ProRealTime SOFTWARE

Programming Guide

Trading Systems (ProBacktest & ProOrder)

ProRealTime's trading system tools let you create investment strategies that can be backtested or used automated trading mode.

Follow ProRealTime Programming on Google+ for updates about ProRealTime programming languages.

V 4.2.4 - 20141219

Table of contents

Warning: ProRealTime does not provide investment advisory services. This document is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

ProRealTime SOFTWARE

Introduction to programming trading systems with ProRealTime v10

This version of the programming guide applies to ProRealTime v10 and above.

The trading system tools in ProRealTime let you create personalized investment strategies via programming or assisted creation (no programming required). These trading systems may be executed:

- As backtests to test their performance over historical data of a security
- As automatic trading systems: Orders are placed in real-time from a trading or PaperTrading account.

Trading system programming uses the ProBuilder programming language that is also used to write indicators in ProRealTime with additional instructions that apply only to programming trading systems.

Trading system programs can include instructions to take positions, set stops and risk management of each trading system based on personalized conditions such as:

- Predefined indicators in the workstation or indicators that you have programmed
- Past performance of your trading system
- Your trading system's latest orders

The results of a trading system are presented in the following format:

- The equity curve shows the status of a system's gains and losses on a particular instrument
- The positions histogram shows the positions of the system (green bars for buying positions and red bars for short selling positions). No bar is shown if there is no position for a particular time period.
- The detailed report in the application indicates the statistics of your system for the security over the time period it was backtested or executed.

In backtesting mode, it is also possible to optimize variables of your trading system to see which values give the best results over the period of history you are examining.

In automatic trading mode, the orders placed by your trading systems appear in your portfolio and order list. The portfolio is updated with the gains and losses made.

This manual is organized in the following manner: The first part explains how to access the trading system creation features. The second part explains the ProBuilder instructions used to program systems. The third part explains how to backtest trading systems with ProBacktest. The fourth part explains how to execute a trading system automatically. The exhibits at the end show how trading system results are displayed and also provides some example programs as well as the glossary for the ProBuilder language.

For beginning users, we advise you to first watch the video "**Backtest your strategies without writing a single line of code**".

The ideas expressed in this manual are only to help you learn to write trading systems and test your own ideas. They are not investment advice in any case.

We wish you the best of success in your trading and hope you will enjoy the manual.

Introduction

Accessing trading system programming

The zone for trading system creation may be accessed with the "Indicators & Trading systems window" in the upper-right corner of every chart in your workstation

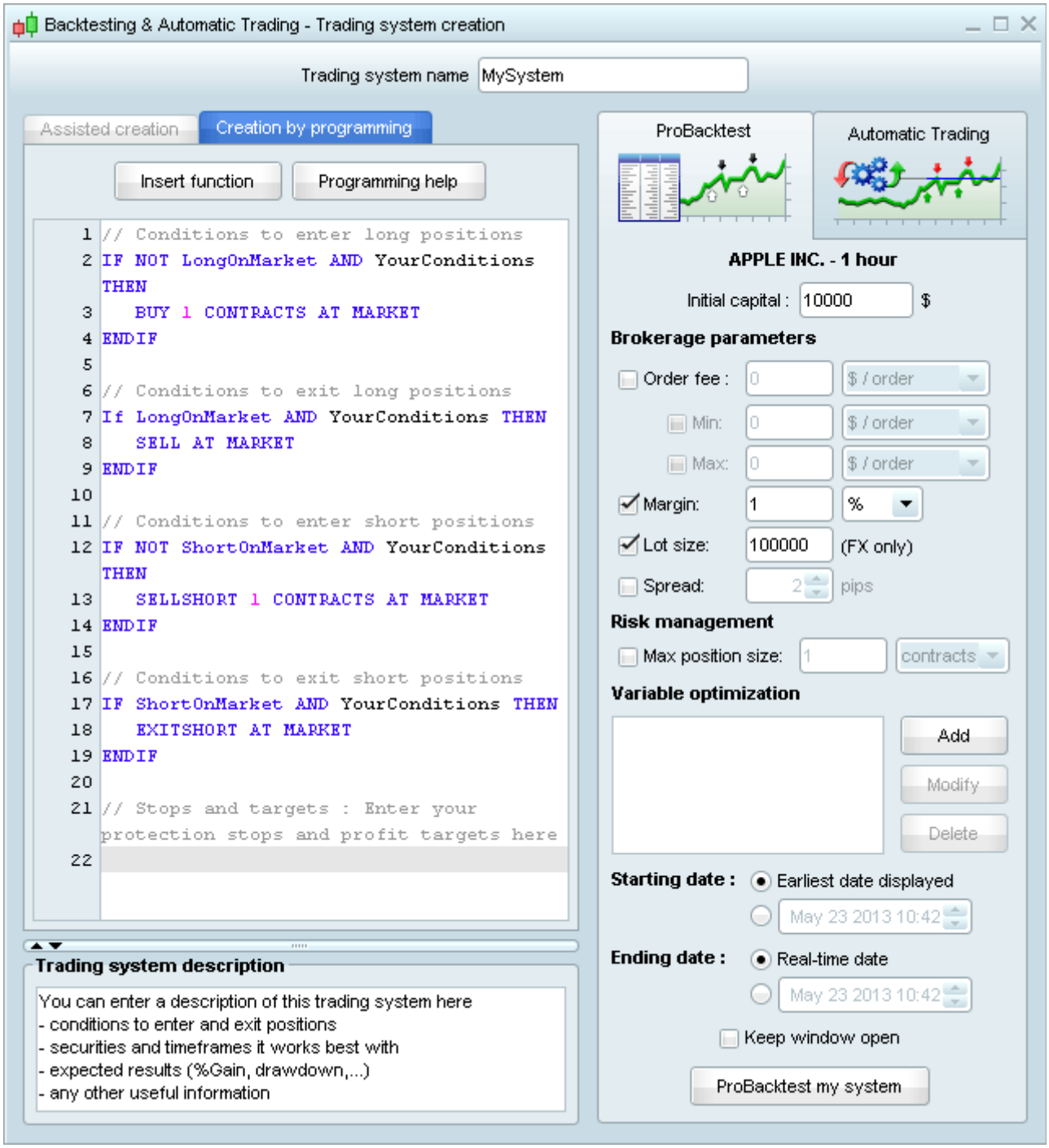
By default, the "Indicators" tab will be selected. Select the second tab "Backtesting & Automatic trading". You will then be able to:

- Access the list of existing trading systems (personal or predefined)
- Create a new trading system or apply an existing system to any security
- Modify or delete an existing trading system
- Import or export trading systems

Trading system creation window

The trading system creation window is composed of two main zones:

- The creation zone (assisted creation or creation by programming) appears on the left.
- The strategy application zone appears on the right. This includes a ProBacktest tab to backtest a trading system with historical data and a ProOrder AutoTrading tab to automatically execute a trading system. The options in the ProBacktest tab are detailed in section 3 of this manual.



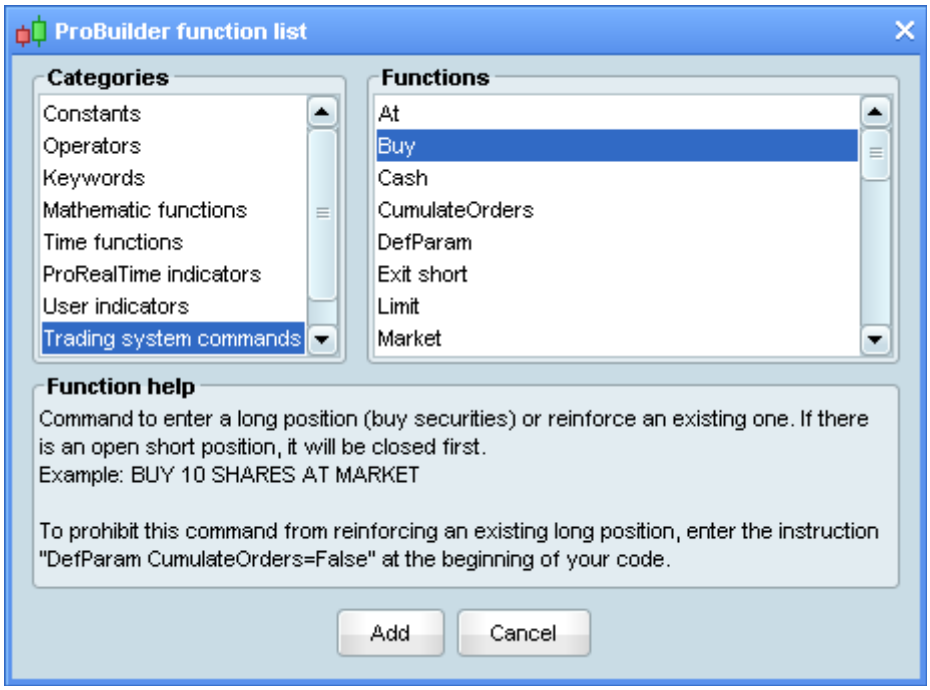
The creation zone allows you to:

- Program the trading system using the text editor
- Use the “insert function” button which allows you to open a new window with a list of ProBuilder and trading system commands separated into different categories that give you contextual help while programming. You can see a help text related to the command or function selected in the lower part of the window.

Example:

Let's use the function library by clicking on “insert function”.

Choose the section “Trading system commands”, click “**BUY**” then click “Add”. The command will be added to your program.



Now lets try to create full line of code. Suppose we want to buy 10 shares at market.

Proceed as above to find the functions "**SHARES**", "**AT**" and "**MARKET**" (separating each word with a space). Specify between "**BUY**" and "**SHARES**" the number to buy (10).

You will then obtain the instruction "**BUY 10 SHARES AT MARKET**" which is an instruction to buy 10 shares, lots or contracts at market price. The next instruction presents all the instructions which are available to program trading systems.

To see some examples of complete trading systems, check Annex B at the end of this manual.

Keyboard shortcuts

The trading system creation window has a number of useful features that can be accessed with keyboard shortcuts starting with ProRealTime version 10:

- The trading system creation window has a number of useful features that can be accessed with keyboard shortcuts starting with ProRealTime version 10:
- Select all (Ctrl + A): Select all text in the programming window
- Copy (Ctrl + C): Copy selected text
- Paste (Ctrl +X): Paste copied text
- Undo (Ctrl + Z): Undo the last action in the programming window
- Redo (Ctrl + Y): Redo the last action in the programming window
- Find / Replace (Ctrl + F): Find a text in the programming window / replace a text in the programming window ((this feature is case-sensitive)
- Comment / Uncomment (Ctrl + R): Comment the selected code / Uncomment the selected code (the commented code will be preceded by "//" or "REM" and colored gray. It will not be taken into account when the code is executed).

For Mac users, the same keyboard shortcuts can be accessed with the "Apple" key in place of the "Ctrl" key. Most of these features can also be accessed by right-clicking in the trading system creation window's programming zone.

Programming trading systems

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Programming in ProBuilder language

We advise you to read the **manual concerning how to program indicators in ProBuilder**. This section details the ProBuilder commands that are specific to trading systems.

ProBuilder is the programming language used In the workstation. It is very simple to use and offers many possibilities. Some important principles of ProBuilder are:

- The calculations are done at the end of each bar. ProBuilder programs including trading systems and all of their functions are evaluated at the end of each bar from the beginning to the end.
- All instructions to place orders are triggered after calculations on the current candlestick are finished (meaning the orders will be executed at the open of the next candlestick at the earliest).

Entering and exiting the market

Different instructions are used depending on the type of position:

Long positions:

- BUY is an enter long position instruction (Buy securities)
- SELL is an exit long position instruction (Sell securities)

BUY allows you to open a long position on the market or add to an existing open position. It is associated with the instruction SELL which allows to close or partially close a position. The instruction SELL has no effect if there is no long position open.

Short positions:

- SELLSHORT is a enter short instruction (shortsell securities)
- EXITSHORT is an instruction to exit a short position (buy back shorted securities)

These 2 instructions work similarly to “BUY” and “SELL”. The instruction EXITSHORT has no effect if there is no short position open.

It is not possible to take a long and short position at the same time on the same security. In practice, this means it is also possible to close a long position with a SELLSHORT command or close a short position with BUY command.

Note: Verify the maximum position size defined in the “risk management” section for ProBacktest or in ProOrder for automatic trading systems. If you try to place an order that increases the size of your position to a larger amount than this maximum position size authorized, the order will be rejected and your original position will be maintained.

Each of these commands may be followed by quantity and at type instructions as shown below:

<Ordre> <Quantité> AT <type>

Example:

```
BUY 1000 CASH AT MARKET or SELL 1 SHARE AT 1,56 LIMIT
```

Quantity

There are two ways to define the quantity:

- SHARES corresponds to one unit of the instrument. “1 Share” can represent 1 stock, 1 future contract, or one Forex contract. SHARES can be used interchangeably with SHARE, CONTRACT, CONTRACTS, LOT, or LOTS for any type of instrument. In he case of Forex, the quantity bought will be multiplied by the size of one lot. If no quantity is specified, the following default values are used:
  - 1 unit for a position entry (Ex: BUY AT MARKET, buys a quantity of "1" at market price)
  - The entire quantity of a position for an exit (Ex: SELL AT MARKET sells the entire long position)
- CASH corresponds to a cash amount (like € or $) and this instruction can only be used for buying or selling shares. The quantity of the order will be calculated at the close of the bar and rounded down by default. Brokerage fees are not taken into account when calculating the quantity to buy or sell in cash.

Example:
```
BUY 1000 CASH AT MARKET
```

The instruction ROUNDEDUP can be used to round this quantity to up instead of down.

Example:
```
BUY 1000 CASH ROUNDEDUP AT MARKET
```

Mode

Three modes are available for these type of orders:

AT MARKET: The order will be placed at market price at the open of the next bar.

Example:
```
BUY 1 SHARE AT MARKET
```

AT <price> LIMIT: A limit order will be placed at the indicated price

AT <price> STOP: A stop order will be placed at the indicated price

Example:
```
BUY 1 SHARE AT 10.5 LIMIT
```

Limit and stop orders with specific levels are valid for one bar by default starting at the open of the next bar. They are canceled if not executed.

These orders are different from protection stop and target orders (see next section) which are linked to an open position and valid until the close of that position.

Some orders may be treated as market orders in the following conditions:

- If a BUY a quantity at price LIMIT order is placed above the market price, the order is treated as a market order.
- If a BUY a quantity at price STOP order is placed below the market price, the order is treated as a market order.
- If an order to SELLSHORT a quantity at price LIMIT order is placed below the market price, the order is treated as a market order.
- If an order to SELLSHORT a quantity at price STOP order is placed above the market price the order is treated as a market order.

Example:

The following program buys 1 share at market price if the RSI is oversold (RSI < 30) and price is under the lower Bollinger band. It sells if the RSI is overbought (RSI > 70) and price is above the upper Bollinger band.

```
MyRSI = RSI[14](Close)
MyBollingerDown = BollingerDown[25](Close)
MyBollingerUp = BollingerUp[25](Close)

IF MyRSI < 30 AND Close < MyBollingerDown THEN
   BUY 1 SHARE AT MARKET
ENDIF
IF MyRSI > 70 AND Close > MyBollingerUp THEN
   SELL AT MARKET
ENDIF
```

You can set the validity length of limit and stop orders.

The following example shows how it is possible to create a limit order with a validity set to a specific number of bars by using variables. The code places a buy limit order at the close price of the bar on which a moving-average crossover occurred. This limit is valid for 10 bars after the bar of the crossing. If it is not executed during these 10 bars, it is cancelled.

Example:

```
// Definition of the validity length of the order
ONCE NbBarLimit = 10

MM20 = Average[20](close)
MM50 = Average[50](close)
// If MM20 crosses over MM50, we define 2 variables "MyLimitBuy" and "MyIndex" containing the close price at that time and the index of the bar of the cross.
IF MM20 CROSSES OVER MM50 THEN
   MyLimitBuy = close
   MyIndex = Barindex
ENDIF

IF BarIndex >= MyIndex + NbBarLimit THEN
   MyLimitBuy = 0
ENDIF
// Place an order at the price MyLimitBuy valid as long as this variable is greater than 0 and we are not in a long position.
// Remember: MyLimitBuy is greater than 0 for the 10 bars after the bar of the crossing.
IF MyLimitBuy > 0 AND NOT LongOnMarket THEN
   BUY 1 SHARES AT MyLimitBuy LIMIT
ENDIF
```

In case the order was not executed, it is possible to replace the expired buy limit order with a buy at market price order. This could be done by adding the following code to the previous one:

```
IF MyIndex + NbBarLimit AND MyLimitBuy > 0 AND NOT LongOnMarket THEN
   BUY 1 SHARES AT MARKET
ENDIF
```

Stops and targets

ProBuilder also lets you define profit targets and protection stops. The syntax is as follows:

SET STOP <type> <value> **or** SET TARGET <type> <value>

Example:

```
SET STOP %LOSS 2 // Set a stop loss of 2%
```

Each instruction is detailed in the next paragraphs.

