

Introduction to MetaTrader 5 and Programming with MQL5

Create your 1st Investment Robot with
MQL5 step by step from ZERO.

RAFAEL F. V. C. SANTOS



MetaTrader 5
MQL5

Introduction to MetaTrader 5 and Programming with MQL5

Create your 1st Investment Robot with MQL5 step by step from ZERO.

All rights reserved. This e-book or any part thereof may not be reproduced or used in any way without the express written permission of the author or publisher, except for the use of short quotations in a review of the e-book.

First edition, 2018.

Author: Rafael F. V. C. Santos (rafaelfvcs@gmail.com)

Specialist in strategic risk management applied to the financial market. He works with the development of automated investment strategies (Robots of investment - Expert Advisor) using machine learning and spatial statistics. Graduated in Chemical Engineering from the Federal University of Pernambuco (UFPE) - Brazil. He holds a Masters and PhD in Civil Engineering (UFPE) in the areas of characterization, modeling and statistical simulation, applied to oil wells and reservoirs. He has several articles, with the theme of applied statistics, published in national and international magazines and congresses.

Copyright©2018 de Rafael F. V. C. Santos.

Summary

1. Introduction

- 1.1. Digital Age
- 1.2. Artificial Intelligence
- 1.3. High Frequency Trading - HFT
- 1.4. Computer Programming
- 1.5. What will we learn?
- 1.6. Why MetaTrader 5?
- 1.7. EA Strategy
- 1.8. Advantages of Using EA
- 1.9. E-book Summary

2. MetaTrader 5

- 2.1. Downloading and installing MetaTrader 5
- 2.2. Main elements of the MetaTrader 5
- 2.3. Navigator Field
- 2.4. Field Toolbox
- 2.5. Looking for Candles
- 2.6. Saving Templates
- 2.7. Drawing Objects
- 2.8. Transition between Graph Times
- 2.9. Adding Indicators
- 2.10. Data Window

3. MQL5 Community

4. MetaEditor MQL5

- 4.1. MetaEditor
- 4.2. Creating a new Project
- 4.3. OnInit()
- 4.4. OnDeinit()
- 4.5. OnTick()
- 4.6. Programming Accessories
 - 4.6.1. Comments
- 4.7. EA Properties

- [4.8. MetaTrader and MetaEditor File Types](#)
- [4.9. Adding Libraries](#)
- [5. Basic Programming Logic with MQL5](#)
 - [5.1. Types of Variables](#)
 - [5.2. Declaration of Variables](#)
 - [5.2.1. Integer Type](#)
 - [5.2.2. Type Double](#)
 - [5.2.3. Type String](#)
 - [5.2.4. Type bool](#)
 - [5.2.5. Type Datetime](#)
 - [5.3. Declaring Constants](#)
 - [5.4. Vector Variables: Arrays](#)
 - [5.5. For loop](#)
 - [5.6. Enum](#)
 - [5.7. Input Type Variables](#)
 - [5.8. Local and Global Variables](#)
 - [5.8.1. Local variables](#)
 - [5.8.2. Global Variables](#)
 - [5.9. Variables Predefined by MQL5](#)
 - [5.10. Math Operations](#)
 - [5.11. Logical and Conditional Relationships](#)
 - [5.12. Ternary Operator](#)
 - [5.13. Methods or Functions](#)
 - [5.14. Candles and Tick Variables](#)
 - [5.15. Functions Comment \(\) and Alert \(\)](#)
 - [5.15.1. Comment\(\)](#)
 - [5.15.2. Alert\(\)](#)
 - [5.16. Adding MQL5 Indicators](#)
- [6. Programming the EA](#)
 - [6.1. Strategy overview](#)
 - [6.1.1. Moving Averages](#)
 - [6.1.2. Relative Strength Index \(RSI\)](#)
 - [6.2. Creating EA](#)
 - [6.3. Declaration of Global Variables](#)

- [6.3.1. Variables for the User](#)
- [6.3.2. Global Variables](#)
- [6.4. OnInit \(\) Function](#)
- [6.5. Function OnDeinit\(\)](#)
- [6.6. Expert Advisor Functions](#)
 - [6.6.1. Error Handling](#)
 - [6.7. Function: OnTick\(\)](#)
- [7. Backtests](#)
 - [7.1. Backtest in MetaTrader 5](#)
 - [7.2. Analyzing the Backtest](#)
 - [7.3. Performance Charts](#)

Chapter 1

L. Introduction

.1. Digital Age

We are living in the digital age. Nowadays, information and knowledge travel through the optical fibers. Everything is in constant development and accelerated growth. This evolutionary dynamic is unprecedented in the history of mankind.

Devices and objects are becoming increasingly intelligent and independent. Nowadays, it is possible to find, for sale, smart phones and watches that recharge the battery with even the temperature of the human body. Research shows that by 2025, at least half of the US fleet will be traveling completely autonomously. That's right, the profession of driver, taxi driver and many others, are counting their days. And for the financial market this is no different. Professional traders are being replaced by algorithms.

These are just small examples of the avalanche of modifications and benefits brought in from the world shouting for change. We are changing the world and the world forcing us to change into a virtuous circle of prosperity.

.2. Artificial Intelligence

Embedded systems are gradually dominating our daily lives. People and things are becoming more and more connected to the internet. A new intelligence is created from this new cosmos. We are opening the doors to **Artificial Intelligence (AI)**. Practically everyone benefits (or will benefit) in some way from the wonders brought by AI.

Many say that this new generation may bring us problems, but we must be optimistic about the uncertainties, as we are on the way to an inevitable and non-return path.

Several companies and investment funds are currently working heavily on the use of artificial intelligence to make their investment decisions. From AI, we can filter and group information and knowledge hidden in apparently meaningless numeric data and connection.

In this race, it is advantageous to dominate subjects and knowledge related to mathematics, statistics and computer programming.

.3. High Frequency Trading - HFT

Investors and speculators in the financial market know that the main enemy to make money in this world is precisely the emotional factor. Human beings have desires, wants, greed, fears and fears that make the activity decide the best time to buy and / or sell a real challenge.

When we are working with Investment Robots, we are discussing a strategy that has been previously defined and structured so that it can be executed in a timely manner. Fortunately, robots do not get tired, stress or have any feelings that allow misunderstandings and / or biases when buying or selling.

Optical fibers allowed for a revolution in communication. In the investment world, for example, every thousandth of a second can cost millions of dollars. So-called High Frequency Traders (HFTs) operating on the world's top stock exchanges makes decisions at infinitesimal speeds and thus are threatening many professional brokers and traders.

