	-	Type of representation to be used (the different types of representation can be checked with the functionSetBarProperties).

Example:

.SetBarRepresentation(0,1,irLineal)

The current bar of the indicator´s line number one will be represented in linear format.

SetBarStyle

Description:
This function set the style of a certain bar.

Syntax:
.SetBarStyle(BarsAgo, Line, Style)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Line		Identifies the data line to which the bar on which the property is established belongs.
Representation	-	Type of representation to be used (the different types of representation can be checked with the function SetBarProperties).

Example:

.SetBarStyle(0,1,irSolid)

The current bar of the indicator´s line number one will be represented in continous line format.

SetBarWidth

Description:
Assigns to the indicated bar, of a certain line, the required width.

Syntax:
.SetBarWidth(BarsAgo, Line, Width)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Line		Identifies the data line to which the bar on which the property is established belongs.

Width	-	Thickness of the bar to be represented in.

Example:

.SetBarWidth(0,1, 2) The current bar of the indicator line number 1 will be represented with thickness 2.

SetHistogramBand

Description:
This function assigns to the indicated bar of a certain data series, the value of the histogramm band, in fact the value serving as limit to paing the histogramm (if the representation used is the histogramm).This function is stricly associated to the use of the property StartBarRepresentation.

Syntax:
.SetHistogramBand(BarsAgo, Line, BandLine)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
Line		Identifies the data line to which the bar on which the property is established belongs.
BandLine	-	Data line used as reference to draw the histogramm.

Example:

Supposing that we use the function.SetBarRepresentation(0,1,irHistogram)the indicator will need a reference value to oscílate around in order to generate the histogram.

In this case:

.SetIndicatorValue (.Close – Close(1),1,0)With line 1 we represent the difference between closes

.SetIndicatorValue(0,2,0)   With line 2 we represent the value 0.

Next, by using the function SetBarRepresentation, we will indicate that line one will be painted with an histogram as default representation:

.SetBarRepresentation (0,1,irHistogram)

And finally we will indicate the line to oscillate around:

.SetHistogramBand (1,2)

This way, line 1 will use as line band (reference line) Line 2.

SetIndicatorPos

Description:
Indicates to a certain bar, of a certain line, a certain positions.

Syntax:
.SetIndicatorPos(BarsAgo, Line, IndicatorPosition)

Parameters:

Name	Default	Description
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.

Line	-	Identifies the data line to which the bar on which the property is established belongs.
IndicatorPosition	-	The trend can be **ipBull**(bullish), **ipBear**(bearish) or**ipNeutral**(flat)

Example:

.SetIndicatorPos(3,2,ipNeutral)Asigns to the bar number three backwards of line number 2 a neutral position.

SetIndicatorValue

Description:
This function assigns a value in a certain bar.A certain trend will be ascribed to this value.

Syntax:
.SetIndicatorPos(Value, Line, BarsAgo, IndicatorPosition)

Parameters:

Name	Default	Description
Value	-	Double type variable
Line	1	Identifies the data line to which the bar on which the property is established belongs.
BarsAgo	-	Number of bars backwards.The value 0 refers to the current bar.
IndicatorPosition	-	The trend can be **ipBull** (bullish), **ipBear** (bearish) or **ipNeutral** (flat)

Example:

Imagine that we define a Double type variable:

```
Dim buffer3 as Double
```

To this variable we assign the value returned by the following calculation:

```
buffer3 = .High(0) - .Low(0) / .High(0) * 100
```

Next, we can use the function .SetIndicatorValue to paint on each bar the value calculated for this variable:

```
If buffer3 > 50 then
        .SetIndicatorValue buffer3, 1,0,ipBull
    Else
        .SetIndicatorValue buffer3, 1,0,ipBear
End If
```

The difference is that, depending on this value, the indicator will be ascribed a bull or bear trend.

SetLineName

Description:
Asigns aNameto the indicated line.

Syntax:
.SetLineName(Line, LineName)

Parameters:

Name	Default	Description
Line	-	Identifies the data line to which the bar on which the property is established belongs.
LineName	-	The Name is ascribed to the corresponding property.

Example:

.SetLineName(2, DT)   Assigns the Name “DT” to line 2.

SetWndBackGroundColor

Description:
Assignsthe background color to the window of an indicator.

Syntax:
.SetWndBackGrounColor()

Parameters:

Name	Default	Description
Color	-	Use of the function **RGB**to identify the background color of a window.

Example:

.SetWndBackgroundColor(RGB(255,0,0)) Assigns the red color to the indicator window.