Note the difference between **STOP** commands:

- **AT <price> STOP** is used to ENTER a position. This order is valid 1 bar by default.
- **SET STOP LOSS <price>**, is a protection stop used to EXIT a position. This order is valid until the position is closed.

Protection stops

Protection stops let us limit the losses of a position. They can be defined in relative or absolute terms:

SET STOP LOSS x: Set a stop loss to close the position x units from entry price.

SET STOP pLOSS x: Set a stop loss to close the position x points from entry price.

SET STOP %LOSS x: Set a stop loss to close the position when the loss reaches x%, brokerage fees not included.

SET STOP $LOSS x: Set a stop loss to close the position of X €,$ (currency of the instrument), brokerage fees not included.

The quantity and direction (exit long or exit short position) of the protection stop order are automatically adapted to the type of position currently open. Protection stops are linked to a position. If there is no position open, the stop loss will not be active.

To deactivate a stop loss in the code, the following instruction can be used:

```
SET STOP LOSS 0, SET STOP pLOSS 0, SET STOP %LOSS 0, SET STOP $LOSS 0
```

Set Target Profit

These types of instructions let you exit a position when gains attain a certain amount.

SET TARGET PROFIT x: Set a profit target to close the position x units from the average position price.

SET TARGET pPROFIT x: Set a profit target to close the position x points from the average position price.

SET TARGET %PROFIT x: Set a profit target to close the position when profit reaches x% (brokerage fees not included).

SET TARGET $PROFIT x: Set a profit target order to close the position when the gain reaches x €,$ (currency of the instrument, brokerage fees not included).

The quantity and direction (exit long or exit short position) of the profit target order are automatically adapted to the type of position currently open. All profit targets are linked to a position. If there is no open position, the profit target order is not active.

To deactivate a profit target in the code, the following instruction can be used:

```
SET TARGET PROFIT 0, SET TARGET pPROFIT 0, SET TARGET %PROFIT 0, SET TARGET $PROFIT 0
```

Trailing stops

A trailing stop is an stop order whose price changes depending on the evolution of price. For long positions, when price increases, the level of a trailing stop increases, but if the price decreases, the level of the trailing stop remains constant. Trailing stops on short positions work in the opposite manner: when price decreases, the level of the trailing stop decreases, but if price increases, the level of the trailing stop remains constant.

Like protection stops, trailing stops can be defined in relative or absolute terms:

SET STOP TRAILING y: Sets a trailing stop y units from average position price.

SET STOP pTRAILING y: Sets a trailing stop y points from average position price.

SET STOP %TRAILING y: Sets a trailing stop y% from average position price, brokerage fees not included.

SET STOP $TRAILING y: Sets a trailing stop y €,$ (currency of the instrument) from average position price, brokerage fees not included.

The quantity and direction (exit long or exit short position) of the trailing stop order are automatically adapted to the type of position currently open. All trailing stops are linked to a position. If there is no open position, the trailing stop is not active.

If the quantity of the position changes, the level of the stop is re-initialized.

To deactivate a trailing stop in the code, the following instruction can be used:

```
SET STOP TRAILING 0, SET STOP pTRAILING 0, SET STOP %TRAILING 0, SET STOP $TRAILING 0
```

Example:

A long position is taken on the DAX at 6000 points and a trailing stop is placed at 50 points:

```
SET STOP pTRAILING 50
```

The stop is initially placed at 5950. Price increases to 6010 then decreases to 5980, the stop will increase 10 points to 5960, then stay there until price increases higher than 6010. It will be triggered if the price reaches 5960.

Use of "Set Target" and "Set Stop" with conditional "IF" statements

It is possible to change the type of target or stop set in your code depending on personalized conditions by using conditional if statements.

Example:

```
REM Use a stop loss of 10% if the gain of the previous trade was at least 10%, otherwise use a stop loss of 5%.
IF PositionPerf(1) > 0.1 THEN
   SET STOP %LOSS 10
ELSE
   SET STOP %LOSS 5
ENDIF
```

Multiple stop and target levels

Only one “Set Stop” and one “Set Target” command can be active at a time under normal circumstances. If there are successive “Set Stop” or “Set Target” commands in a code, the last command replaces the previous command.

Example:

```
SET STOP %LOSS 10 // Set a stop loss of 10%
SET TARGET PROFIT 50 // Set a profit target of 50 units
SET TARGET %PROFIT 5 // Removes the previous target of 50 units and replaces it with a profit target of 5%
SET STOP %TRAILING 2 // Removes the previous 10% stop loss and replaces it with a trailing stop of 2%
```

However, it is possible to combine fixed stops and trailing stops or stop losses and trailing stops with a single instruction shown below:

SET STOP <Mode> <value> (fixed) <TrailingType> <value> (trailing)

Mode: Loss, pLOSS, %LOSS, $LOSS

Trailing Type: TRAILING, pTRAILING, %TRAILING, $TRAILING

This instruction appears in the following form:

```
SET STOP [LOSS/pLOSS/$LOSS/%LOSS] <value> [TRAILING/pTRAILING/$TRAILING/%TRAILING] <value>
```

Examples of use:

`SET STOP LOSS x TRAILING y`: A stop loss is placed at x **units** from entry price and it becomes a trailing stop of y units if the trailing stop level becomes closer to current price than the stop loss level (when price varies favorably by **y units – x units**).

`SET STOP LOSS x pTRAILING y`: A stop loss is placed at x **units** from average position price and it becomes a trailing stop of y points if the trailing stop level becomes closer to current price than the stop loss level (when price varies favorably by **y points – x units**).

`SET STOP LOSS x $TRAILING y`: A stop loss is placed at x **units** from average position price and it becomes a trailing stop of y **$ or €** (currency of the instrument) if the trailing stop level becomes closer to the current price than the stop loss level.

`SET STOP LOSS x %TRAILING y`: A stop loss is placed at x **units** from average position price and it becomes a trailing stop of y% if the trailing stop level becomes closer to the current price than the stop loss level.

`SET STOP pLOSS x TRAILING y`: A stop loss is placed at x **points** from average position price and it becomes a trailing stop of y **units** if the trailing stop level becomes closer to current price than the stop loss level (this occurs when price varies favorably by **y units – x points**).

`SET STOP pLOSS x pTRAILING y`: A stop loss is placed at x **points** from average position price and it becomes a trailing stop of y **points** if the trailing stop level becomes closer to current price than the stop loss level (this occurs price varies favorably by **y points – x points**).

`SET STOP pLOSS x $TRAILING y`: A stop loss is placed at x **points** from average position price and it becomes a trailing stop of y **$ or €** (currency of the instrument) if the trailing stop level becomes closer to the current price than the stop loss level.

`SET STOP pLOSS x %TRAILING y`: A stop loss is placed at x **points** from average position price and it becomes a trailing stop of y% if the trailing stop level becomes closer to the current price than the stop loss level.

`SET STOP $LOSS x TRAILING y`: A stop loss of x **$ or €** (currency of the instrument) is placed and it becomes a trailing stop of y **units** if the trailing stop level becomes closer to current price than the stop loss level.

`SET STOP $LOSS x pTRAILING y`: A stop loss of x **$ or €** (currency of the instrument) is placed and it becomes a trailing stop of y **points** if the trailing stop level becomes closer to current price than the stop loss level.

`SET STOP $LOSS x $TRAILING y`: A stop loss of x **$ or €** (currency of the instrument) is placed and it becomes a trailing stop of y **$ or €** (currency of the instrument) if the trailing stop level is closer to the current price than the stop loss level.

`SET STOP $LOSS x %TRAILING y`: A stop loss of x **$ or €** (currency of the instrument) is placed and it becomes a trailing stop of y**%** if the trailing stop level becomes closer to the current price than the stop loss level.

`SET STOP %LOSS x TRAILING y`: A stop loss of x% is placed and it becomes a trailing stop of y **units** if the trailing stop level becomes closer to current price than the stop loss level.

`SET STOP %LOSS x pTRAILING y`: A stop loss of x% is placed and it becomes a trailing stop of y **points** if the trailing stop level becomes closer to current price than the stop loss level.

`SET STOP %LOSS x $TRAILING y`: A stop loss of x% is placed and it becomes a trailing stop of y **$ or €** (currency of the instrument) if the trailing stop level is closer to the current price than the stop loss level.

`SET STOP %LOSS x %TRAILING y`: A stop loss of x% is placed and it becomes a trailing stop of y**%** if the trailing stop level becomes closer to the current price than the stop loss level.

Example:

SET STOP LOSS x TRAILING y:

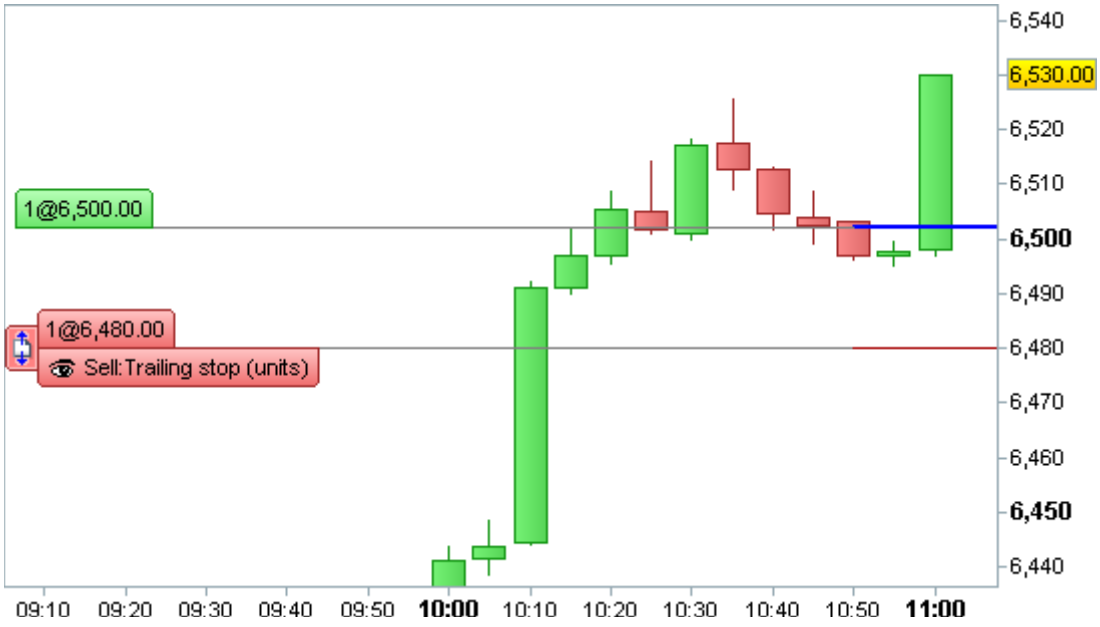
A stop is placed at x units from average position price and it becomes a trailing stop of y units if the trailing stop level becomes closer to current price than the stop loss level.

For example, if you enter a long position on the DAX future at 6500, the following code places a stop loss 20 units from the average position price which becomes a 50-unit trailing stop if price passes 6530.

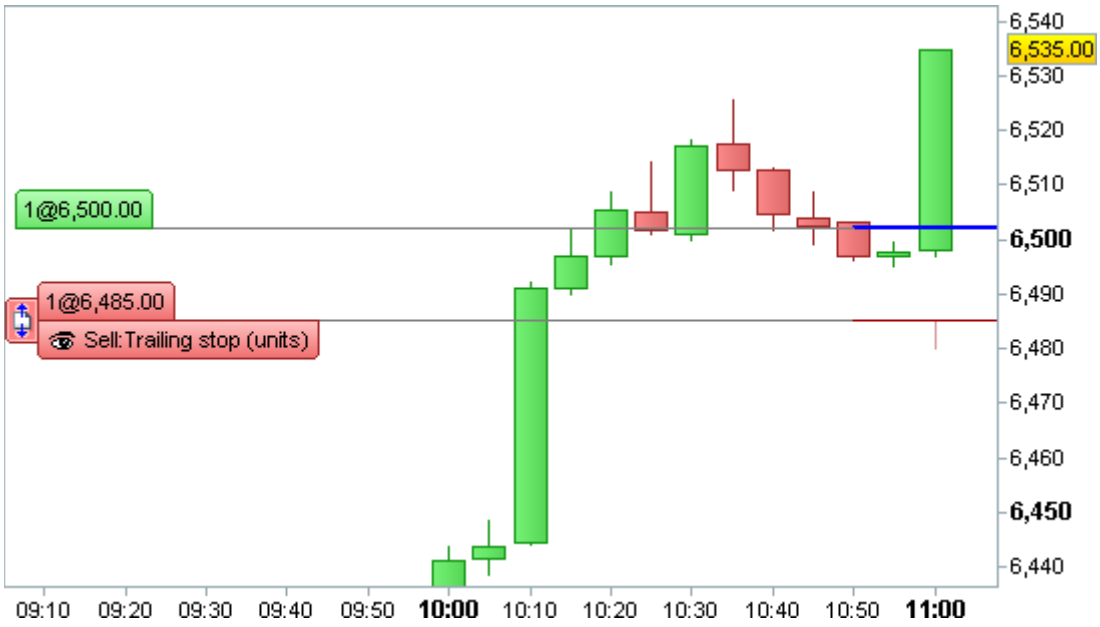
```
SET STOP LOSS 20 TRAILING 50
```

The following images illustrate the example:

The initial stop is placed at a fixed level 20 units below the position opening price (6480)



Only if price reaches 6530 (=6500 + (50-20)), the stop becomes 50-unit trailing stop.



If price rises to 6535 as shown in the image above, the trailing stop would rise to 6485.

Note concerning the use of points or units for the distance of stops and limits

Point size may vary depending on the type of instrument you are looking at, whereas unit size (variation of 1 unit on the chart) is always constant. Depending on your code and the instruments which it is applied to, you may prefer to use distances in points or units. Example.

On the EuroDollar (EUR/USD), 1 point =0.0001 units on the chart

On index Futures (DAX, FCE), 1 point = 1 unit on the chart

On futures on european interest rates,1 point = 0.01 units on the chart

Stopping a trading system with QUIT

The "Quit" instruction lets you stop a trading system. The stop occurs after the current bar. The pending orders are then canceled and any open positions are closed. This lets you stop a trading system in case of high losses or after a certain date for example.

Example:

```
If date > 20130101 THEN // Stop the strategy after January 1st, 2013
   QUIT
ENDIF
```

Position tracking

Position status variables

3 variables allow you to check the status of your trading systems positions:

ONMARKET: is equal to 1 if you have an open position, or zero otherwise

LONGONMARKET: is equal to one if you have a long position, or zero otherwise.

SHORTONMARKET: is equal to 1 if you have a short position, or zero otherwise.

They can be used with brackets. For example ONMARKET[1] is equal to 1 if you had an open position at the close of the previous bar, or 0 otherwise.

These variables are usually introduced with IF commands prior to entering a position:

Example:

```
REM Define the MACD
Indicator1 = MACD[12,26,9](Close)
REM Observe crossings of the MACD histogram
c1 = (Indicator1 CROSSES OVER 0)
REM BUY: if there is no long position and MACD > 0, buy 3 lots.
IF NOT LONGONMARKET AND c1 THEN
   BUY 3 SHARES AT MARKET
ENDIF
```

Size of position variables

These 3 variables allow you to know the quantity of an open position:

COUNTOFPOSITION: size of the position (in lots, shares, contracts...). It has a positive value if there is a long position open and a negative value if there is a short position open.

COUNTOFLONGSHARES: size of a long position (in lots, shares, contracts...) if there is a long position open. 0 otherwise.

COUNTOFSHORTSHARES: size of a short position (in lots, shares, contracts...). It has a positive value if there is a short position open and is 0 otherwise.