The HFTs' decision to buy and / or sell is taken in milliseconds, up to ten to fifty times faster than a simple wink of the human eye (lasting approximately 300 milliseconds). It is practically unfeasible and unfair the operational comparison of any computational system of this category with a normal human being.

Recent data (year 2018) show that more than 50% of US financial operations are carried out by investment robots. And the results show that these robots can increase financial returns by up to 15%. There are some stock exchanges where more than 90% of operations are carried out by investment robots.

.4. Computer Programming

Most people have the following objections when it comes to programming: it's very difficult, needs a lot of practice, has to know a lot of math, I've never been good with numbers. However, many of these objections are nothing more than a misunderstanding of this programming activity, which is as trivial as telling a brief story with beginning middle and end.

The interesting thing is that anyone able to write the steps of running a cake recipe on a sheet of paper is able to create a computer program. To program is to communicate with the outside, with the other, in order to conduct actions. For our case, this other is the machine.

We have a structure of activities to be followed and we need, for a good realization of it, an adequate sequence of order of execution. Programming is to master a language the machine understands and know how to position our deliberative needs in time and space, so that they are executed correctly.

Therefore, it is up to every curious investor and with the patience to learn something new to look at in the world of programming. This is because scheduling computers is a much simpler activity than many imagine.

.5. What will we learn?

In this e-book we will learn how to create, step by step, an automated investment strategy. We will create an Investment Robot (Expert Advisor - EA) from scratch using the MetaTrader 5 and MQL5 programming language.

Thus, having a program written in a high-level language (MQL5) (those closest to the understanding of humans, low-level languages are closer to the understanding of machines) we will be able to leave the arduous and exhausting work of monitoring purchase and sale of a financial asset on the stock exchange for the robot.

We will spend a few hours to program a robot known as **Expert Advisor** and spend the rest of the time monitoring the execution of the strategy in a much quieter and safer way. That leaves more time for a contemplative and pleasurable life.

.6. Why MetaTrader 5?

MetaTrader 5 is a powerful platform for visualizing, operating and scheduling investment strategies for various types of financial markets.

It was created in the year 2005 and continues to be improved and distributed by MetaQuotes (<https://www.metaquotes.net>). We can use MetaTrader 5 to operate in the stock markets, futures, options, currency pairs (FOREX) among others.

MetaTrader 5 was chosen to build our first EA, as it is a 'free' tool, quite powerful and complete from the point of view of the main functionalities that a trader platform should have.

Inside MetaTrader 5 we have the option to create scripts, indicators and, of course, investment robots using the MQL5 programming language. This language is very similar to C ++, Java and C #. So anyone who already knows any of them will feel extremely comfortable learning MQL5.

.7. EA Strategy

We will develop from scratch a strategy of crossing two moving averages (one fast and one slow) with the possibility of an input filter in operations with the rather popular indicator **RSI** (relative strength index).

Our robot will be multi-strategy. It will be able to work, using as input triggers in the operations, only the crossing of the moving averages, only the RSI or both (crossing of averages plus RSI).

We will also learn how to make backtests of the created robot. Thus we will be able to test various setups on historical data of various financial assets. The book is full of guidelines, with screenshots, using the key parts of **MetaTrader 5** and **MetaEditor**.

We will learn the programming syntax in MetaEditor. We have several features when programming with MQL5. It already presents a very significant set of libraries to work with graphical manipulation, mathematical operations, statistics, with price structures and control of the sending of trading orders.

.8. Advantages of Using EA

Let's go to the advantages behind using investment robots. They are many, but we go to the main ones:

- Eliminates all emotional side;
- Test ready strategies with backtests;
- Simulate adverse market situations;
- EA are free from fatigue, stress, lose attention, or fail to execute the strategy;
- We can optimize the strategy from a Robot;
- EA are infinitely disciplined and speedy compared to us;
- Statistical validations can be made. EA statistically validated are winners and consistent in the medium and long term.
- EA has a single mindset that is to execute the scheduled strategy;
- EA operate from the first second of market opening to the last second before closing, without truce and misconduct.

.9. E-book Summary

So to achieve the goal of creating our first Expert Advisor, this e-book organizes and simplifies in a didactic way the necessary steps with the knowledge and usage information of MetaTrader 5 and its MQL5 programming language.

Here's a summary of what **you will learn** from reading this e-book:

- Learn how to install MetaTrader 5;
- Know the main benefits of the MetaTrader online portal;
- Understand the main features and differences between MetaTrader and MQL5:
- Learn to add indicators and EA;
- Know MetaEditor and some of the main shortcuts to facilitate programming;
- Learn the programming syntax of MQL5;
- Understand the main functions of the MQL5 development libraries.
- Create a strategy using trend indicators (moving averages) and consolidated market indicators (Relative Strength Index - RSI);
- Program step-by-step a Multistrategy EA;
- Make backtests of the created EA;
- Know the backtest statistics;
- And much more.

Now we are moving towards the future of investments. Congratulations on your decision to enter this grand and fascinating world of automated investments.

Thank you for your confidence in purchasing this material. The future is here and we will build it together.

Everyone a great read!

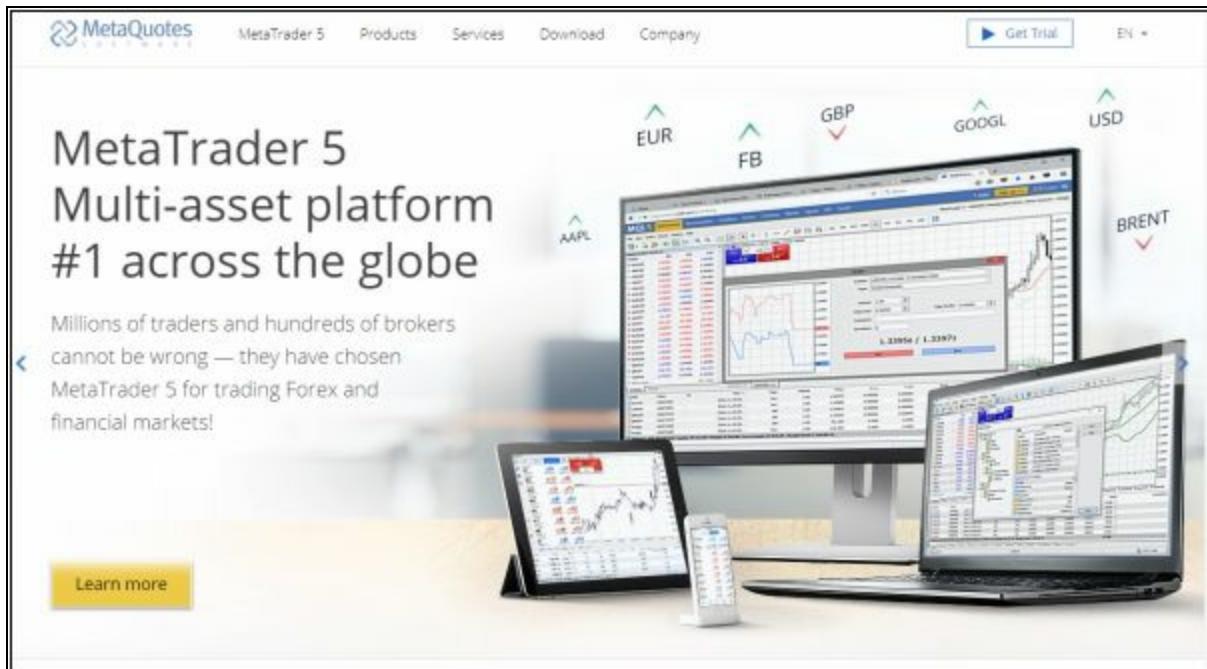
Important note^[1].