ShouldTerminated

Description:
This function is used to stop the calculation process f a strategy. **ShouldTerminated**is a boolean variable, initialized with the value false, so that, in order to interrupt the calculations we must assign to it the value True.
This function turns out to be very useful when working with extended historical data and we want to stop the calculation under certain circumstances.

Syntax:
.ShouldTerminated = True

Parameters:

Name	Default	Description
-	-	-

Example:

```
If .CurrentBar = 1000 And .NetProfit < 100 Then
.ShouldTerminate = True
End If
```

In this case, the calculations will stop when 1000 bars of the calculation have gone by and the net profit is lower than 100 euros.

Slope

Description:
This function returns the value of the regression line slope, this regression line is formed by the indicated prices between **StartBar**and**EndBar**.

Syntax:
.Slope(StarBar. EndBar, StarPrice, EndPrice, Data)

Parameters:

Name	Default	Description
StarBar	-	End bar of the regression line.
EndBar	-	Start bar of the regression line.
StarPrice	-	Start price of the regression line.
EndPrice	-	End price of the regression line.
Data	-	Data series from which we extract the iformation. It can be an asset but also an indicator.

StandardDeviation

Description:
Returns the standard deviation.

Syntax:
.StandardDeviation(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the forma on which we want the information to be shown, **ByPoints**, or**Percentage**.

Example:

Dim DEas double

DE=.StandardDeviation(ByPoints)

When we use this property, the value of the standard eviation in points will be ascribed to the variable DE,.

StarBar

Description:
Enables to specify the start bar for the system calculation.

Syntax:
.StarBar=(Number of bars)

Parameters:

Name	Default	Description
-	-	-

Example:

.StarBar=25   The system will start its calculation after bar number 25 of the historical data.

Time

Description
Returns the value of the field Timeof a certain bar. The time of a bar is given by the en dime of the temporary period resuming the bar. The time of a bar is considered in military format (HHMM), so if the time of a bar is 5:35pm, in visual chart it will be considered as the numerical format 17:35h.

Syntax:
.Time(BarsAgo, Identifier)

Parameters:

Name	Default	Description
BarsAgo	0	Number of bars backward.By default it refers to the current bar.
Identifier	Data	Data source on which the function is applied. If there are more than 1 chart inserted, they will be codified as Data1, Data2, etc.

Example:

.Time(3, Data1)   Returns the time in military format (HHMM) from Data1 three bars backwards.

TimeEx

Description
Returs the date of the reference bar in Date format (DD/MM/AAAA HH:MM:SS).

Syntax:
.TimeEx(BarsAgo, Identifier, TickIndex)

Parameters:

Name	Default	Description
BarsAgo	0	Bar from which we are willing to extract the date, by default it refers to the current bar
Identifier	Data	Data source on which the function is applied. If there are more than 1 chart inserted, they will be codified as Data1, Data2, etc.

TickIndex is a Start parameter. When we are trading with a tick chart, some of them may have the same date.This parameter is filled by indicating the nth tick position referring to the same date.

Example:

.TimeExe(1, Data2)

Returns the date (DD/MM/AAAA HH:MM:SS) of the previous bar for the series Data2.

TimeToMinutes

Description:
Returns the number of minutes passed since 00:00.

Syntax:
.TimeToMinutes(Time)

Parameters:

Name	Default	Description
Time	-	Time in military format (HHMM)

Example:

.TimeToMinutes(1735)

Returns 1055 minutes.

This

Description:
Property with which we refer to the data we are woking in. It is used to assigne it to any alement requesting a **DataIdentifier** as entry parameter (for Example, the function .GetIndicatorIdentifier(Code_Ind, .This, Period, PriceClose, etc.)

Syntax:
.This

Parameters:

Name	Default	Description
-	-	-

Volume

Description:
This function returns the value of the volume (stocks/negotiated contracts) in a certain bar.

Syntax:
.Volume (BarsAgo, Identifier)

Parameters:

Name	Default	Description
BarsAgo	0	Bar number. The valueDefaultrefers to the current bar.Upon this parameter, we can indicate any numerical value contained under a variable or even type the number. It can also be specified as a function replacing the numerical value.
Identifier	Data	Data series from which we obtain the volumen in the specified bar. If there is more than 1 chart in the same window, they will be codified as Data1, Data2, etc.

Example:

.Volume(15, Data2)

Returns the volumen negotiated 15 barsbackwards, in the data series Data2.

WorstSeries

Description:
This function returns the worse series of losses according to their results.

Syntax:
.Worstseries(Show)

Parameters:

Name	Default	Description
Show	Bypoints	Enables to indicate the format on which the information will show up, **ByPoints**, or**Percentage**.

VisualChart
Real Time Financial Information & Trading Software