These variables are usually introduced with IF commands prior to entering a position.

Tip on using status variables: The code is evaluated at the end of each bar, and the orders are placed at the open of the next bar.

For example, in the following block of code, the variable “long” will not be equal to 1 at the close of the first candlestick, but only at the close of the second candlestick because the first buy orders I placed at the open of the second bar.

```
BUY 1 SHARE AT MARKET
IF NOT LONGONMARKET THEN
   long = 1
ENDIF
```

TradeIndex

The command **TRADEINDEX(n)** lets you access the bar index of the nth previous executed order:

TRADEINDEX(nth previous order)

Note: It is possible to use TradeIndex without a number between parenthesis. In this case, the program considers the bar of the last executed order: TradeIndex=TradeIndex(1).

TradeIndex is usually used conjointly with BarIndex.

Example:

```
REM: Close a long position if it has been open for at least 3 bars
IF LONGONMARKET AND (BarIndex - TradeIndex) >= 3 THEN
   SELL AT MARKET
ENDIF
```

TradePrice

The command **TRADEPRICE(n)** lets you find the price of the previously executed transaction.

The syntax is as follows:

TRADEPRICE(nth previous order)

If n is not specified, the price of the last executed order is referenced: TradePrice=TradePrice(1)

Example:

```
REM: Close a long position position if price is greater than the price of the previous order
plus 2%.
IF LONGONMARKET AND CLOSE > 1.02 * TRADEPRICE THEN
   SELL AT MARKET
ENDIF
```

PositionPerf

The instruction **POSITIONPERF(n)** returns:

- The performance (ratio gain/cost of the position) of the n-th last position closed if n>0 (not including brokerage fees)
- The performance (ratio gains/cost of the position) of the currently open position if n=0 (not including brokerage fees)

The syntax is as follows:

POSITIONPERF(nth previous position)

If n is not specified, we suppose that n=0. PositionPerf=PositionPerf(0).

Example:

```
REM BUY if the previous trade made at least 20% Gain.
IF NOTONMARKET AND PositionPerf(1) > 0.2 THEN
   BUY 1000 CASH AT MARKET
ENDIF
```

PositionPrice

The command PositionPrice lets you know the average purchase price of the currently open position.

POSITIONPRICE

It is calculated as the sum of all entry prices weighted by the quantity of each order. Only adding to a position will change the value of PositionPrice.

This instructions may be used with brackets to introduce an offset: POSITIONPRICE[1] returns the value of PositionPrice at the close of the previous bar.

Example:

If you buy one stock at a price of 5 € and buy the same stock again when the price is 10€ and buy the same stock again when the price is 15€, PositionPrice would be equal to: (5 + 10 + 15)/3 = 10 €.

If you then sell one share at a price of 20 €, PositionPrice would still be equal to 10 € (no change).

StrategyProfit

This command returns the gains or losses (in absolute and in the currency of the instrument, not including brokerage fees) realized since the beginning of the trading system. It is typically used with “QUIT” to stop a trading system that has lost too much.

STRATEGYPROFIT

This instruction can be used with brackets: StrategyProfit[1] gives the profit at the close of the previous bar.

Example:

```
IF STRATEGYPROFIT < -500 THEN
   QUIT
ENDIF
```

Note:

Remember that the trading systems are evaluated at the close of a bar. In the example above, losses may be greater than 500 € in case of a large loss during a single candlestick or in case of a gap.

As a result, a user who wanted to stop a trading system after 500 € of loss should first set a STOP LOSS to limit the losses, then use the block of code above to stop the system.

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Definition of parameters of execution of trading systems

Additional parameters may be defined with the instruction **DEFPARAM**.

Cumulate orders

The variable “CumulateOrders” lets you authorize or forbid cumulating orders to enter the market or add to a position. This parameter is set to “True” by default for codes created by programming which means that a trading system may add to an existing position at every bar where the conditions to enter that position are true. It is also possible to have multiple limit or stop orders to enter the market active at the same time in this case.

To prevent a strategy from increasing the size of an already open position, the following instruction must be set at the beginning of the code:

```
DEFPARAM CumulateOrders = False
```

DefParam instructions remain valid for the entire execution of the trading system. It is not possible to change the trading system's setting for cumulating orders or not during the execution of the trading system.

Examples:

```
// This code will buy 1 share every bar, up to a maximum of 3.
DEFPARAM CumulateOrders = True
If CountOfPosition < 3 THEN
   Buy 1 shares at market
Endif
// This code will buy 1 share at a price of 2 and an additional share at a price of 3
DEFPARAM CumulateOrders = True
If CountOfPosition < 2 THEN
   Buy 1 shares at 2 Limit
   Buy 1 shares at 3 Limit
Endif
// This code will buy 5 shares only once
DEFPARAM CumulateOrders = False
Buy 5 shares at market
```

It is possible to have several orders to exit the market in the same direction even while the parameter CumulateOrders is set to false.

Example:

```
// This code will buy 5 shares. Up to 3 shares will be sold if price crosses under the
40-period moving average. All shares will be sold in case of a 10% loss.
DEFPARAM CumulateOrders = False
Buy 5 shares at market
If close CROSSES UNDER Average[40] THEN
   SELL 3 SHARES AT MARKET
   Set stop %Trailing 10
Endif
```

Note on stops and target levels while CumulateOrders is active: If you use the instructions to set a stop loss, trailing stop or profit target with cumulate orders activated, the level is calculated based on your positions average entry price and is recalculated each time the position's quantity is modified.

Example:

If you buy 1 share at $10.00 and set a 10% stop loss and a 150% profit target, the initial levels would be: Stop at $9.00 and target at $25.00. If you buy a second share at a price of $20, the average entry price would be $15 and as a result the new levels would be: Stop at $13.50 and target at $37.50 (for the entire position).

Note on codes created with assisted creation mode: CumulateOrders is set to false by default for trading systems created in assisted creation mode (the instruction “DefParam CumulateOrders = False” will be present at the beginning of these codes.

PreLoadBars

The instruction "DefParam PreLoadBars" lets you configure the maximum amount of bars that are preloaded prior to the start of a trading system for the calculation of indicators used in the system prior to the system's start (personal or predefined indicators). By default this parameter is equal to 1000. It cannot be less than 0 or higher than 5000. If you want to deactivate preloading data, set PreLoadBars = 0.

The value selected is a maximum because the amount of bars that can be preloaded depends on the amount of data available for a given instrument and timeframe.

Example:

```
DEFPARAM PreLoadBars = 300
a = (close + open) / 2
If price CROSSES OVER Average[250](a) THEN
   BUY 1 SHARE AT MARKET
Endif
```

If PreLoadBars is set to 300, it means that a moving average of 250 bars of a would be defined at the very first bar after a trading system started. This would not be the case if only 200 bars were preloaded.

Note that the value of 300 is a maximum: If less than 300 bars are available prior to the start of the trading system, only the available number of bars will be preloaded.

In the example where 300 bars are preloaded, the BarIndex of the first bar after the start of the trading system is equal to 300. On the other hand, if 0 bars are preloaded, the BarIndex of the first bar after the start of the strategy would be equal to zero.

FlatBefore and FlatAfter

```
DEFPARAM FlatBefore = HHMMSS
DEFPARAM FlatAfter = HHMMSS
```

HHMMSS is a time where HH indicates the hour, MM indicates the minutes and SS indicates the seconds.

These instructions let you cancel any pending orders, close all positions and prevent placement of additional orders before a certain time of day in the case of FlatBefore or after a certain time of day in the case of FlatAfter in the time zone of the strategy.

The parameter FlatBefore must always be later than the market opening time (customized or not), and FlatAfter must be earlier than the standard market close (customized or not), otherwise they would have no effect. If the chosen time is not a multiple of the main timeframe of the trading system, (it occurs in the middle of a candlestick), the instruction DEFPARAM FlatAfter will take effect at the close of that candlestick and the instruction DEFPARAM FlatBefore will be applied until the close of the preceding candlestick.

Orders are restricted during this period, meaning that no orders will be placed and any such orders will not be placed at the opening of the next period when the trading system is authorized to place orders. As a result "OnMarket" type variables will always be false during these times.

Example:

```
DEFPARAM FlatBefore = 093000 // Cancel any pending orders, close any positions and prevent placement of additional orders by the trading system before 9:30:00 in the time zone of the strategy.
DEFPARAM FlatAfter = 160000 // Cancel any pending orders, close any positions and prevent placement of additional orders by the trading system after 16:00:00 in the time zone of the strategy.
```

NoCashUpdate (backtest only)

```
DEFPARAM NoCashUpdate = True
```

If this option is activated, the available cash is not updated with gains, losses and brokerage fees. By default, NoCashUpdate = False.

Example:

Initial capital 10 000 €, with NoCashUpdate = True. The maximum investment will be limited to 10 000€, whatever the gains and losses realized for the entire duration of the backtest.

Note:

Parameters defined with the DEFPARAM instruction must be defined in the first lines of the code (after any comments).

MinOrder and MaxOrder (backtest only)

```
DEFPARAM MinOrder = n
DEFPARAM MaxOrder = p
```

This option lets you block all orders whose quantity (in lots, contracts or shares) is below n or above p.

Example:

```
DEFPARAM MinOrder = 100
Buy 1000 cash at market
```

If the current price is above 10€, the order quantity will be below 100 therefore the order will be rejected.

Calling indicators

ProRealTime indicators

All of the functions including ProRealTime indicators available for programming your own indicators are also accessible to program trading systems (see the glossary at the end of the manual for a complete list).

We advise you to check the ProBuilder manual for more detail about these functions.

The quantity of historical data necessary to calculate an indicator depends on the type of indicator.

For example, to calculate an exponential moving average over n periods (ExponentialAverage[N]), we generally consider that 10*N bars are necessary to obtain a precise result.

If the beginning of the backtest is very close to the beginning of the chart, additional history may be provided by the server for the calculation of the trading system so that the indicators have results at the beginning of the bar.

Personal Indicators

It is possible to call ProBuilder indicators that you have programmed using the “CALL” instruction in a trading system.

Example:

```
a,b = CALL "HistoMACD[5,6]" // a and b are the outputs of the function. 5 & 6 are the inputs.
```

To learn more about optimal use of the CALL function, read the section dedicated to optimizing your programs carefully.

Programming advice

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

The time required to calculate a trading system is strongly dependent on the complexity of the indicators used and the way they are called. The following paragraph provides some simple advice to optimize your codes from a programming point of view.

Reduce the number of calls to indicators

If you use the same indicator more than once in a program, stock the indicator in an intermediary variable (avg40 in the example below) rather than calling the indicator again. This will speed up the execution significantly.

NON-OPTIMAL CODE	OPTIMAL CODE

`IF NOT LONGONMARKET AND close > Average[40] THEN`
`  BUY 1 SHARE AT MARKET`
`ENDIF`
 	
`IF NOT SHORTONMARKET AND close < Average[40] THEN`
`  SELLSHORT 1 SHARE AT MARKET`
`ENDIF`
	`avg40 = Average[40]`
	`IF NOT LONGONMARKET AND close > avg40 THEN`
	`  BUY 1 SHARE AT MARKET`
	`ENDIF`
	
	`IF NOT SHORTONMARKET AND close < avg40 THEN`
	`  SELLSHORT 1 SHARE AT MARKET`
	`ENDIF`

This is also valid if you want to use the same indicator several times but with a different offset.

NON-OPTIMAL CODE	OPTIMAL CODE

`a = ExponentialAverage[40](close)`
`b = ExponentialAverage[40](close[1])`
`c = ExponentialAverage[40](close)`
 	
`IF a > b THEN`
`  BUY 1 SHARE AT MARKET`
`ENDIF`
 	
`IF a < c[1] THEN`
`  SELLSHORT 1 SHARE AT MARKET`
`ENDIF`
	`a = ExponentialAverage[40](close)`
	
	`IF a > a[1] THEN`
	`  BUY 1 SHARE AT MARKET`
	`ENDIF`
	
	`IF a < a[1] THEN`
	`  SELLSHORT 1 SHARE AT MARKET`
	`ENDIF`

Calls to personal indicators

Calling personal indicators with the **CALL** instruction is more costly in calculation time using ProRealTime indicators. For ProRealTime indicators, we know in advance what calculations are necessary and can control how they are done. This lets us increase the speed of calculations which is not possible with personal indicators which the user programs themselves.

To improve the execution speed of a trading system with the **CALL** instruction, it is important to use **CALL** as efficiently as possible in the program.

Limit the number of identical calls:

As for ProRealTime indicators, limit the number of times an indicator is called in the program.

NON-OPTIMAL CODE	OPTIMAL CODE

`myindic1 = CALL "My Function"`
 	
`IF NOT LONGONMARKET AND close > myindic1 THEN`
`  BUY 1 SHARE AT MARKET`
`ENDIF`
 	
`myindic2 = CALL "My Function"`
`IF NOT SHORTONMARKET AND close < myindic2 THEN`
`  SELLSHORT 1 SHARE AT MARKET`
`ENDIF`
	`myindic = CALL "My Function"`
	
	`IF NOT LONGONMARKET AND close > myindic THEN`
	`  BUY 1 SHARE AT MARKET`
	`ENDIF`
	
	`IF NOT SHORTONMARKET AND close < myindic THEN`
	`  SELLSHORT 1 SHARE AT MARKET`
	`ENDIF`

Limit nested calls:

If you are using a personal indicator in the code of your backtest, check that this personal indicator does not have a **CALL** instruction in its code.

Calling a personal indicator which calls another personal indicator is costly in terms of calculation time. If possible, always duplicate the code of the personal indicator you want to call directly in your ProBacktest rather than using the **CALL** function so that your backtest codes only use standard ProRealTime indicators.

NON-OPTIMAL CODE	OPTIMAL CODE

Code of the trading system:
`myindic = CALL "MyDCLOSEMix LinearReg"`
 	
`IF NOT longonmarket AND close > myindic THEN`
`  BUY 1 SHARE AT MARKET`
`ENDIF`
 	
`IF NOT shortonmarket AND close < myindic THEN`
`  SELLSHORT 1 SHARE AT MARKET`
`ENDIF`
 	
Code of "MyDCLOSEMix LinearReg":
`dayclosemix = CALL "MyDCLOSEMix"`
 	
`RETURN LinearRegression[5](dayclosemix)`
 	
Code of "MyDCLOSEMix":
`mix = (DClose(0) + 3.5 * DClose(1) + 4.5 * DClose(2) + 3 * DClose(3) + 0.5 * DClose(4) - 0.5 * DClose(5) - 1.5 * DClose(6)) / 10.5`
 	
`RETURN mix`
	Code of the trading system:
	`dayclosemix = (DClose(0) + 3.5 * DClose(1) + 4.5 * DClose(2) + 3 * DClose(3) + 0.5 * DClose(4) - 0.5 * DClose(5) - 1.5 * DClose(6)) / 10.5`
	`myindic = LinearRegression[5](dayclosemix)`
	
	`IF NOT longonmarket AND close > myindic THEN`
	`  BUY 1 SHARE AT MARKET`
	`ENDIF`
	
	`IF NOT shortonmarket AND close < myindic THEN`
	`  SELLSHORT 1 SHARE AT MARKET`
	`ENDIF`

Limit nested loops:

For all conditional instructions (IF...THEN...ENDIF), it is always preferable in terms of calculation time to use one condition that verifies n conditions rather than to use n instructions as shown below.

NON-OPTIMAL CODE	OPTIMAL CODE

IF CLOSE >= 0.0014 THEN
  IF CLOSE <= 0.0047 THEN
  IF INTRADAYBARINDEX >= 5 THEN
  IF INTRADAYBARINDEX <= 20 THEN
  IF NOT SHORTONMARKET THEN
  BUY 1 SHARES AT MARKET
  ENDIF
  ENDIF
  ENDIF
  ENDIF
ENDIF
	IF CLOSE >= 0.0014 AND CLOSE <= 0.0047 AND INTRADAYBARINDEX >= 5 AND INTRADAYBARINDEX <= 20 THEN
	  IF NOT SHORTONMARKET THEN
	  BUY 1 SHARES AT MARKET
	  ENDIF
	ENDIF

Use of for loops:

Use of FOR loops is sometimes necessary, but its better to limit their use when possible as they increase calculation time.