Chapter 2

!. MetaTrader 5

MetaTrader is a platform developed to assist in the visualization of prices of financial assets and execution of buy and sell orders on the stock exchange. It is also possible to create scripts (algorithms for specific actions), indicators and investment robots.

It is a free tool developed and distributed by MetaQuotes Softwares (<https://www.metaquotes.net/>). Initially, it was developed thinking about the Forex market. However, several brokerages have been allowing for the use in the stock market, options and futures.



We need a company (investment broker) to provide routing pricing data for the platform. Many investment brokers offer this service for free. Some even offer a demo account where we can simulate buy and sell operations in the financial market with dummy money. Also, in the **demo account**, we can put our robots to test in real time before using them in a real account.

MetaTrader 4 was **created in 2005** and since then has been undergoing major transformation of improvements in its system and security features. It is currently in its second major release called MetaTrader 5 which was released in 2010.

We can use MetaTrader on both **desktop** and **smartphones**. It is available to work on the Web, on Mac, Linux, Windows and Android systems. The leading Internet browsers on the market support MetaTrader 5 online.

MetaTrader 5 Web Trading

Trade on financial markets from any browser



The MetaTrader 5 Web platform allows you to start trading on the Forex, exchange and futures markets from any browser and operating system. With the MetaTrader 5 Web Platform, all you need to have is an Internet connection.

MetaTrader 5 is not only a tool for visualizing and routing orders (buy/sell) of financial assets, but also an excellent platform for developing investment strategies from scripts, indicators and robots (Expert Advisors). For this it counts on the environment of development for the language MQL5 (MetaQuotes Language) that presents/displays great similarities with C++, Java and C#. Therefore, those who have had some knowledge in any of these programming languages will feel comfortable with the MQL5 syntax.

In comparison to MetaTrader 4, MetaTrader 5 allows developing strategies using the object-oriented programming paradigm (OOP).

1.1. Downloading and installing MetaTrader 5

In the world several brokerages offer free use of MetaTrader 5. All we need is that we are a client of the investment broker. We need, after downloading the platform, to request a release of a demo and / or real account so that we can have access to the routing of the data of the financial assets.

The direct download of the platform can be done by the site: <https://www.metaquotes.net/en/metatrader5/>.

As said before, several brokers around the world offer free routing, but many for the Forex market. Some Forex brokers only need a valid phone number to release a demo account. In this e-book we will use the systems with the prices of financial assets present in the São Paulo (Brazil) BM&FBovespa stock exchange.

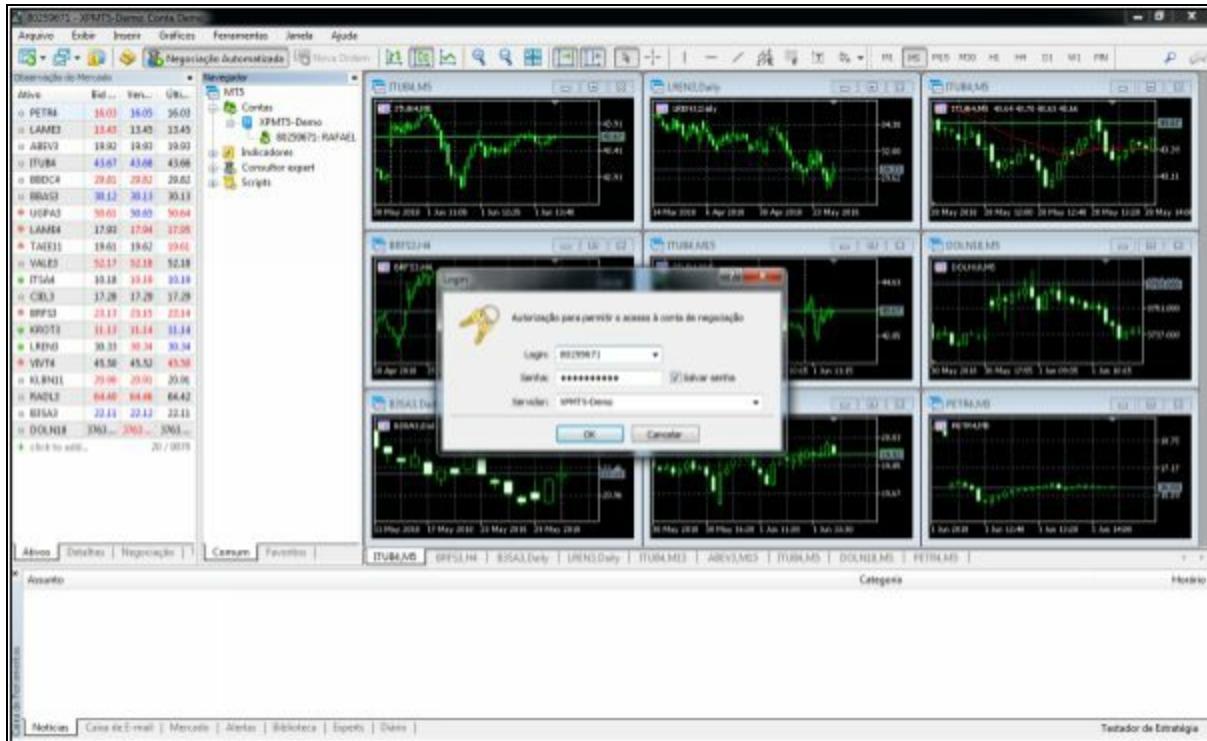
However, to follow the studies in the book, you do not necessarily need to have an account with a Brazilian investment broker.