Here are several separate examples where using a FOR loop is avoided:

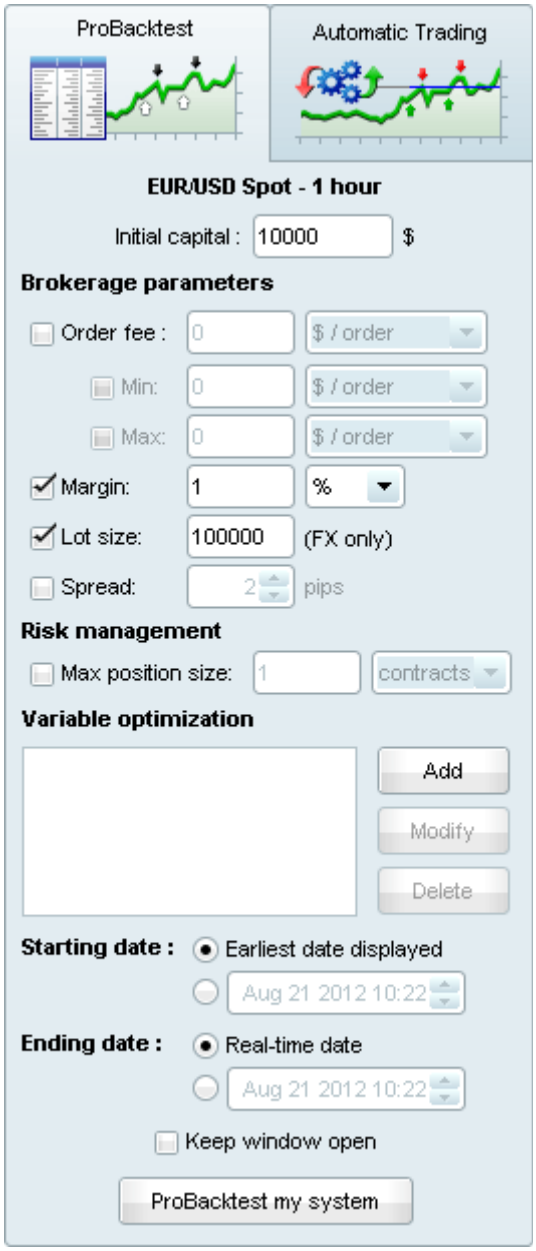
```
// Determine if the condition C1 is true for at least once over the last n candlesticks:
IF HIGHEST[n](c1) = 1 THEN
   ...
// Determine if the condition c1 was always true over the last n candlesticks:
IF LOWEST[n](c1) = 0 THEN
   ...
// Determine the number of times the condition c1 was verified over the n last candlesticks:
num = SUMMATION[n](c1)

// Determine the number of bars since c1 was true:
IF c1 THEN
   lastoccurence = barindex
ENDIF
timesince = barindex - lastoccurence

// Find the maximum of variables (a,b,c,d,e,f,g):
top = MAX(a , MAX(b , MAX(c , MAX(d , MAX(e , MAX(f , g) ) ) ) ) )
```

ProBacktest – trading system backtesting

The “ProBacktest” tab of the trading system creation window lets you configure the parameters of the system:



- Initial Capital
- Brokerage parameters & Risk management
- Period of execution
- Variable optimization

Money Management

Initial capital

This section lets you assign the capital that is available for the trading system to trade with. The maximum authorized investment of the system depends on this.

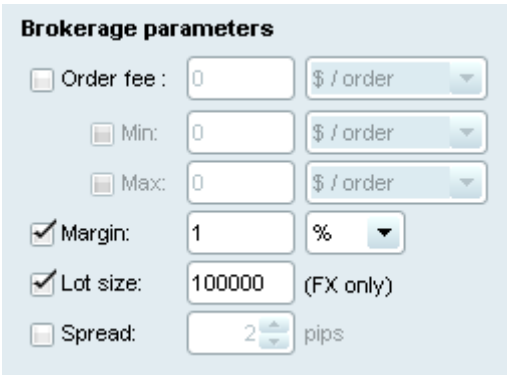
During the execution of a backtest, gains, losses and brokerage fees effect the amount of capital available to a trading system (unless you activate the **NoCashUpdate** setting). See the section **Reinvest gains** for more information. For automatic trading systems, the capital available to your systems is the capital in your portfolio.

If a backtest does not place any trades, try increasing the amount of initial capital.

Brokerage fees and risk management

You can customize these parameters to accurately reflect the fees and other parameters of your broker. Any kind of brokerage fee can be applied to any type of instrument. The types of brokerage fees available include:

- Cash per order: Fixed amount of cash (in the currency of the instrument) applied every time an order is executed. You can also specify a minimum and a maximum value per order.
- % Transaction: percentage of the transaction (in the currency of the instrument) applied every time an order is executed
- Cash per lot: Fixed amount of cash (in the currency of the instrument) applied per lot or contract
- Margin: Deposit required in % or cash per lot. It determines the max. leverage (=100/margin)
- Lot size (Forex only): It's the minimum order quantity on the instrument. Every order quantity entered in BUY/SELL instruction is multiplied by the lotsize.
- Spread (in pips): value added to the mid price to reflect the bid-ask spread.



The risk management section lets you define max position size: Any order that tries to increase the position size beyond this will be rejected. Max position size is typically set in number of contracts for futures and forex and cash or % of capital for stocks. Parameters in the "Risk management" section do not take brokerage fees into account.

Futures:

For futures, Brokerage fees are typically defined as a fee per lot and per transaction.

The margin is the cash necessary to buy a contract. The value of a point is automatically calculated by the workstation for the future on which you are backtesting the trading system.

Here are the point values of the main futures.

FUTURE NAME	POINT VALUE
FCE CAC 40	10€
DAX	25€
DJ Eurostoxx 50	10€
BUND	10€
Euro FX	12,5$
Mini S&P 500	50$
Mini Nasdaq 100	20$
Mini Dow	5$

Forex:

The spread, lot size and margin can be defined and are applied to each order.

Example with EURUSD:

Lot size: 100 000

Spread: 2 pips

Margin: 5%

The instruction `BUY 1 LOT AT MARKET` on EURUSD buys 1 lot of 100 000 with a spread of 2 pips. (0.0002). As the leverage is 20:1 (5% margin), a deposit of 5 000 € is necessary to place the order.

Stocks:

The brokerage fees are usually defined in fees per order in € or in % of the transaction. It's also possible to define minimum or maximum brokerage fees per transaction.

Example configuration for stocks:

- Fee per order: $10
- Margin: 20%

With this configuration, each order may have leverage of up to 5:1.

Variable optimization

Variable optimization lets you test different combinations of variables in a backtest to see which combinations give the best results for a given instrument and timeframe over the period of historical data tested.

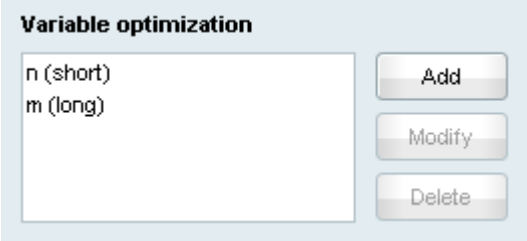
To learn more and see an example, we suggest you watch the video "**Money management, stops and optimization**".

The result of the optimization is presented in an “Optimization report”. You will see the statistics of the best combinations of variables tested and use this information to determine which variable you want to use in your trading system.

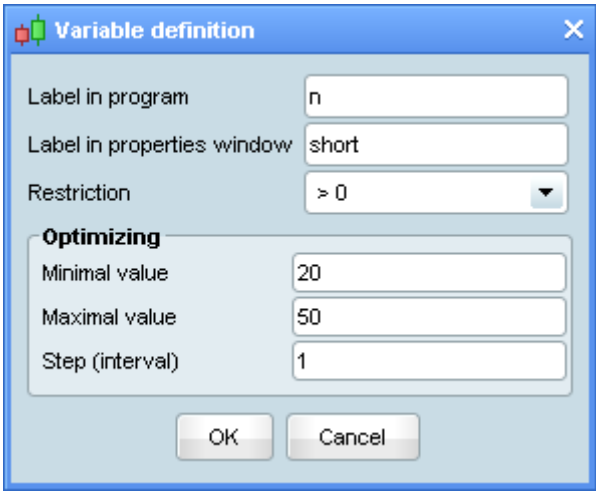
Here is an example of a program that can be optimized with 2 moving averages with periods of n and m.:

```
AVGm = ExponentialAverage[m](Close)
AVGn = ExponentialAverage[n](Close)
IF AVGm CROSSES OVER AVGn THEN
   BUY 100 SHARES AT MARKET
ENDIF
IF AVGm CROSSES UNDER AVGn THEN
   SELL 100 SHARES AT MARKET
ENDIF
```

The variables n and m can then be defined by clicking on the “Add” button in the optimization section:

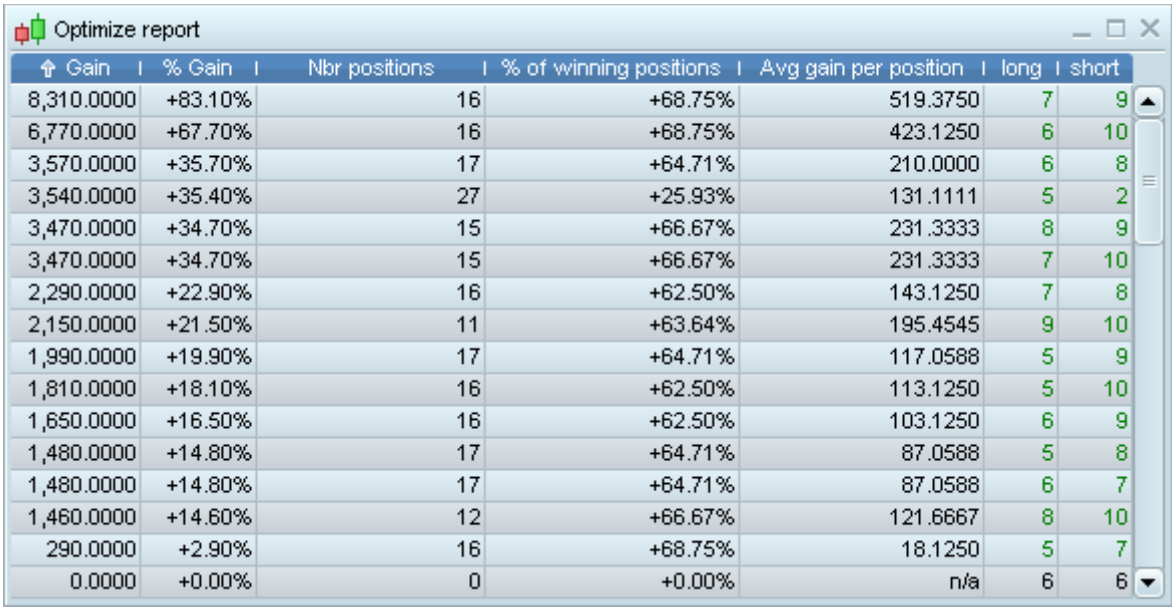


The following window then opens where you can set up the optimization:



- **"Label in program"** is the name of the variable in the code (n in this case). Variable names are case-sensitive.
- **"Label in the properties window"** is a label that you can give to the variable to recognize it more easily (for example “short” or “number of periods” for n).
- **"Minimal value" et "Maximal value"** are the limits of the variable for the optimization test.
- **"Step"** is the interval of variables to be tested in the optimization.

Here is an example of an optimization report:


Optimize report

Gain	% Gain	Nbr positions	% of winning positions	Avg gain per position	long	short
8,310.0000	+83.10%	16	+68.75%	519.3750	7	9
6,770.0000	+67.70%	16	+68.75%	423.1250	6	10
3,570.0000	+35.70%	17	+64.71%	210.0000	6	8
3,540.0000	+35.40%	27	+25.93%	131.1111	5	2
3,470.0000	+34.70%	15	+66.67%	231.3333	8	9
3,470.0000	+34.70%	15	+66.67%	231.3333	7	10
2,290.0000	+22.90%	16	+62.50%	143.1250	7	8
2,150.0000	+21.50%	11	+63.64%	195.4545	9	10
1,990.0000	+19.90%	17	+64.71%	117.0588	5	9
1,810.0000	+18.10%	16	+62.50%	113.1250	5	10
1,650.0000	+16.50%	16	+62.50%	103.1250	6	9
1,480.0000	+14.80%	17	+64.71%	87.0588	5	8
1,480.0000	+14.80%	17	+64.71%	87.0588	6	7
1,460.0000	+14.60%	12	+66.67%	121.6667	8	10
290.0000	+2.90%	16	+68.75%	18.1250	5	7
0.0000	+0.00%	0	+0.00%	n/a	6	6

The optimization report gives 5 statistics for each combination of variables tested. These statistics are as follows:

- **"Gains"**, is the gain or loss realized by the trading system. The calculation formula is:

Gain = Final capital – Initial capital

This statistic lets you evaluate the absolute gain potential with the trading system defined for the historical period tested and for each variable combination.

Note: Brokerage fees as defined in the “Brokerage parameters” section are taken into account in this calculation.

- **"%Gains"**, is the gain or loss in %. The calculation formula is:

%Gain = 100 x Profit / Initial capital

This indicates the relative performance of the backtest configured with the corresponding variables.

- **"Nb positions"** indicates the number of positions opened during the backtest.

- **"% winning"** indicates the % of winning positions. It is calculated as:

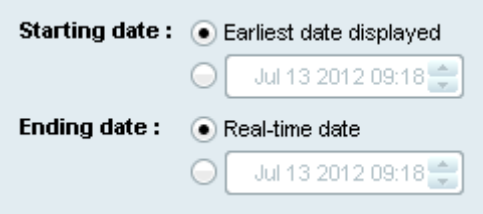
% winning = (100 x number of winning positions) / Number of positions

- **"Avg gain per position"** is the average gain per position. It can be useful to determine efficiency of orders placed. It is defined as:

Avg gain per position = Gain / Number of positions

Note: The results of optimization reports may change for a given trading system depending on the security, timeframe or amount of historical data used.

Definition of the period of execution of the backtest



This area lets you define the beginning and end of the backtest. Note that the amount of data that can be used for the backtest is generally limited to the amount of data displayed on the chart. You can increase the amount of historical data loaded on your chart by using the left dropdown menu of each chart. You need to load the amount of data on which you want to run the test before executing the backtest.

In the case of a “real-time” backtest, the orders appear on the chart whenever a signal is triggered. It is also possible to associate these orders to popups or sounds from by selecting “Alert and sound configuration” from the “Options” menu.

If an ending date is defined, all positions opened at that date are closed.

Note:

If your backtest takes a long time to execute, you can reduce the period of execution: the amount of time it takes to backtest a trading system is proportional to the amount of historical data on which the trading system is tested.

After executing a backtest, the results are displayed in the following manner:

- Equity curve showing the gain and losses of the trading system
- Positions histogram
- Detailed report

For more information about the display of results of backtests, check Annex A at the end of this document.

Customization of trading hours for backtesting

The menu "Options / Set time zones and trading hours" lets you define customized trading hours for a market.

Customized trading hours: If you define reduced trading hours for a market, that means that only data during these reduced trading hours will be shown on charts (and taken into account by ProBacktest). Note that it is only possible to define reduced trading hours within a given day. For example, a the market is open 24 hours per day, you could choose to take into account only data between 10:00 and 14:00, but it is not possible to take into account data between 21:00 one day and 9:00 the next day. If an order is placed at the customized close of the last bar of the trading day with customized trading hours configured, this order will be placed at the customized open of the next trading day.

Notes concerning customized time zones and weekend data:

Some 24-hour markets have options to "use intraday quotes to build daily candles". This option is not taken into account by ProBacktest, which always uses official daily candles based on the standard (local market) time zone.

Some markets (such as Forex) include weekend data. There is a checkbox for these markets in "Options / Set time zones and trading hours" which allows the users do hide weekend data in the charts. Weekend data is always taken into account for backtesting purposes. For the Forex market, Sunday's data is included in the Monday daily candle for backtesting purposes (there is no daily candle for Sunday).

Notes concerning customized time zones:

The code is always executed in the user's time zone. This means that time-based instructions (time, flatafter, flatbefore) will take this time zone into account for their calculation. The user's timezone means the time zone chosen for the market to which the instrument belongs. It is customizable in the menu Platform Options > Time zones & Trading hours.

By default, the time zone of the instrument is the time zone of your computer.