The installation process is very simple and straightforward. After two clicks on the file an installation wizard will guide us with the necessary steps.



Just go ahead until the end of the installation. At the end you will be asked for an **account** with **login** and **password**. It is necessary for us to use the pricing data for financial assets. In this e-book we will use a demo account from *XP Investimentos*.

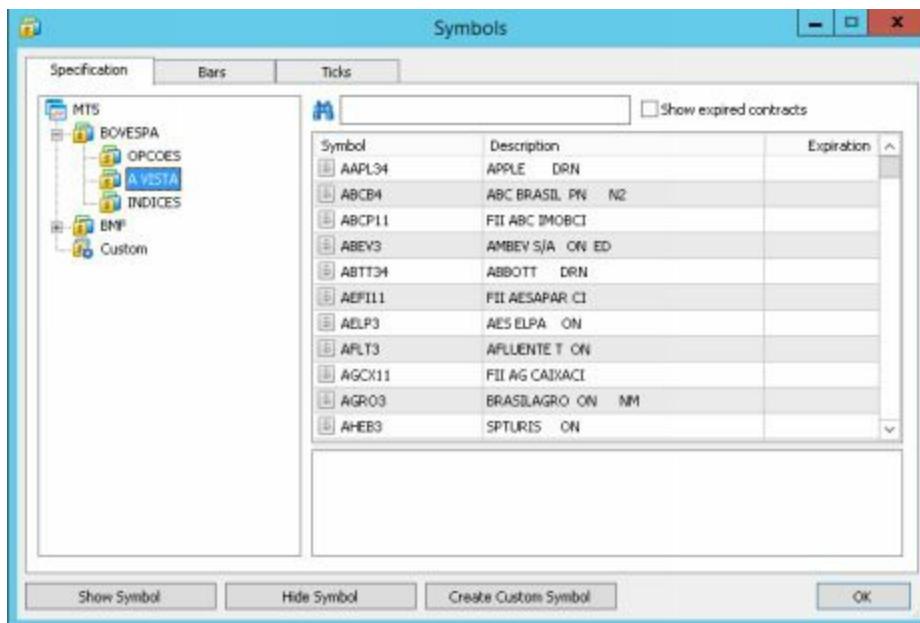
See below the open program requesting the **login**, **password** and type of server (if real or demo).



1.2. Main elements of the MetaTrader 5

One of our main interests is to **visualize asset prices** from candles (candlesticks). So, let's get to know the main visualization features of the MetaTrader 5 platform.

In the **menu:** View -> Symbols, we have the option to choose the financial assets that we wish to work.



We can choose between Bovespa and BMF Assets (continuous series). In the assets of Bovespa we have the OPTIONS (calls and puts), the shares in A VISTA and the INDICES (futures market) traded in the Brazilian market.

It is important to note that each brokerage will provide distinct data routing. Some with more data and some with less. For example, if we were with some brokerage that works with the Forex market, the options that would appear in this window of Assets would be completely different.

The addition of a symbol that represents a company, which is being consulted in the prices, can be executed in the **Market Watch**.



In this Market Observation field we can add the assets related to the companies traded on the stock exchange in which the brokerage firm is released to operate.

Still within the **Market Watch** field we have the **Details** tab that offers additional information regarding the financial asset being evaluated. Below, for example, we can see the information on this tab for *Petrobras* shares. In the **Trading** tab we have several options where we can make purchases and sales with just one click.

Market Watch: 10:23:42

Symbol	Bid	Ask	Last
PETR4	14.68	14.69	14.69
VALE3	48.02	48.04	48.02
ABEV3	18.33	18.35	18.33
IBOV			69905
click to add... 4 / 9834			

Market Watch: 10:24:10

IBOV, IBOVESPA	
Bid	
Ask	
Last	69965
Last High	70757
Last Low	69574
Volume	1.845271M
Deals	1867078
Deals Volume	51.084994M
Open Price	70757
Average Wei...	69994

Market Watch: 10:24:35

PETR4	
SELL	100.00
14 71	14 72
LOW	HIGH
14.62	14.77
VALE3 10:24:34	
SELL	100.00
48 13	48 16
LOW	HIGH
47.86	48.62
ABEV3 10:24:34	
SELL	100.00
18 30	18 31
LOW	HIGH
18.15	18.37

Symbols Details Trading

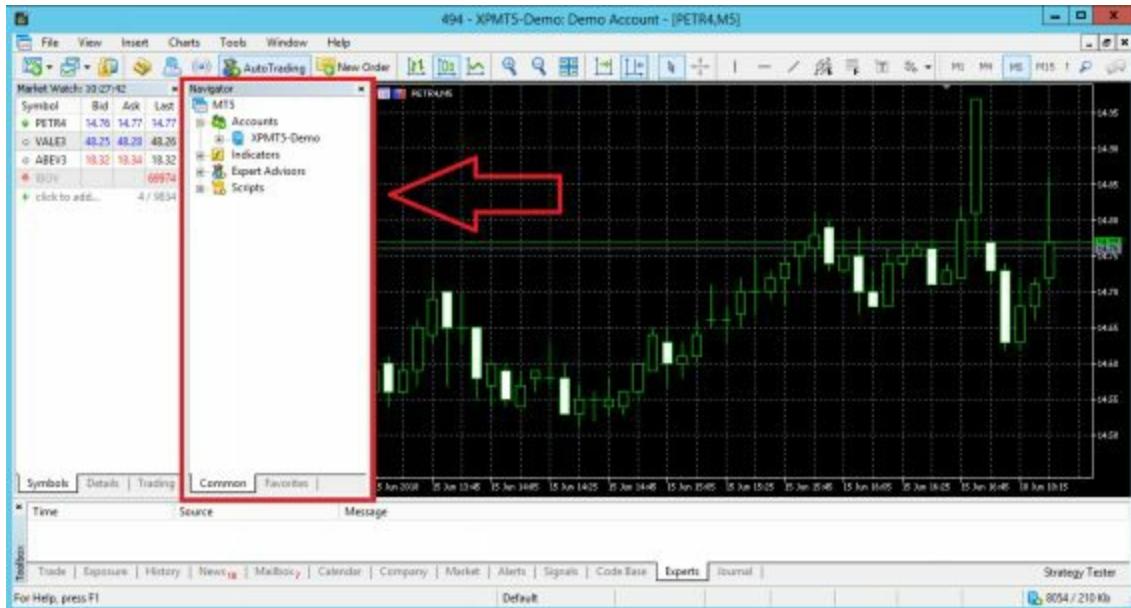
Symbols Details Trading

Symbols Details Trading

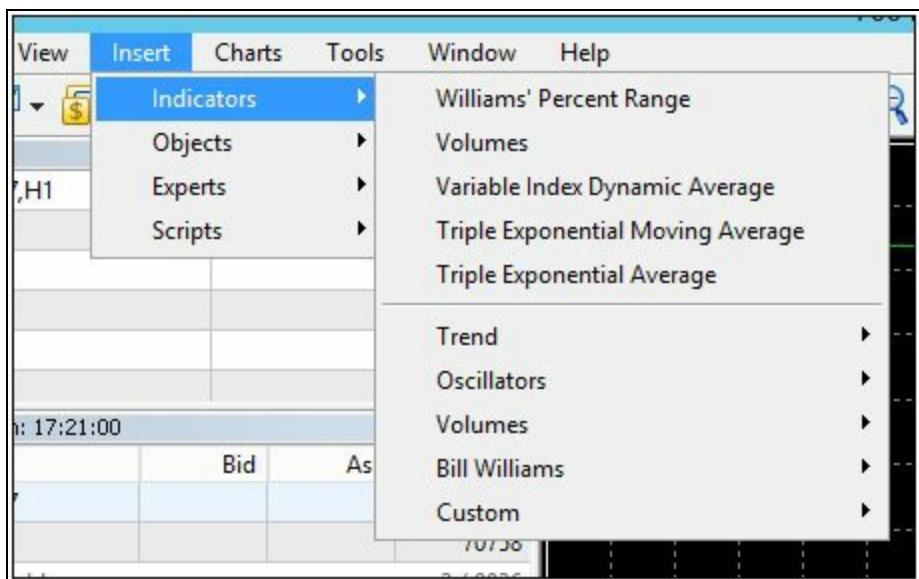
1.3. Navigator Field

The field **Navigator** is where it contains the information regarding the type of account of the user if demo or real, as well as the indicators and robots of investments available by the platform and those that were developed by ourselves.

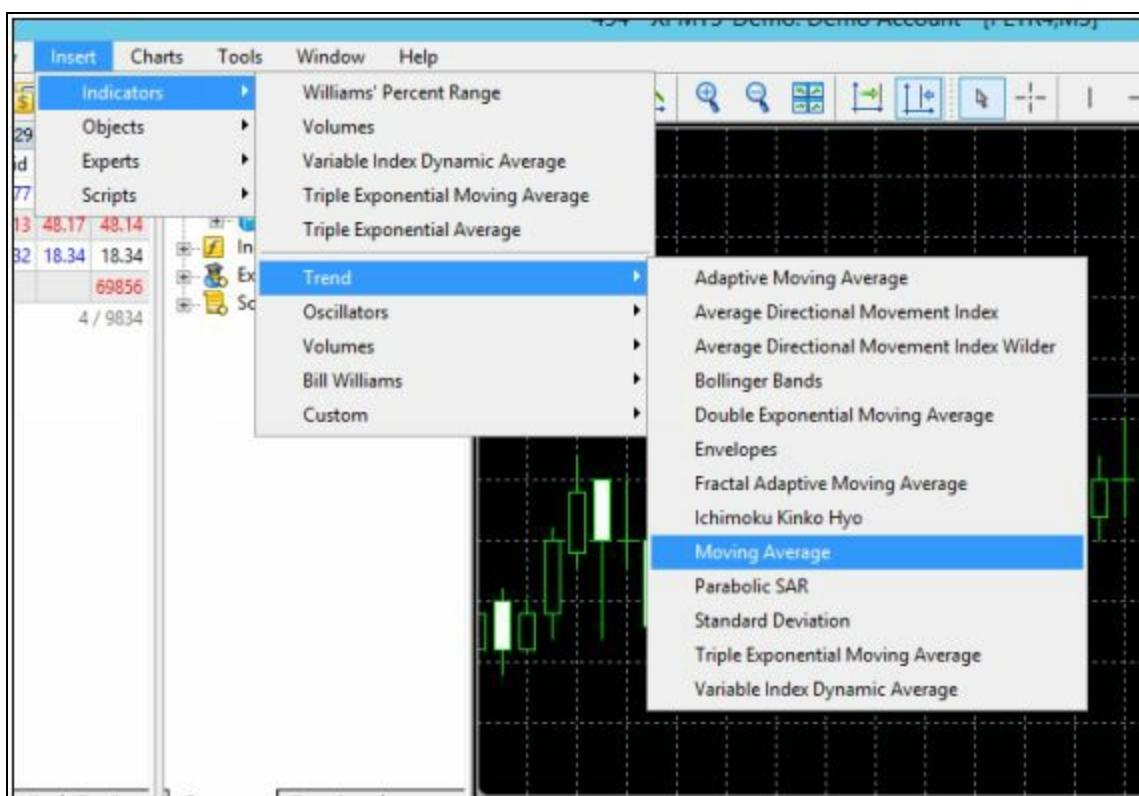
In this field, we can choose the type of indicator, script or EA and drag it directly to the financial asset chart of interest.



Another way to add indicators or EA is from the **menu Insert -> Indicators**, **Insert -> Expert (Robots)**.



Let's add an indicator from the menu:



In the figure above, we can see that the indicators are separated by trend indicators, oscillators, volumes, Bill Williams and Custom. Trend indicators, as the name implies, are more apt to warn when the market is within some well-defined directional behavior. On the other hand, the Oscillator indicators

are more adequate to warn of opportunities within a lateralized market, a market in consolidation.

Bill Williams^[2] is a very popular trader who has developed several important and widely used indicators. Therefore, a specific section was created for the indicators created by it.

It is interesting the possibility that we have to use indicators and native EA of MetaTrader itself and indicators and EA created by us or even offered in the **Market** tab of the **field Toolbox**.

4. Field Toolbox

In the **Toolbox** field we can find several important informations regarding the sale of the main indicators and investment robots developed for MetaTrader platform. We can also download those provided for free (tab **Libraries**). We can view the main news (tab **News**) in real time of trading as market in auction and output of news relevant to the market.



In the **Experts** tab, we can view the main information logs regarding the operation of our robots. In this place, we are informed about the sending and execution of orders, problems with variables, errors of execution and operations.