Its possible to change the time zone of a market from the Platform Options > Time zones & Trading hours menu. In case of a modification, the modified time zone will be taken into account the next time you run a backtest.

Example: On Vodafone (on the LSE – timezone GMT+1 during summer time or BST), I set the chart to Paris time (GMT+2 during summer time or CET), the instruction time will return 1300 (time of the close of the 15 minute candle beginning at 1145 BST: 12:00 converted to 13:00 CET).

All intraday time instructions are concerned:

- Time and its derivative instructions (hour, minute...)
- Opentime and its derivative instructions (openhour, openminute...)
- Flatafter and Flatbefore
- IntradayBarIndex (reset to zero at the open of the market in the user's time zone)

Daily time-based instructions are not affected by the time zone selected:

- Dopen, Dhigh, Dlow, Dclose
- Date and its derivative instructions (year, month, day)
- DayOfWeek and Days

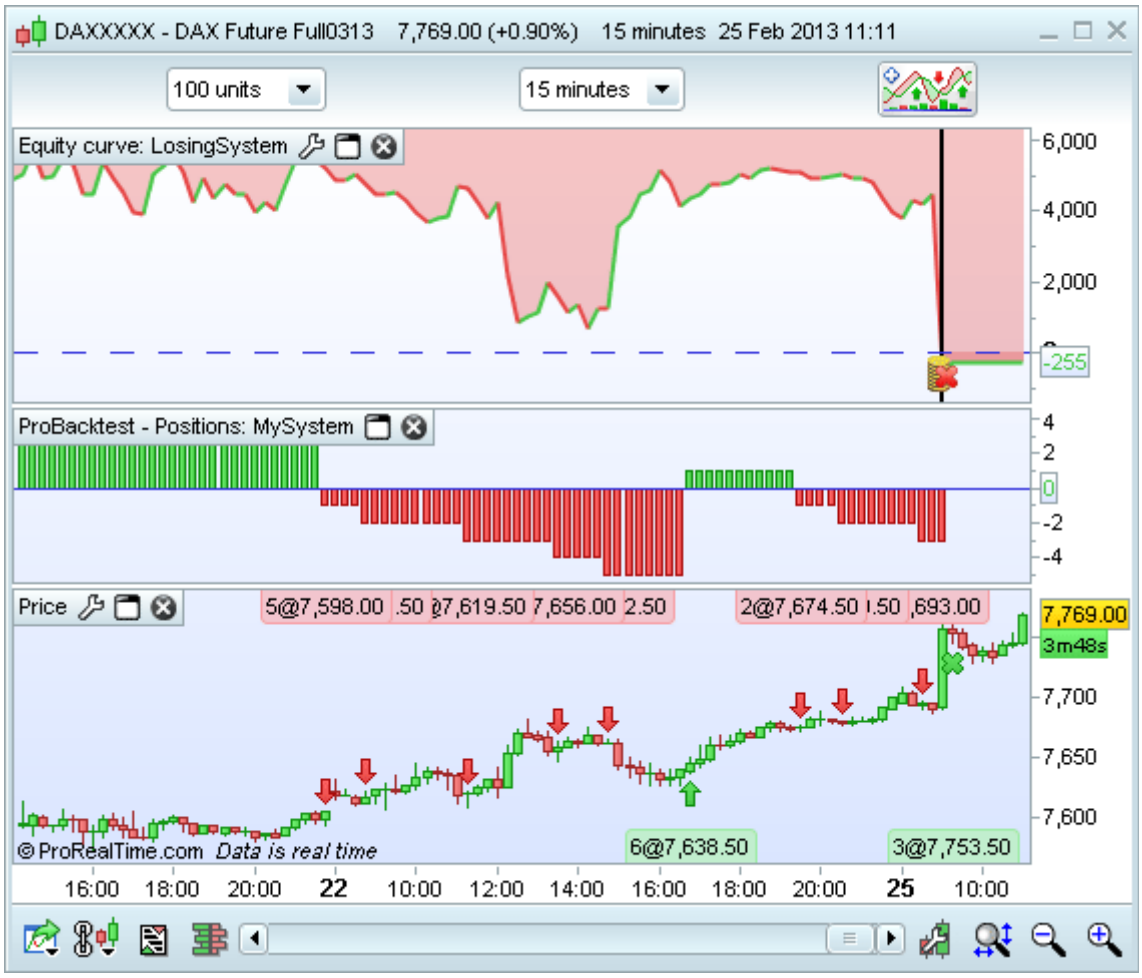
These instructions take into account the time zone of the local market.

Reasons a ProBacktest may stop

A ProBacktest may stop in one of the following cases:

- The backtest reaches the end time specified in the programming window. In this case, the end of the backtest is shown only by a black vertical line on the chart.
- There is a "Quit" instruction in the code which is executed. In this case, the end of the backtest is shown by the following icon: 
- The available capital no longer sufficient to cover losses ("estimated" capital is negative). In this case, the end of the backtest will be shown by this icon 
- An order is rejected due to insufficient cash. This order will appear in the order list of the detailed report. In this case, the end of the backtest will be shown by this icon: 

Here is an example of a backtest stopped due to insufficient capital:



Display of the values of backtest variables (in ProRealTime 10.2 and higher)

The GRAPH instruction lets you display the values of variables you use in your ProBacktest program.

This instruction works in the following way:

```
GRAPH myvariable AS "my variable name"
```

- `myvariable` is the name of the variable in the code
- `"my variable name"` is the label of the variable which will be displayed on the chart

It is also possible to define a colour for the variable with the optional COLOURED parameter:

```
GRAPH myvariable COLOURED (r,g,b) AS "my variable name"
```

"r","g", and "b" are whole numbers from 0 to 255 (RGB format)

ex: (255,0,0) for a red curve

You can also set the transparency of the line as follows:

```
GRAPH myvariable COLOURED (r,g,b,a) AS "my variable name"
```

a is a whole number between 0 and 255 indicating the level of transparency. (0 for a completely transparent line, 255 for a completely opaque line)

A new chart panel appears under the equity curve panel of your backtest. This panel contains the values of `myvariable` at the close of each bar, as shown in the example below:

Note:

The GRAPH instruction cannot be used in automatic trading mode.

5 variables maximum may be displayed per chart with the GRAPH instruction. Any additional variable will be ignored by this instruction.

ProOrder Automatic Trading

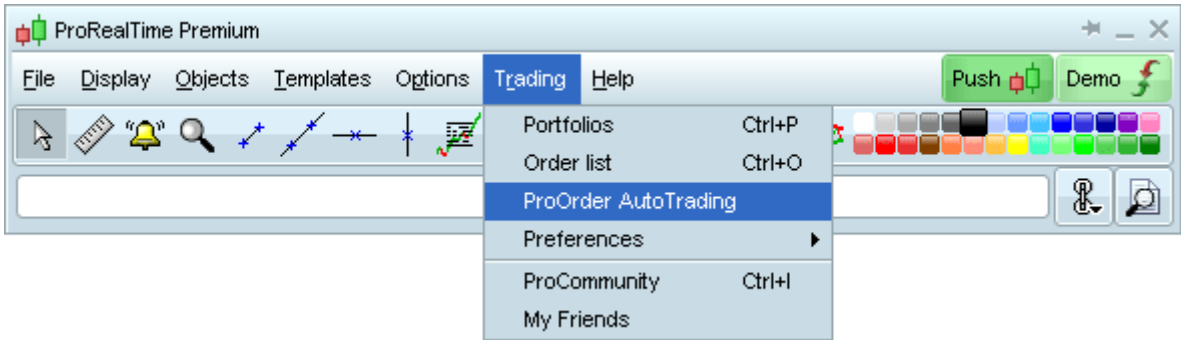
This part of the manual explains how to take trading systems which you have previously backtested and execute them as automatic trading systems.

- The first section explains how to send a trading system to ProOrder to prepare it for execution as an automatic trading system.
- The second section explains how to start a trading system and check the results.
- The third section explains the parameters of trading systems and their conditions of execution.
- The fourth section explains how manual and automatic trading systems coexist in the workstation.
- The fifth section explains implications of executing multiple automatic trading systems on the same instrument.
- The last section contains a list of indicators which cannot be used in automatic trading due to their method of calculation.

It is recommended that you read the entire manual before starting a trading system to learn about the execution of trading systems.

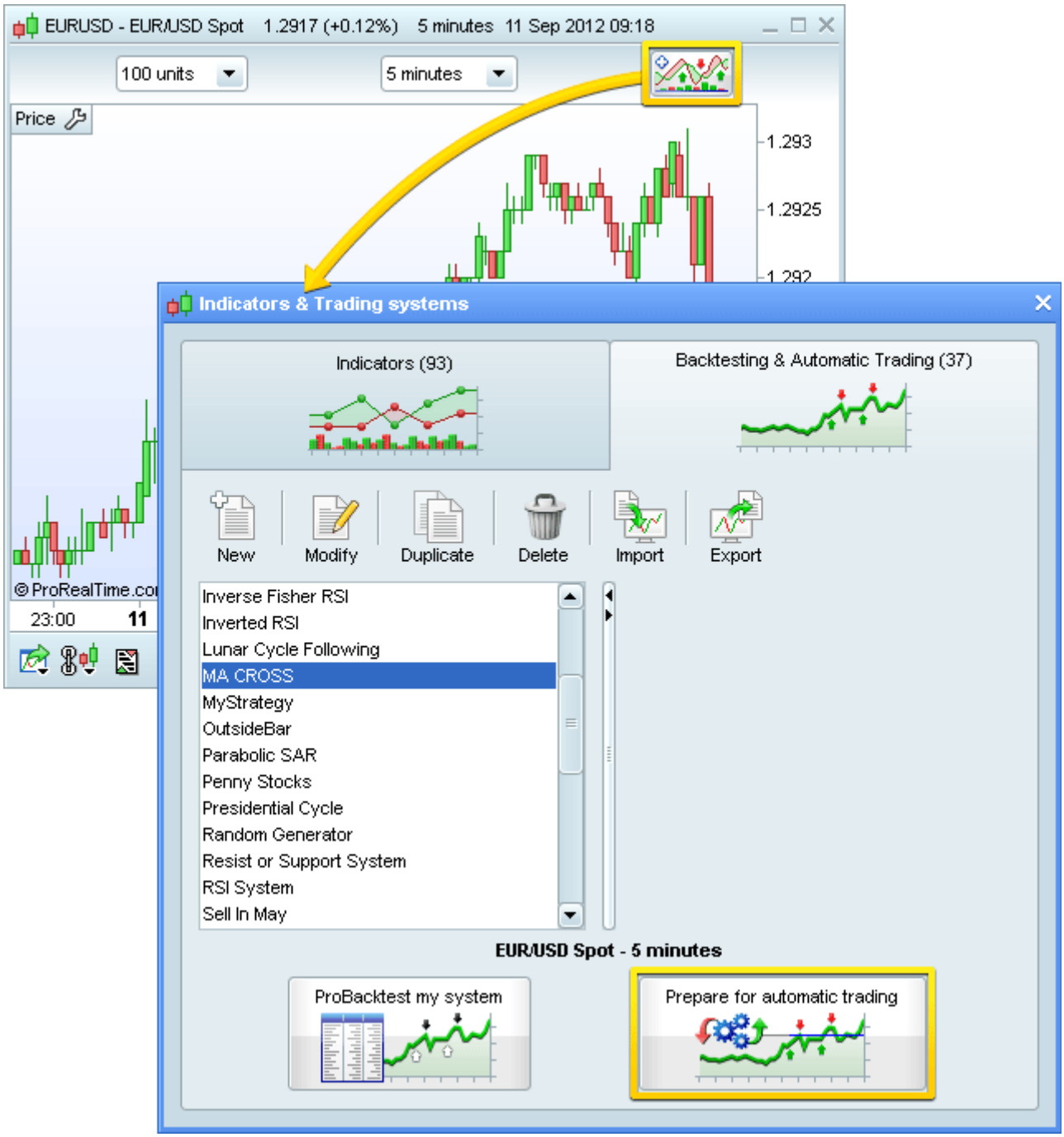
Prepare a trading system for automatic execution

Begin by opening the ProOrder window from the trading menu:



The following window will be displayed containing instructions to prepare a trading system for automatic trading.

First choose the chart and timeframe you want to execute your trading system on and clicked the  button. The Indicators & Trading systems window will appear. Click on "Backtesting and automatic trading" to see the list of your trading systems:



Choose the trading system you want to execute automatically and push the button "Prepare for automatic trading". The trading system will then appear in ProOrder.

How to start a trading system in ProOrder and check the results

Once you have added a trading system to ProOrder, you can define the maximum position size for this system then begin trading with it by pressing the start button:

A popup window will ask you to confirm that you want to execute the system which you should read carefully.

Note that the “Max position size” which can be set in the ProOrder window prior to starting a trading system takes precedence over quantities in the code. Max position size for futures and forex is defined in number of lots or contracts. For example, if your code has an instruction to buy 3 lots, but you limit the max position size to 1, this buy 3 order will be ignored. Similarly, if your code has an instruction to buy 1 lot, then sellshort 3 lots, the sellshort order will be ignored and you will remain in the long 1 position. You should always check the max position size prior to executing a code.

For stocks, max position size is defined in cash (not including brokerage fees).

After you push the “Start” button, the system will be displayed in the "running" section as shown below.

Once a trading system has started, its position, latent gain and total gain will be shown in the ProOrder window. It is possible to click on the link in the "version" section to see a copy of the code of this system.

You can also click the button shown in yellow below to see the equity curve of the system and a detailed report on its performance.

Here is an example of an equity curve of a running system and its detailed report:

MA CROSS - Trading detailed report - EUR/USD Spot

EUR/USD Spot	MA CROSS Version: 11-Sep-2012 09:18:44	Start: 11 Sep 2012 09:19:09 [0 USD] Current: 12 Sep 2012 08:15:00 [930 USD]

Statistics of closed positions | Orders list | Closed positions list

	All positions	Long positions	Short positions
Gain:	2,090 USD	2,090 USD	0 USD
Gross profit:	2,410	2,410	0
Gross loss:	-320	-320	0
Profit/Loss ratio:	7.53	7.53	n/a
Nbr positions:	9	9	0
% of winning positions:	55.56%	55.56%	0%
Winning / Even / Losing:	5 0 4	5 0 4	0 0 0
Avg gain:	232.22	232.22	n/a
Avg gain of winning positions:	482	482	n/a
Avg loss of losing positions:	-80	-80	n/a
Profit best position:	1,110	1,110	n/a
Loss worst position:	-190	-190	n/a
Standard deviation on P&L :	476.63	476.63	n/a
Max Drawdown:	760	710	0
% Max Drawdown:	447.06%	417.65%	0%
Max consecutive losses:	3	3	0
Max Runup:	1,860	1,860	0
% Max Runup:	146.46%	146.46%	0%
Max consecutive wins:	3	3	0
Avg orders exec/day:	n/a	n/a	n/a
Avg orders exec/month	n/a	n/a	n/a
Brokerage fees:	0	0	0

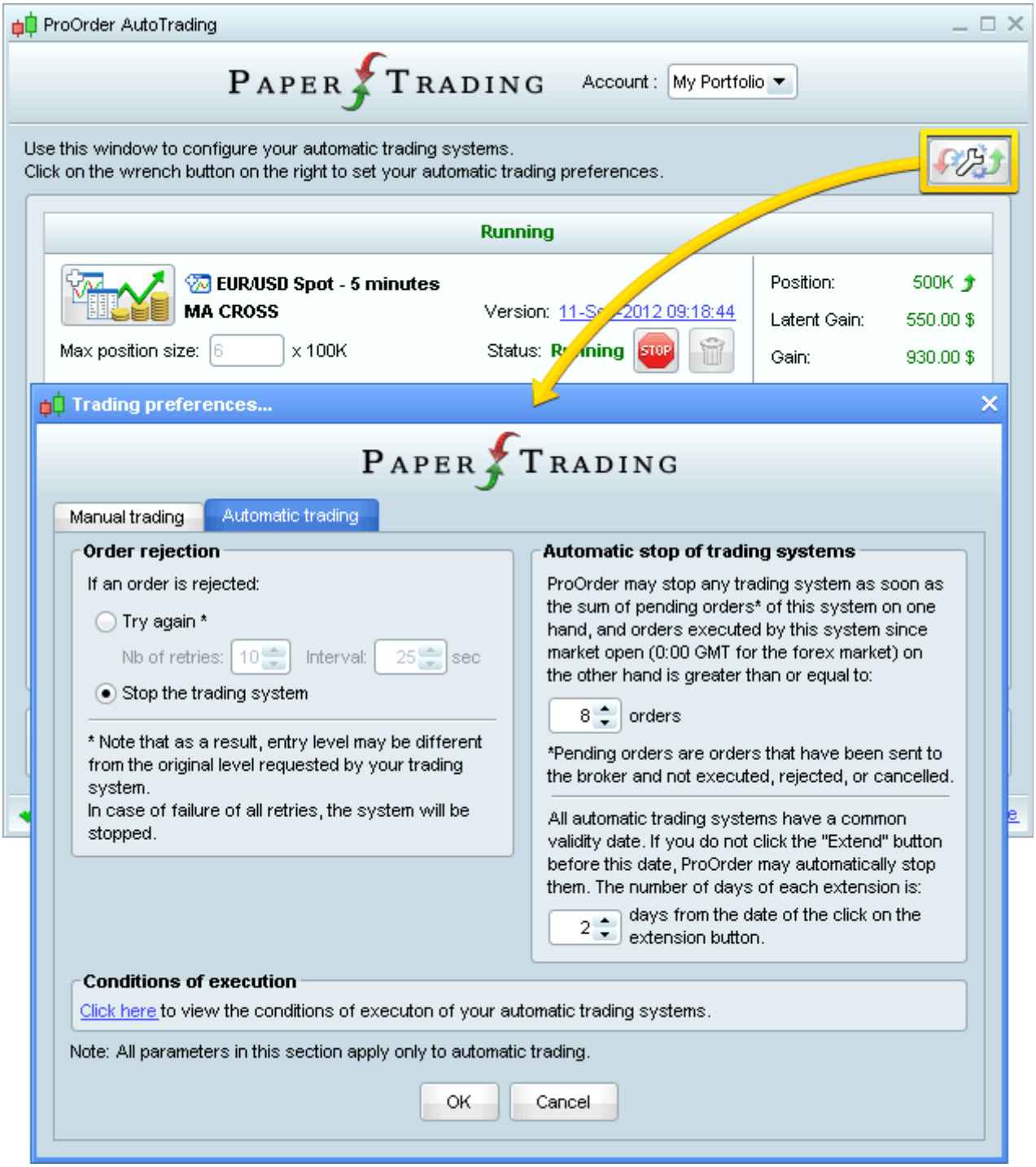
The statistics above relate to past data. Past performance is not indicative of future results.

Note: The gain in the section “Statistics of closed positions” may be different from the value of the equity curve because the system is still running and the equity curve takes into account positions that are still open.

Automatic trading parameters and conditions of execution

Trading system parameters

Before executing any trading system, you should click the wrench icon shown in yellow below to configure your trading preferences:



A link to a showing the conditions of execution of your trading systems is also included at the bottom of this window (“Click here”). You should read these conditions carefully.

Automatic stop of trading systems

Validity date: All running trading systems have a common validity date. If you do not click the "Extend" button before this date, ProOrder may automatically stop them. You can view the validity date in the ProOrder window (expressed in your computer's time zone) and extend your trading systems validity via the "Extend" button at the bottom of the ProOrder window when a trading system is running:

The amount of time of each extension can be configured from the “Automatic Trading” tab in the “Trading preferences” window. It is possible increase this parameter while you have trading systems running. Your change will be applied to the next extension you make.

Note: if you modify your computer's timezone while the platform is open, you will need to restart your workstation in order for the “Valid until” time to be displayed in the new timezone.

Number of orders placed: ProOrder may stop any given trading system as soon as the sum of pending orders placed by this system on one hand and number of orders executed by this system since market open on the other hand (0:00 GMT for the forex market) of this system is greater than or equal to the quantity chosen in the "Automatic trading" tab of the "Trading preferences" window. A pending order is an order that was sent to the broker and not executed, rejected, or canceled.

For example, each "Set stop" or "Set Trailing stop" or "Set target" instruction as long as the corresponding order has not been canceled, rejected, or executed.

In addition, 3 different limit orders or 3 different stop orders that have not been canceled, rejected or executed will count as 3 pending orders. This is true if the 3 orders are on the same price level or on different price levels.

For example, if you choose a stopping level of 8 orders and since the market open 5 orders have been

executed by a given trading system and this system has 2 pending orders (one "set target" and one "set stop") and the system needs to send an additional order to the market: this 8th order will not be sent (5+2+1 reaches the stopping level), this trading system will be stopped with its pending orders canceled first, and then its position closed.

It is possible increase this parameter while you have trading systems running.

Order rejection: ProOrder may stop any given trading system if too many orders of this trading system are rejected. You can choose to stop a trading system in case of a single order rejection or choose a fixed number of retries. It is not possible to change order rejection parameters while a trading system is running.

Co-existence of manual and automatic trading in the workstation

If a trading system is running on a security, it will no longer be possible to manually trade on that security from the workstation while that system is being executed. You can continue to manually trade on other securities. On securities where trading systems are being executed, manual trading tools will be replaced with an button showing that automatic trading enabled for that security:

It is possible to click on this button to open the ProOrder window and check the trading systems being executed.

Running multiple trading systems on the same security

If you are running multiple trading systems on the same security, your net position is determined by all of these trading systems. For example, if there are 2 trading systems and one buys 1 lot and the other sells 1 lot, your net position will be 0. Only your net position given by all your trading systems will be open in the market at any given time for a given security.

When you open an equity curve of a trading system, you will see a “Positions” chart. This indicates the position of an individual trading system on that security which can be different from your net position on that security (determined by all systems running on that security). The net position is shown by the position line.

Example:

2 trading systems are running on the same security: one is buying 6 lots of size 100 000 each and another is buying 2 lots of size 100 000 each = net position +800 000.

In this example the position of the trading system shown in the chart is +600 000. The net position shown by the position line is +800 000.

The net position of + 800 000 is also shown in the “Trading” > “Portfolios” window:

When you are running multiple trading systems on the same security, each instance of the system running tracks its own current position, orders, trades and gains independently. As a result, the instructions “LongOnMarket”, “ShortOnMarket” tell you whether the currently running system is long or short.

Your net position on a security may be different from the position of a given system. In the same manner, other status variables such as “CountOfLongShares”, “CountOfShortShares”, “CountOfPosition”, “PositionPrice”, “StrategyProfit”, “TradeIndex”, “TradePrice” and “PositionPerf” apply only to the current strategy.

Indicator restrictions

The following indicators may not be used for automatic trading because their mode of calculation does not allow for real-time usage:

- ZigZag: signals based on this indicator are recalculated after the fact and as a result, the signals given in real-time may be very different from signals given during backtests.

Note concerning personnalized time zones and trading hours

When a trading system is sent to ProOrder, the time zone and trading hours which were defined for the market of the instrument are associated to the strategy. These parameters are applied at each launch of the strategy. To modify the time zone and trading hours of a strategy, you need to delete the strategy from Proorder, modify these parameters in the Options > Time Zones & Trading hours menu, then send the strategy to ProOrder again.

See the section "Customization of trading hours for backtests" for instructions concerned by the time zone of the chart and for an explanation of customized trading hours.

Annex A: Display of trading system results

A trading system's results are displayed in 3 complementary ways.

Equity Curve

The equity curve of a backtest shows the profit and loss of the trading system or backtest:

- The **horizontal blue line** corresponds to the initial capital in the case of a backtest or 0 in the case of an automatic trading system. So, in the case of a backtest that had an initial capital of 10 000 and a loss of 2705, the value of the equity curve would be 7295 as shown in the example below. In the case of an automatic trading system with the same loss, the beginning value would be 0 and the ending value would be -2705.

- The **color shading of the equity curve** is green if the performance is positive (gain since the starting point). It is red if the performance is negative.
- The **line of the equity curve** is green when it has increased from the previous point and red when it has decreased.

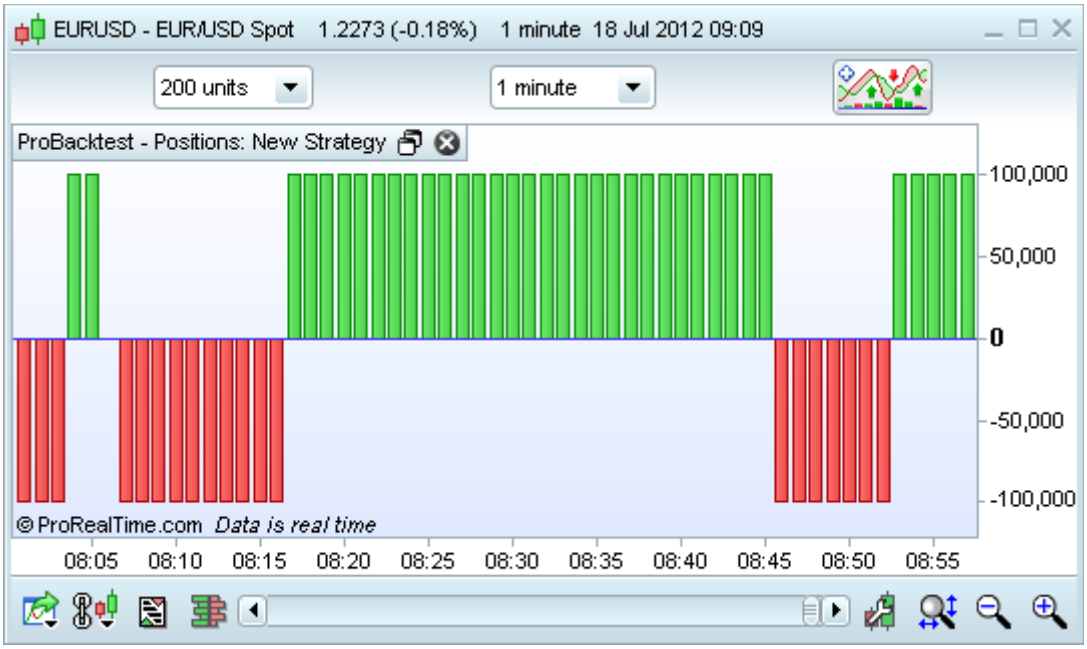
Positions chart

The positions histogram allows you to show in histogram form the evolution of your positions during the trading system simulation.

- A green bar indicates an open long position.
- A red bar indicates an open short position.
- No bar indicates no open position.

Several consecutive bars of the same color indicate the position(s) is still open.

On the vertical axis on the right-side of the chart, you will see how many positions you have open currently (highlighted). In the example below, there is currently a long position open with 1 lot of 100 000 on EUR/USD.



Detailed report

The detailed report lets you view the statistics of your trading system and the details of each position and order:

The detailed report is shown in an independent window made up of 3 tabs:

The **"Statistics of closed positions"** tab gives an exhaustive view of performance of your trading system (Gain or loss, number of winning trades, etc) and indicators of risk such as max drawdown. Note that this tab does not include analysis of currently open positions at the time the report is generated (only closed positions are taken into account). Here is the list of statistics:

- **"Gain"** is the gain or loss realized by the trading system. The calculation formula is:

  Gain = Final capital – Initial capital

  This statistic lets you evaluate the absolute gain potential with the trading system defined for the historical period tested and for each variable combination.

  Note: Brokerage fees as defined in the “Brokerage parameters” section are taken into account in this calculation.

- **"%Gain"** is the gain or loss in %. The calculation formula is:

  %Gain = 100 x Profit / Initial capital

- **"Nb positions"** indicates the number of positions opened during the backtest.

- **"% winning"** indicates the % of winning positions. It is calculated as:

  % winning = (100 x number of winning positions) / Number of positions

- **"Avg gain per position"** is the average gain per position. It can be useful to determine efficiency of orders placed. Average gain per position is particularly important when creating a trading system which places a low number of orders. It is defined as:

  Avg gain per position = Gain / Number of positions

- **"Profit best position"** is the maximum gain on a given position and **"Loss worst position"** is the highest loss on a given position opened since the beginning of the trading system. **"Standard deviation on P&L"** is the standard deviation of results of each position.

- **Max Drawdown** is the maximum potential loss of the trading system. Drawdown is defined as the distance between a given point and the highest point before it on the equity curve:

  $$DD(n)= Max\ t€[0\ ;n]\ P(t) - P(n)$$

  "Max drawdown" is calculated as the largest drawdown over the entire history of the trading system.

  $$MaxDD(N) = Max\ n€[0:N]\ (\ Max\ t€[0;n]\ P(t) - P(n)\ )$$

- **Max Runup** is the potential maximum gain of the trading system. The runup is defined as the difference between a given point and the highest point before it on the equity curve:

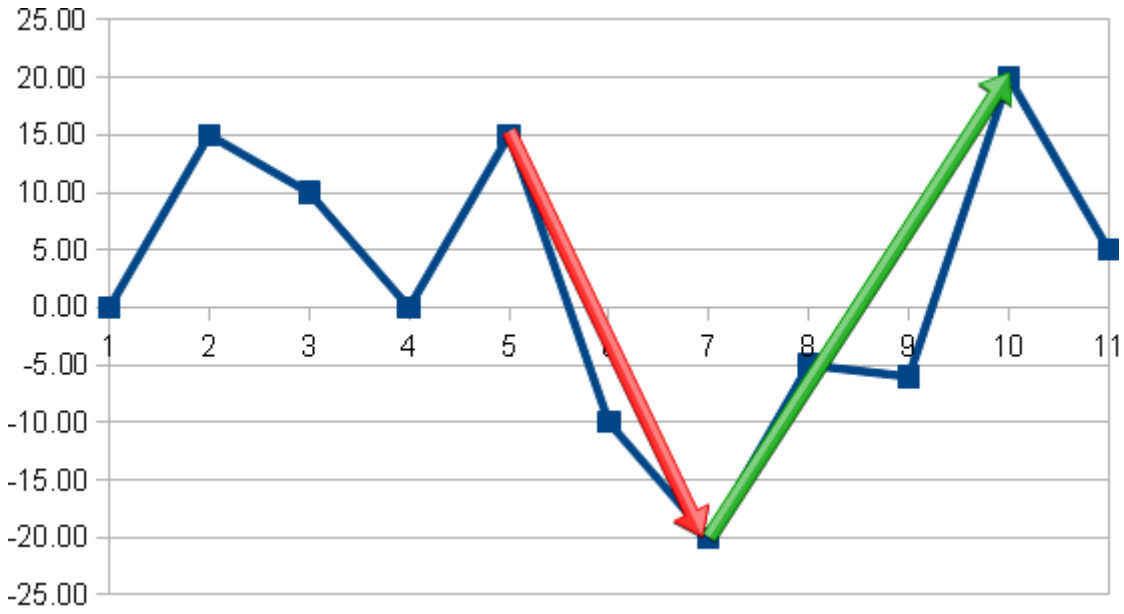
  $$RU(n)= P(n) – Min\ t€[0;n]\ P(t)$$

- The **"Max runup"** is calculated as the maximum of this value over the entire history of the trading system.

  $$MaxRU(N) = Max\ n€[0;N]\ (\ P(n) – Min\ t€[0;n]\ P(t)\ )$$

Example:

BARINDEX	P & L	DRAWDOWN	RUNUP
1	0.00	0.00	0.00
2	15.00	0.00	-15.00
3	10.00	5.00	-10.00
4	0.00	15.00	0.00
5	15.00	0.00	-15.00
6	-10.00	25.00	0.00
7	-20.00	35.00	0.00
8	-5.00	20.00	-15.00
9	-6.00	21.00	-14.00
10	20.00	0.00	-40.00
11	5.00	15.00	-25.00
	Max:	**-35.00**	**40.00**



"% Max risk exposure": Exposure to risk is the relationship between the maximum loss possible for the position and the current amount of capital. The % max risk exposure is then maximum of this value expressed as a percent. The calculations are as follows for stocks, futures and Forex:

- Stocks:

  % Max risk exposure = Max position (quantity * average price / capital) * 100

- Futures:

  % Max risk exposure= Max position (quantity * deposit / capital) * 100

- Forex:

  % Max risk exposure = Max position (quantity * average price * leverage / capital) * 100

Similarly, **"% avg risk exposure"** is the average percent risk exposure.

"Brokerage fees" count total brokerage fees of each order since the beginning of the trading system. These brokerage fees are defined in the settings in the case of a backtest.

% time in the market is calculated as the number of bars with a position open divided by the number of bars of the trading system.