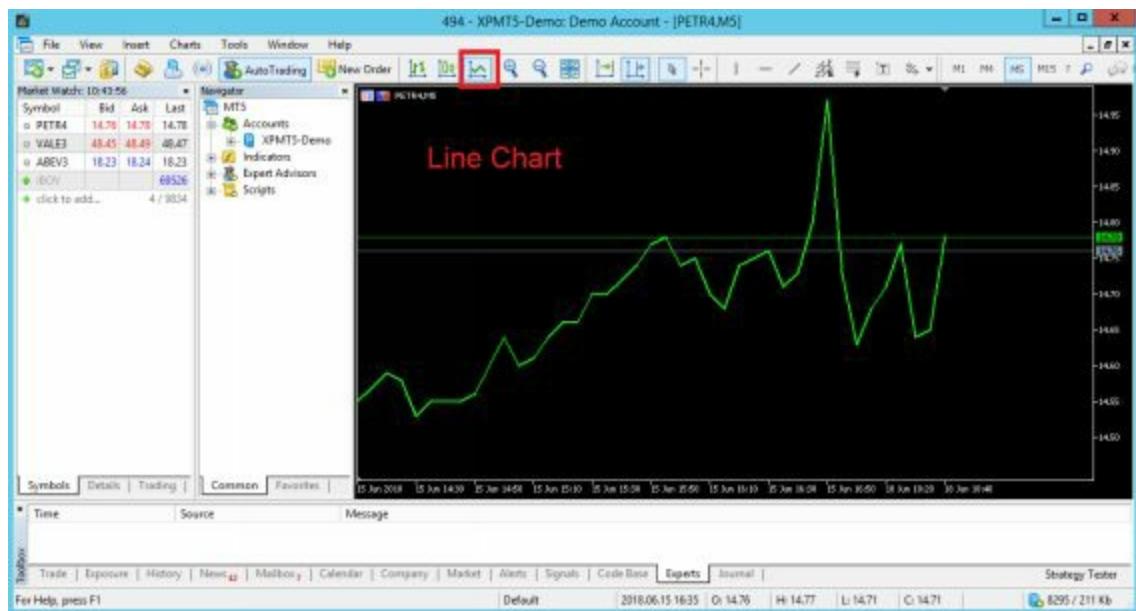
Now that we already know, in general, the main fields of MetaTrader 5, we go to the main graphic field, which is the window of visualization of the price candles.

1.5. Looking for Candles

The **MetaTrader 5** platform allows us to customize the way we visualize prices in different ways. For example, we can change the background of the screen, the colors of the contours and fill of the high/low/doji candles, the separation lines of periods, grids, among several other properties.

We can switch between three types of price visualization of financial assets. These visual types are given through **Bar Chart**, **Candlestick** or **Line Chart**.

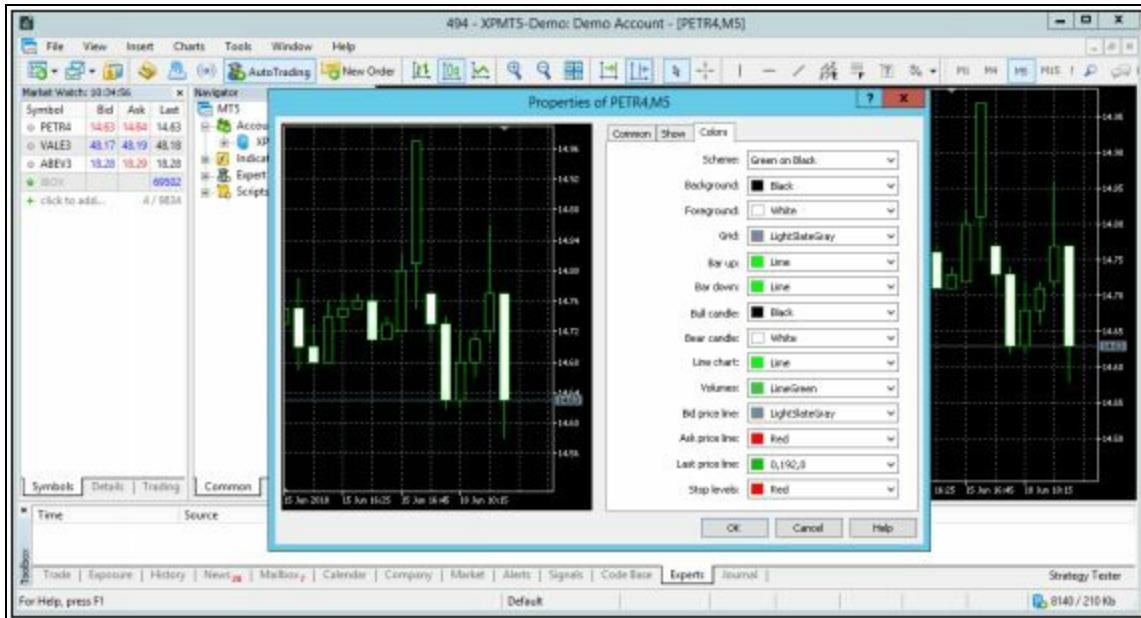




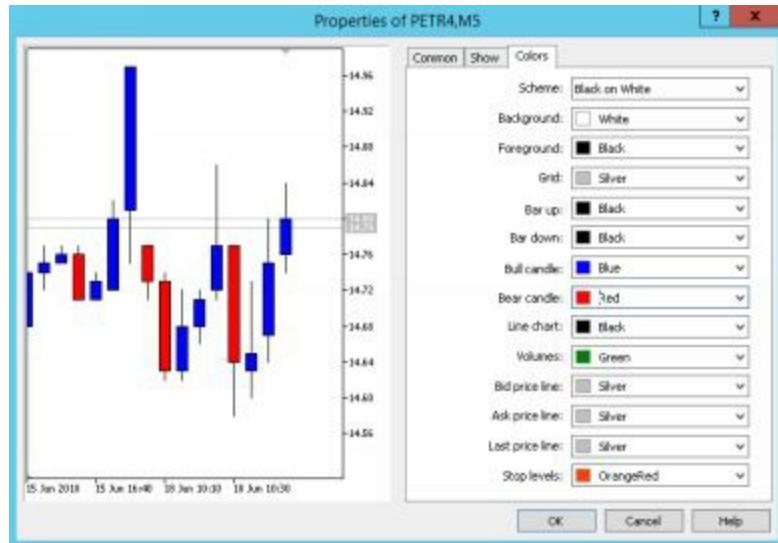
1.6. Saving Templates

It is important to note that each time a new chart is requested to be opened in MetaTrader it will appear on the screen with the default settings. However, there is a way to save our screen preview customization from creating a **template**.

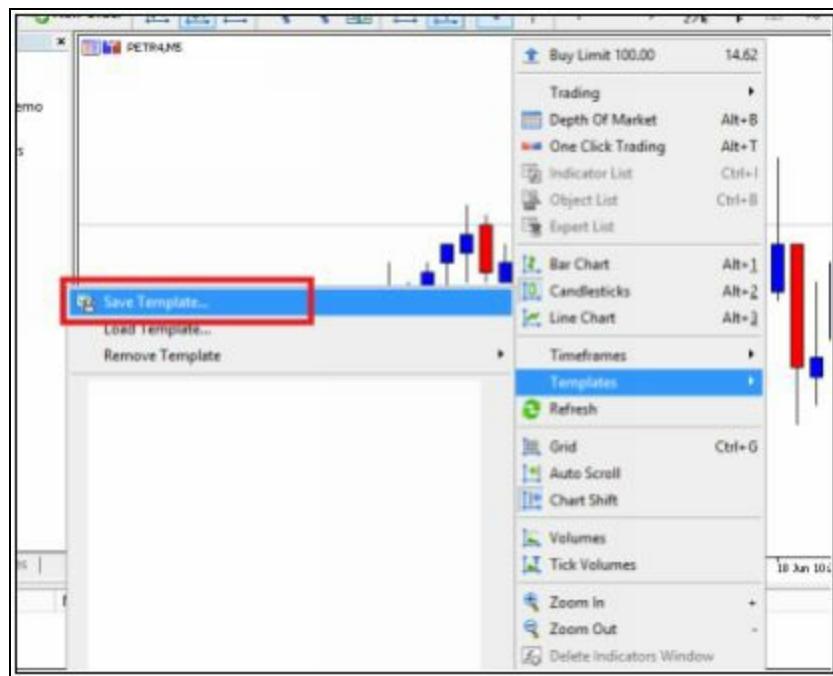
For example, let's set up a template. For this we must make our customizations. With a right click on the graphic field we can choose **Properties** and make the appropriate color settings to our liking.



Let's consider the following changes:



Then right click on the graphic screen and go to **Template -> Save Templates**. We chose a name and that's it.

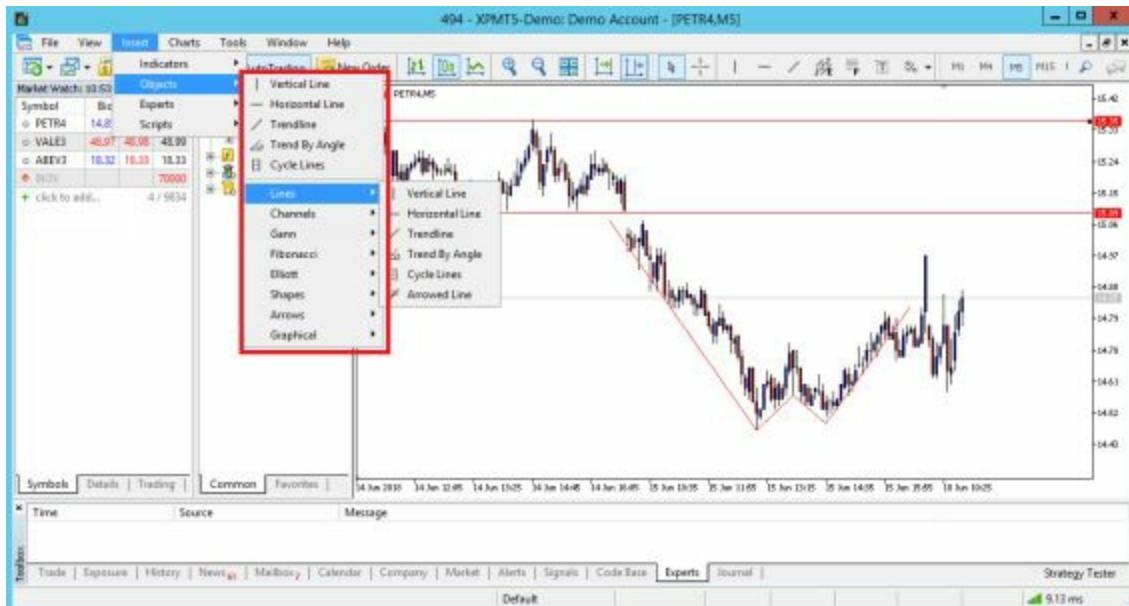


So, just load the saved template, to a new graphic, so that the settings are immediately applied. That way, we have a lot of flexibility to be able to configure visualization templates for setups of different strategies.

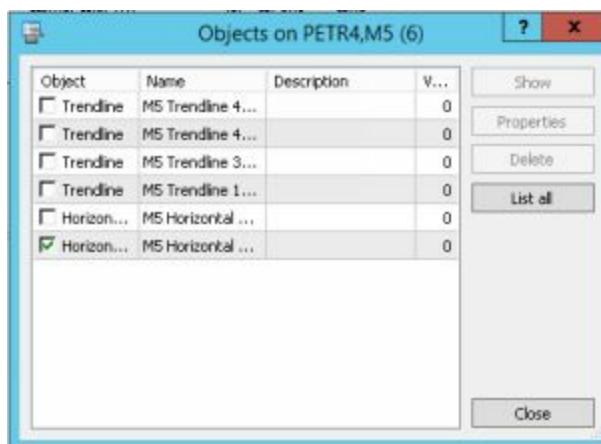
1.7. Drawing Objects

You can draw several objects in price charts. We can add channels, support lines and resistance, Fibonacci, up/down arrows etc.

In this way, it is possible to make a visual technical analysis very complete with the main graphic objects present in the platform.



Right-click on the graphic you can see the list of objects and make different types of edits like modify subtitles, colors etc.



It is worth spending some time familiarizing yourself with the use of graphics and objects.

.8. Transition between Graph Times

The transition between the graphical scales of candle formation observation is quite simple and fast. Just right click on top of the chart, go to **Timeframes** and choose the time of candles.