The next 2 tabs give information about orders placed and positions opened and closed closed during the execution of the trading system.

- In **"Order list"** you will find a list of all the orders placed by including their date & time, direction, quantity and price. The times displayed are in the timezone of the instrument.
- In **"Closed positions list"** you will find information about the positions taken by your trading system (long or short, duration in number of bars, performance of each position, opening date and closing date. If there is a position still open at the time the report is generated, it will not be included in this list. In the case of a backtest, if you want to close all positions at the end of the backtest, choose a fixed ending date instead of the real-time date.

Annex B: Detailed examples of codes

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Heiken ashi trading system

This trading system generates a buy signal when a green Heiken Ashi candle appears after a red one.

A sell signal is given if a red Heiken Ashi candle appears after a green one.

This backtest reconstructs the Heiken Ashi view from normal candlesticks. It must be applied to a chart using the normal candlestick style (not the Heiken ashi candlestick style).

```
ONCE PreviousStatus = 0
IF BarIndex = 0 THEN
   XClose = TotalPrice
   XOpen = (Open + Close) / 2
ELSE
   XClose = TotalPrice
   XOpen = (XOpen[1] + Xclose[1]) / 2
ENDIF
IF XClose >= XOpen THEN
   IF PreviousStatus <> 1 THEN
      BUY 1 SHARES AT MARKET
      PreviousStatus = 1
   ENDIF
ELSE
   IF PreviousStatus <> -1 THEN
      SELLSHORT 1 SHARES AT MARKET
      PreviousStatus = -1
   ENDIF
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

ZigZag trading system

This is a backtest based on the ZigZag to determine what would have been the best buy and sell opportunities. The excellent results of this trading system on both stocks and futures are related to the non-predictive character of the ZigZag. The signals are recalculated after the fact and as a result do not give valid signals in real-time.

The reason the results of this trading system are interesting is that they give nearly ideal results that can be compared to other trading systems.

```
// The periods of the zigzag indicator can be optimized using variable optimization
myZigZag = ZigZag[10]
c11 = (myZigZag > myZigZag[1])
c12 = (myZigZag < myZigZag[1])
IF c11 AND NOT LONGONMARKET THEN
   BUY 1 SHARES AT MARKET
ENDIF
IF c12 AND NOT SHORTONMARKET THEN
   SELLSHORT 1 SHARES AT MARKET
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Simple breakout range with trailing stop

This is a basic breakout intraday trading system that takes only long positions. The initial range is determined by the highest and lowest points of the first 2 candlesticks of the day. A support is defined at the lowest point and a resistance at the highest point.

If price crosses over the resistance and the 10-period moving average is increasing, a long position is taken.

A profit target of 1% is defined.

A protection stop is set at the level of the support, if price reaches this level, the position will be closed with a stop order.

The position is also closed at 5 PM local market time in order to not keep any position overnight. Access to intraday data is necessary to test this trading system.

```
DEFPARAM CumulateOrders = False
MM = Average[10](close)
MyTarget = 1
EndTime = 170000
IF INTRADAYBARINDEX = 2 THEN
   MyResistance = highest[2](high)
   MySupport = lowest[2](low)
ENDIF
REM Enter Long:
IF MM > MM[1] AND close CROSSES OVER MyResistance THEN
   BUY 1 SHARES AT MARKET
ENDIF
REM Exit Long:
IF time > EndTime THEN
   SELL AT MARKET
ENDIF
SELL AT MySupport STOP
SET TARGET %Profit MyTarget
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Smoothed stochastic trading system

This trading system is based on a smoothed stochastic applied to median price (indicator 1) and an exponential moving average of this value (indicator2). The trading system buys when indicator 1 is above indicator 2 or enters a short position when indicator 1 is below indicator 2.

A target (limit) is defined 1% above the entry price.

```
DEFPARAM CumulateOrders = False
REM Buy
Indicator1 = SmoothedStochastic[9,9](MedianPrice)
Indicator2 = ExponentialAverage[9](Indicator1)
// InitVariable
StopLimit = 1
REM Enter long conditions
c1 = (Indicator1 >= Indicator2)
IF c1 THEN
   BUY 1 SHARES AT MARKET
ENDIF
REM Enter short conditions
IF NOT c1 THEN
   SELLSHORT 1 SHARES AT MARKET
ENDIF
SET TARGET %PROFIT StopLimit
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Swing Trading, ADX and Moving Average

This backtest uses the ADX indicator and its position with regard to the level 30 since at least 5 days, with the goal of reducing false signals and minimizing risk. It must be executed on a daily timeframe.
The trading system has many conditions that limit the number of trading opportunities.

```
DEFPARAM CumulateOrders = False
MyADX12 = ADX[12]
ADXperiods = 5
MyMM20 = Average[20](Close)
IsLow30 = 0

IF lowest[ADXperiods + 1](MyADX12) < 30 THEN
   IsLow30 = 1
ENDIF

// ACHAT
// ADX 12 must be greater than 30 for at least 5 bars
Condition1 = NOT IsLow30
// If the 20-period moving average of the current period is between the high and low of the current period and the moving average of the previous period is between the high and low of the previous period
Condition2 = High > MyMM20 AND Low < MyMM20 AND High[1] < MyMM20[1] AND Low[1] < MyMM20[1]
// If the high of the current day is higher than the high of the previous day
Condition3 = Dhigh(0) > Dhigh(1)
IF Condition1 AND Condition2 AND Condition3 THEN
   BUY 1 SHARES AT MARKET
ENDIF

// SHORT
// ADX 12 is greater than 30 since at least 5 bars
Condition4 = NOT IsLow30
// If the 20-period moving average of the current period is between the high and low of the current period and the moving average of the previous period is between the high and low of the previous period
Condition5 = High > MyMM20 AND Low < MyMM20 AND High[1] > MyMM20[1] AND Low[1] > MyMM20[1]
// If the low of the current day is lower than the low of the previous day
Condition6 = Dlow(0) < Dlow(1)
IF Condition4 AND Condition5 AND Condition6 THEN
   SELLSHORT 1 SHARES AT MARKET
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Trading system with a position counter

Inverse Fisher transform applied to RSI

This trading system uses the “Inverse Fisher Transform RSI” to place buy or sell orders.

It enters a long position when the Inverse Fisher Transform RSI crosses over 50 and exits a long position when it crosses under 80.

It enters a short position when Inverse Fisher Transform RSI crosses under 50 and exits short when Inverse Fisher Transform RSI crosses over 20.

This trading system can be backtested on futures in 1-hour view or stocks in daily view.

```
REM Inverse fisher transform applied to RSI.
REM Parameters: n = number of bars for calculation of the RSI.
n = 10
Ind = RSI[n](Close)
x = 0.1 * (Ind - 50)
y = (EXP (2 * x) - 1) / (EXP (2 * x) + 1)
z = 50 * (y + 1)
myInverseFisherTransformsRSI = z

IF (myInverseFisherTransformsRSI CROSSES OVER 50) THEN
   BUY 1 SHARES AT MARKET
ENDIF

IF (myInverseFisherTransformsRSI CROSSES UNDER 80) THEN
   SELL AT MARKET
ENDIF

IF (myInverseFisherTransformsRSI CROSSES UNDER 50) THEN
   SELLSHORT 1 SHARES AT MARKET
ENDIF

IF (myInverseFisherTransformsRSI CROSSES OVER 20) THEN
   EXITSHORT AT MARKET
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Trading system with TRADEINDEX – find inside bar

The following example trading system is based on a frequently used price pattern called an “Inside Bar” and based on 2 candlestick forms:

- The first form occurs if the range of the 2nd candle preceding the current candle is greater than the range of the candle preceding the current candle. The candle preceding the current candle must be white (close > open). In this case, a long position is taken.
- The second form occurs if the range of the 2nd candle preceding the current candle is lower than the range of the candle preceding the current candle and the candle preceding the current one is black (close < open). In this case we take a short position.

All positions are systematically closed 3 bars after they are opened.

```
DEFPARAM CumulateOrders = False
Condition1 = (High[2] >= High[1] AND Low[2] <= Low[1])
Condition2 = (High[2] <= High[1] AND Low[2] <= Low[1])
Condition3 = (Close[1] > Open[1])
Condition4 = (Close[1] < Open[1])

IF (Condition1 AND Condition3) THEN
   BUY 1 Share AT MARKET
ENDIF

IF LONGONMARKET AND (BarIndex - TRADEINDEX) = 3 THEN
   SELL 1 share AT MARKET
ENDIF

IF (Condition2 AND Condition4) THEN
   SELLSHORT 1 share AT MARKET
ENDIF

IF SHORTONMARKET AND (BarIndex - TRADEINDEX) = 3 THEN
   EXITSHORT AT MARKET
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Money management strategies

A backtest's result can be improved by using money management strategies.

These strategies are sometimes formalized in “martingales”. They are aimed at optimizing the mathematical expectancy of a trading system. The expectancy is the average win or loss for each transaction if many transactions are done. This implies being able to estimate the probability of a transaction being winning and the probable amount of profit or loss.

In order to implement a martingale, it can be very useful to have stop loss, take profit and inactivity orders coded directly in your trading system, so that they are fully customizable, and to have sub-strategies allowing us to dynamically manage a position's size.

Protection stops and profit targets

For more information about protection stops, trailing stops and profit targets, see the dedicated sections in the manual above.

Inactivity stops

The following code allows you to use an inactivity stop in your trading system. Don't forget to define the conditions of your stop, here called InactivityStopLong and InactivityStopShort. In the following example, the stop is triggered after 10 bars.

```
ONCE Count = 10
REM Choice of the number of bars after which the position will be automatically closed
IF ONMARKET AND (BARINDEX - TRADEINDEX + 1) > Count THEN
   IF LONGONMARKET THEN
      SELL AT MARKET
   ENDIF

   IF SHORTONMARKET THEN
      EXITSHORT AT MARKET
   ENDIF
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

Cumulate orders – Adding to an existing position with use of a position counter

An example of cumulating orders to increase position size is given in the following section.

To enable cumulating orders, enter the command **"DEFPARAM CumulateOrders=True"** at the beginning of the program.

This trading system uses the first condition based on the RSI to take the initial position only. Additional shares are added each bar where open is greater than previous close up to a maximum of 3.

"Countofposition" is used in this code to limit the maximum position size to three.

```
DEFPARAM CumulateOrders = True

REM Buy 1 when RSI < 30 and there is no position already open
IF RSI[14](Close) < 30 AND NOTONMARKET THEN
   BUY 1 SHARES AT MARKET
ENDIF

REM If there is an open long position and open > previous close, each time we buy an
additional quantity up to a maximum of three.
IF LONGONMARKET AND COUNTOFPOSITION < 3 THEN AND Open > Close[1] THEN
   BUY 1 SHARES AT MARKET
ENDIF

REM When price crosses under a simple moving average, close the position
IF Close Crosses Under Average[14](Close) THEN
   SELL AT MARKET
ENDIF
```

With these tools, we can now look at martingales. Here are some of the most popular. These techniques can be added to any trading system.

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

The classic martingale

The classic martingale doubles the position size when it loses in order to make up for the loss if the next trade is a winner. The disadvantage of a trading system like this is that successive losses make it more and more difficult (or impossible) to double your position. Starting with 1000€ for example, if you lose 5 times in a row, you would need 1000 x 32 = 32000€ to continue with this trading system.

As a result, trading systems with the martingale may be more adapted to trading stocks than Futures or Forex because the initial capital required to trade may be much larger in these 2 types of markets.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE ExitIndex = -2
REM Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF Barindex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      OrderSize = OrderSize * 2
      REM Double OrderSize if the last position was a losing position.
   ELSIF PositionPerf(1) > 0 THEN
      OrderSize = 1
      REM Reset position size to 1 if the last trade was a winning trade.
   ENDIF
ENDIF
//*********************//
REM The position size must be determined depending on the variable OrderSize in the entire code.
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

The great martingale

The great martingale is similar to the classic martingale, except that in addition to doubling the position size after each loss, we add one additional unit.

This is more risky than the classic martingale in case of successive losses but it allows significantly increasing gains otherwise.

This code must be integrated with your own entry and exit conditions

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE ExitIndex = -2
REM Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF Barindex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      OrderSize = OrderSize * 2 + 1 // if the last trade was losing, double OrderSize and
      add 1.
   ELSIF PositionPerf(1) >= 0 THEN
      OrderSize = 1 // if the last trade was winning, set the OrderSize to 1.
   ENDIF
ENDIF
//*********************//
REM The position size must be determined depending on the variable OrderSize in the
entire code.
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

The Piquemouche

The Piquemouche is another variant of the classic martingale. In case of loss, we increase the position size by 1 if there are less than 3 consecutive losses. If there are more than 3 consecutive losses, we double the position. A gain resets the position size to 1 unit.

This trading system is less risky than the 2 previous ones because the position size is not exponentially increased until 3 successive losses are attained.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE BadTrades = 0
ONCE ExitIndex = -2
REM Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex
//***********Code to insert at the end of the trading system**********//
IF Barindex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      BadTrades = BadTrades + 1
      IF BadTrades < 3 THEN
         // If the last trade was losing and there are less than 3 successive losses,
         // add one to OrderSize.
         OrderSize = OrderSize + 1
      ELSIF BadTrades MOD 3 = 0 THEN
         // If the last position was losing and there are more than 3 consecutive losses,
         // double OrderSize.
         OrderSize = OrderSize * 2
      ENDIF
   ELSIF PositionPerf(1) >= 0 THEN
      // If the previous trade was winning, reset OrderSize to 1.
      OrderSize = 1
      BadTrades = 0
   ENDIF
ENDIF
//*********************//
REM The position size must be determined depending on the variable OrderSize in the
entire code.
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

The d'Alembert pyramid

This martingale was made famous by d'Alembert, a French 18th century mathematician. In case of loss, the position size is increased by 1 unit, in case of gain it is decreased by 1 unit.

This technique of position size management is relevant only if we suppose that successive gains reduce the probability of winning again and successive losses reduce the probability of losing again.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE ExitIndex = -2
REM Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF Barindex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      OrderSize = OrderSize + 1
   ELSIF PositionPerf(1) >= 0 THEN
      OrderSize = MAX(OrderSize -1, 1)
   ENDIF
ENDIF
//*********************//
REM The position size must be determined depending on the variable OrderSize in the
entire code.
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation tobuy or sell any financial instrument. Past performance is not indicative of futrure results. Any trading system may expose you to a risk of loss greater than your initial investment.

The contre d'Alembert

This is a reciprocal trading system of the D'Alembert Pyramid. We decrease the position size in case of a loss and increase the position size in case of a gain.

This technique is relevant if we believe that a past loss increases the probability of a future loss and a past gain increases the probability of a future gain.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE ExitIndex = -2
REM Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF Barindex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      OrderSize = MAX(OrderSize -1, 1)
   ELSIF PositionPerf(1) >= 0 THEN
      OrderSize = OrderSize + 1
   ENDIF
ENDIF

//*********************//
REM The position size must be determined depending on the variable OrderSize in the entire code.
```

Glossary

A

CODE	SYNTAX	FUNCTION
ABS	ABS(a)	Mathematical function "Absolute Value" of a
AccumDistr	AccumDistr(price)	Classical Accumulation/Distribution indicator
ADX	ADX[N]	Indicator Average Directional Index or "ADX" of n periods
ADXR	ADXR[N]	Indicator Average Directional Index Rate or "ADXR" of n periods
AND	a AND b	Logical AND Operator
AroonDown	AroonDown[P]	Aroon Down indicator of n periods
AroonUp	AroonUp[P]	Aroon Up indicator of n periods
ATAN	ATAN(a)	Mathematical function "Arctangent" of a
AS	RETURN Result AS "ResultName"	Instruction used to name a line or indicator displayed on chart. Used with "RETURN"
AT	AT (price)	Associates a command to a price
Average	Average[N](price)	Simple Moving Average of n periods
AverageTrueRange	AverageTrueRange[N](price)	"Average True Range" - True Range smoothed with the Wilder method

B

CODE	SYNTAX	FUNCTION
BarIndex	BarIndex	Number of bars since the beginning of data loaded (in a chart in the case of a ProBuilder indicator or for a trading system in the case of ProBacktestProBacktest or ProOrder)
BollingerBandWidth	BollingerBandWidth[N](price)	Bollinger Bandwidth indicator
BollingerDown	BollingerDown[N](price)	Lower Bollinger band
BollingerUp	BollingerUp[N](price)	Upper Bollinger band
BREAK	(FOR...DO...BREAK...NEXT) or (WHILE...DO...BREAK...WEND)	Instruction forcing the exit of FOR loop or WHILE loop
BUY	BUY x SHARES	Instruction to open a long position

C

CODE	SYNTAX	FUNCTION
CALL	myResult=CALL myFunction	Calls a user indicator to be used in the program you are coding
CASH	BUY x CASH	Amount of cash used in the position
CCI	CCI[N](price) or CCI[N]	Commodity Channel Index indicator
ChaikinOsc	ChaikinOsc[Ch1, Ch2](price)	Chaikin oscillator
Chandle	Chandle[N](price)	Chande Momentum Oscillator
ChandeKrollStopUp	ChandeKrollStopUp[Pp, Qq, X]	Chande and Kroll Protection Stop on long positions
ChandeKrollStopDown	ChandeKrollStopDown[Pp, Qq, X]	Chande and Kroll Protection Stop on short positions
Close	Close[N]	Closing price of the current bar or of the n-th last bar
COLOURED	RETURN x COLOURED(R,G,B)	Colors a curve with the color you defined using the RGB convention
COS	COS(a)	Cosine Function
COUNTOFLONGSHARES	COUNTOFLONGSHARES	Counts the number of open long shares or lots
COUNTOFPOSITION	COUNTOFPOSITION	Counts the number of open shares or lots
COUNTOFSHORTSHARES	COUNTOFSHORTSHARES	Counts the number of open short shares or lots
CONTRACT	BUY 1 CONTRACT	Designates the number of contracts to buy. Equivalent to 'SHARES'
CROSSES OVER	a CROSSES OVER b	Boolean Operator checking whether a curve has crossed over another one
CROSSES UNDER	a CROSSES UNDER b	Boolean Operator checking whether a curve has crossed under another one
cumsum	cumsum(price)	Sums a certain price on the whole data loaded
CumulateOrders	DEFPARAM CumulateOrders=true/false	When set to false, prohibits a code from reinforcing positions and setting multiple orders to enter the market in the same direction.
CustomClose	CustomClose[N]	Constant which is customizable in the settings window of the chart (default: Close)
Cycle	Cycle(price)	Cycle Indicator

D

CODE	SYNTAX	FUNCTION
Date	Date[N]	Reports the date of each bar loaded on the chart
Day	Day[N]	Reports the day of each bar loaded in the chart
Days	Days[N]	Counter of days since 1900
DayOfWeek	DayOfWeek[N]	Day of the week of each bar
DClose	DClose(N)	Close of the n-th day before the current one
DEFPARAM	DEFPARAM	Lets you define parameters like CumulateOrders
DEMA	DEMA[N](price)	Double Exponential Moving Average
DHigh	DHigh(N)	High of the n-th bar before the current bar
DI	DI[N](price)	Represents DI+ minus DI-
DIminus	DIminus[N](price)	Represents the DI- indicator
DIplus	DIplus[N](price)	Represents the DI+ indicator
DLow	DLow(N)	Low of the n-th day before the current one
DO	See FOR and WHILE	Optional instruction in FOR loop and WHILE loop to define the loop action
DOpen	DOpen(N)	Open of the n-th day before the current one
DOWNTO	See FOR	Instruction used in FOR loop to process the loop with a descending order
DPO	DPO[N](price)	Detrented Price Oscillator

E

CODE	SYNTAX	FUNCTION
EaseOfMovement	EaseOfMovement[I]	Ease of Movement indicator
ELSE	See IF/THEN/ELSE/ENDIF	Instruction used to call the second condition of If-conditional statements
ELSEIF	See IF/THEN/ELSIF/ELSE/ENDIF	Stands for Else If (to be used inside of conditional loop)
EMV	EMV[N]	Ease of Movement Value indicator
ENDIF	See IF/THEN/ELSE/ENDIF	Ending Instruction of IF-conditional statement
EndPointAverage	EndPointAverage[N](price)	End Point Moving Average of a
EXITSHORT	EXITSHORT x SHARES	Instruction to close a short position
EXP	EXP(a)	Mathematical Function "Exponential"
ExponentialAverage	ExponentialAverage[N](price)	Exponential Moving Average

F - G

CODE	SYNTAX	FUNCTION
FOR/TO/NEXT	FOR i=a TO b DO a NEXT	FOR loop (processes all the values with an ascending (TO) or a descending order (DOWNTO))
FLATAFTER	DefParam FlatAfter = HHMMSS	Closes positions, cancels pending orders and prevents placement of additional orders after the time of day specified (in hours, minutes and seconds) in the user's time zone
FLATBEFORE	Defparam FlatBefore = HHMMSS	Closes positions, cancels pending orders and prevents placement of additional orders before the time of day specified (in hours, minutes and seconds) in the user's time zone
ForceIndex	ForceIndex(price)	Force Index indicator (determines who controls the market (buyer or seller)
GRAPH	GRAPH myvariable AS "myvariable"	Instruction to display the historical values of ProBacktest variables on charts

H

CODE	SYNTAX	FUNCTION
High	High[N]	High of the current bar or of the n-th last bar
highest	highest[N](price)	Highest price over a number of bars to be defined
HistoricVolatility	HistoricVolatility[N](price)	Historic Volatility (or statistic volatility)
Hour	Hour[N]	The hour of the close of each bar loaded in the chart in the user's time zone

I - J - K

CODE	SYNTAX	FUNCTION
IF/THEN/ENDIF	IF a THEN b ENDIF	Group of conditional instructions without second instruction
IF/THEN/ELSE/ENDIF	IF a THEN b ELSE c ENDIF	Group of conditional instructions
IntradayBarIndex	IntradayBarIndex[N]	Counts how many bars are displayed in one day on the whole data loaded

L

CODE	SYNTAX	FUNCTION
LIMIT	BUY AT x LIMIT	Instruction introducing a limit order
LinearRegression	LinearRegression[N](price)	Linear Regression inidcator
LinearRegressionSlope	LinearRegressionSlope[N] (price)	Slope of the Linear Regression inidcator
LOG	LOG(a)	Mathematical Function "Neperian logarithm" of a
LONGONMARKET	LONGONMARKET	Indicates whether you have open long positions or not
Low	Low[N]	Low of the current bar or of the n-th last bar
lowest	lowest[N](price)	Lowest price over a number of bars to be defined
LOSS	Set STOP LOSS x	Set a stop loss x units from the average position price
%LOSS	SET STOP %LOSS x	Set a stop loss x% from the average position price
$LOSS	SET STOP $LOSS x	Set a stop loss of x €,$ (in the currency of the instrument)
LOT	BUY 1 LOT	Define the number of lots to buy or sell (equivalent to "SHARE")

M

CODE	SYNTAX	FUNCTION
MACD	MACD[S,L,Si](price)	Moving Average Convergence Divergence (MACD) in histogram
MACDline	MACDLine[S,L](price)	MACD line indicator
MARKET	BUY AT MARKET	Designates an order at market price
MassIndex	MassIndex[N]	Mass Index Indicator applied over N bars
MAX	MAX(a,b)	Mathematical Function "Maximum"
MedianPrice	MedianPrice	Average of the high and the low
MIN	MIN(a,b)	Mathematical Function "Minimum"
Minute	Minute	The minute of the close of each bar loaded in the chart in the user's time zone
MOD	a MOD b	Mathematical Function "remainder of the division"
Momentum	Momentum[l]	Momentum indicator (close – close of the n-th last bar)
MoneyFlow	MoneyFlow[N](price)	MoneyFlow indicator (result between -1 and 1)
MoneyFlowIndex	MoneyFlowIndex[N]	MoneyFlow Index indicator
Month	Month[N]	Represents the month of each bar loaded in the chart

N

CODE	SYNTAX	FUNCTION
NegativeVolumeIndex	NegativeVolumeIndex[N]	Negative Volume Index indicator
NEXT	See FOR/TO/NEXT	Ending Instruction of FOR loop
NextBarOpen	AT MARKET NextBarOpen	Designates an order to be executed on the open of the next bar
NOCASHUPDATE	DEFPARAM NOCASHUPDATE=true/false	Allows backtests to not update their initial capital with gains and losses (instruction for backtests only).
NOT	Not A	Logical Operator NOT

O

CODE	SYNTAX	FUNCTION
OBV	OBV(price)	On-Balance-Volume indicator
ONCE	ONCE VariableName = VariableValue	Introduces a definition statement which will be processed only once
ONMARKET	ONMARKET	Indicates whether or not a position is open
Open	Open[N]	Open of the current bar or of the n-th last bar
OpenDate	OpenDate	Dateof the open of the current bar in the format YYYYMMDD
OpenDay	OpenDay	Day of the open of the current bar
OpenDayOfWeek	OpenDayOfWeek	Day of week of the open of the current bar
OpenHour	OpenHour	Opening hour of the current bar in the user's time zone
OpenMinute	OpenMinute	Opening minute of the current bar in the user's time zone
OpenMonth	OpenMonth	Opening month of the current bar
OpenTime	OpenTime	Opening time of the current bar in the format HHMMSS in the user's time zone
OpenYear	OpenYear	Year of the open of the current bar
OR	a OR b	Logical Operator OR

P - Q

CODE	SYNTAX	FUNCTION
PIPVALUE	PipValue	Value in €/$ of one pip (or one point) in the currency of the instrument. PipValue=PointValue
PIPSIZE	PipSize	Size of a pip (or point), PipSize=PointSize
POINTVALUE	PointValue	Value in €/$ of one pip (or one point) in the currency of the instrument. PipValue=PointValue
POINTSIZE	PointSize	Size of a pip (or point), PipSize=PointSize
POSITIONPERF	PositionPerf(n)	Indicates the percent of gain or loss of the nth previous position
POSITIONPRICE	PositionPrice	Indicates the current average position price
PRELOADBARS	DEFPARAM PRELOADBARS = 200	Sets the maximum amount of bars preloaded for the calculation of indicators used in a trading system.
PriceOscillator	PriceOscillator[S,L](price)	Percentage Price oscillator
PositiveVolumeIndex	PriceVolumeIndex(price)	Positive Volume Index indicator
PVT	PVT(price)	Price Volume Trend indicator
QUIT	QUIT	Used to stop a trading system

R

CODE	SYNTAX	FUNCTION
R2	R2[N](price)	R-Squared indicator (error rate of the linear regression on price)
Range	Range[N]	calculates the Range (High minus Low)
REM	REM comment	Introduces a remark (not taken into account by the code)
Repulse	Repulse[N](price)	Repulse indicator (measure the buyers and sellers force for each candlestick)
RETURN	RETURN Result	Instruction returning the result
ROC	ROC[N](price)	Price Rate of Change indicator
RSI	RSI[N](price)	Relative Strength Index indicator
ROUND	ROUND(a)	Mathematical Function "Round a to the nearest whole number"
ROUNDEDUP	ROUNDEDUP	Round up quantities to buy or sell (used for stocks with the instruction to buy an amount in cash)
ROUNDEDDOWN	ROUNDEDDOWN	Round down quantities to buy or sell (used for stocks with the instruction to buy an amount in cash)

S

CODE	SYNTAX	FUNCTION
SAR	SAR[At,St,Lim]	Parabolic SAR indicator
SARatdmf	SARatdmf[At,St,Lim](price)	Smoothed Parabolic SAR indicator
SELL	SELL x SHARES	Instruction to close a long position
SELLSHORT	SELLSHORT x SHARES	Instruction to open a short position
SET	SET	Determines the type of order: either limit or stop
SHARES	BUY x SHARES	Designates the number of shares to buy or sell
SHORTONMARKET	SHORTONMARKET	Indicates whether there are open short positions or not
SIN	SIN(a)	Mathematical Function "Sine"
SGN	SGN(a)	Mathematical Function "Sign of" a (it is positive or negative)
SMI	SMI[N,SS,DS](price)	Stochastic Momentum Index indicator
SmoothedStochastic	SmoothedStochastic[N,K](price)	Smoothed Stochastic
SQUARE	SQUARE(a)	Mathematical Function "a Squared"
SQRT	SQRT(a)	Mathematical Function "Squared Root" of a
STD	STD[N](price)	Statistical Function "Standard Deviation"
STE	STE[N](price)	Statistical Function "Standard Error"
Stochastic	Stochastic[N,K](price)	%K Line of the Stochastic indicator
STOP	SET STOP LOSS	Lets you place a stop (see LOSS in the glossary)
summation	summation[N](price)	Sums a certain price over the N last candlesticks
Supertrend	Supertrend[STF,N]	Super Trend indicator

T

CODE	SYNTAX	FUNCTION
TAN	TAN(a)	Mathematical Function "Tangent" of a
TARGET	SET TARGET PROFIT x	Instruction to set a target order at price level x
TEMA	TEMA[N](price)	Triple Exponential Moving Average
THEN	See IF/THEN/ELSE/ENDIF	Instruction following the first condition of "IF"
TICKSIZE	TICKSIZE	Tickszie of the instrument (smallest possible variation of price)
Time	Time[N]	Current time (closing time of the current bar = time the candle is evaluated in automatic trading mode) in the user's time zone
TimeSeriesAverage	TimeSeriesAverage[N](price)	Temporal series moving average
TO	See FOR/TO/NEXT	Directional Instruction in the "FOR" loop
Today	Today[N]	Date of the bar n-periods before the current bar
TomorrowOpen	AT MARKET TomorrowOpen	Designates an order to be executed on the open of the next day
TotalPrice	TotalPrice[N]	(Close + Open + High + Low) / 4
TR	TR(price)	True Range indicator
TRADEINDEX	TRADEINDEX(n)	Index of the bar at which the nth last order was executed
TRADEPRICE	TRADEPRICE(n)	Price level at which the nth last order was executed
TRAILING	SET STOP TRAILING x	Set a trailing stop x units from average position price
%TRAILING	SET STOP %TRAILING x	Set a trailing stop x units from average position price
$TRAILING	SET STOP $TRAILING x	Set a trailing stop of x €,$ (in the currency of the instrument)
TriangularAverage	TriangularAverage[N](price)	Triangular Moving Average
TRIX	TRIX[N](price)	Triple Smoothed Exponential Moving Average
TypicalPrice	TypicalPrice[N]	Represents the Typical Price (Average of the High, Low and Close)

U

CODE	SYNTAX	FUNCTION
Undefined	a = Undefined	Sets a the value of a variable to undefined

V

CODE	SYNTAX	FUNCTION
Variation	Variation(price)	Difference between the close of the last bar and the close of the current bar in %
Volatility	Volatility[S, L]	Chaikin volatility
Volume	Volume[N]	Volume indicator
VolumeOscillator	VolumeOscillator[S,L]	Volume Oscillator
VolumeROC	VolumeROC[N]	Volume of the Price Rate Of Change

W

CODE	SYNTAX	FUNCTION
WeightedAverage	WeightedAverage[N](price)	Represents the Weighted Moving Average
WeightedClose	WeightedClose[N]	Average of (2 * Close), (1 * High) and (1 * Low)
WEND	See WHILE/DO/WEND	Ending Instruction of WHILE loop
WHILE/DO/WEND	WHILE (condition) DO (action) WEND	WHILE loop
WilderAverage	WilderAverage[N](price)	Represents Wilder Moving Average
Williams	Williams[N](close)	%R de Williams indicator
WilliamsAccumDistr	WilliamsAccumDistr(price)	Accumulation/Distribution of Williams Indicator

X

CODE	SYNTAX	FUNCTION
XOR	a XOR b	Logical Operator eXclusive OR

Y

CODE	SYNTAX	FUNCTION
Year	Year[N]	Year of the bar n periods before the current bar
Yesterday	Yesterday[N]	Donne le jour précédent au format YYYMMDD

Z

CODE	SYNTAX	FUNCTION
ZigZag	ZigZag[Zr](price)	Zig-Zag de la théorie des vagues d'Eliott
ZigZagPoint	ZigZagPoint[Zp](price)	Date of the day preceeding the bar n periods before the current bar

Other

CODE	FUNCTION	CODE	FUNCTION
+	Addition Operator	<	Strict Inferiority Operator
-	Substraction Operator	>	Strict Superiority Operator
*	Multiplication Operator	<=	Inferiority Operator
/	Division Operator	>=	Superiority Operator
=	Equality Operator	//	Introduces a commentary line
<>	Difference Operator		

ProRealTime SOFTWARE