We can also change the candle time from the shortcut bar, as shown below:



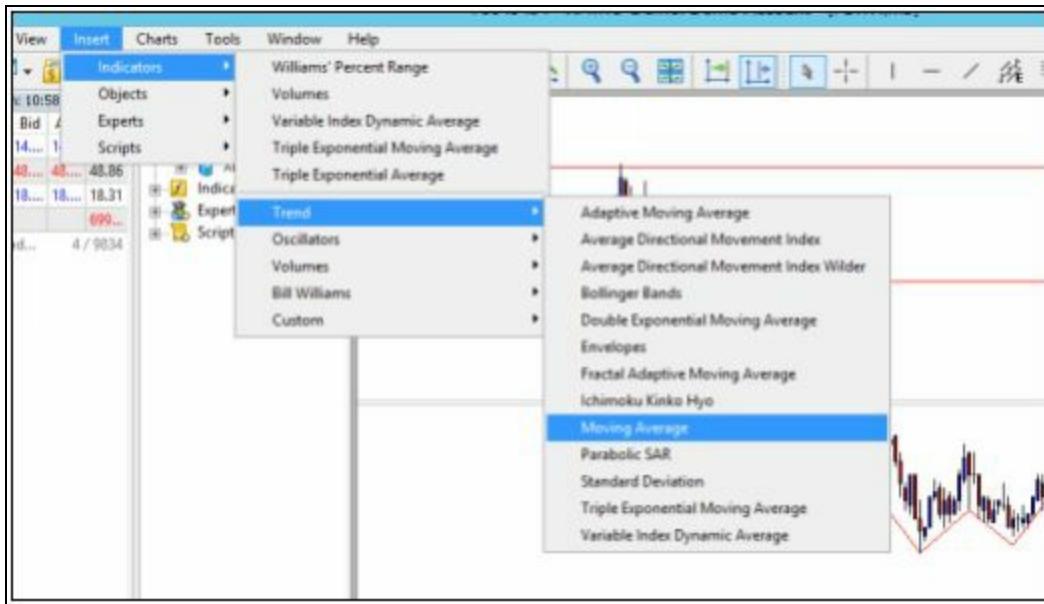
1.9. Adding Indicators

The indicators, for many traders, are the main guiding factors for decision making. These are the indicators that often alert us to enter and exit an operation in timely situations. And when you talk about indicators, MetaTrader does not leave you wanting.

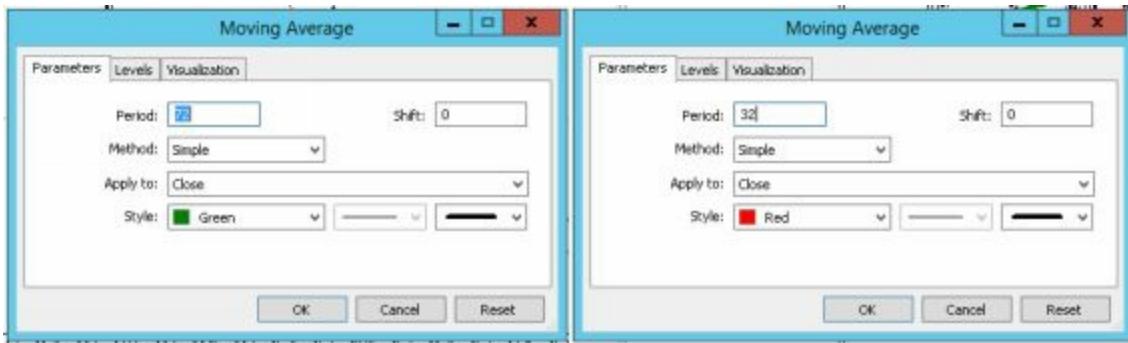
Currently, the platform has more than 2,000 indicators available for free download. This download can be done either by the site itself or by the platform in the **Toolbox** field on the **Code Base** tab.

Let's, for example, add moving averages to the chart. We can do this either through the menu **Insert** -> **Indicators** or directly clicking in the field **Navigator** -> **indicators**, click and hold the chosen indicator and drag to the graphic window.

As an example, let's insert a Moving Average indicator:



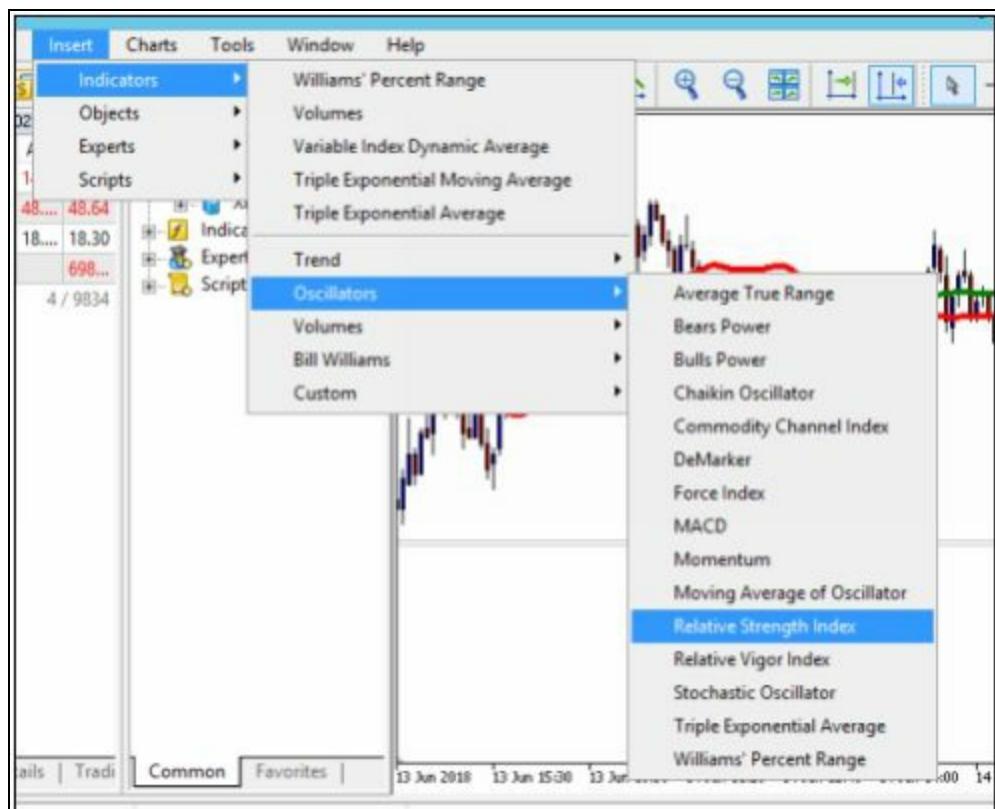
Now let's add two moving averages. Our averages should have periods of 72 and 32 with zero **shift**, exponential methods and applied to closing prices:



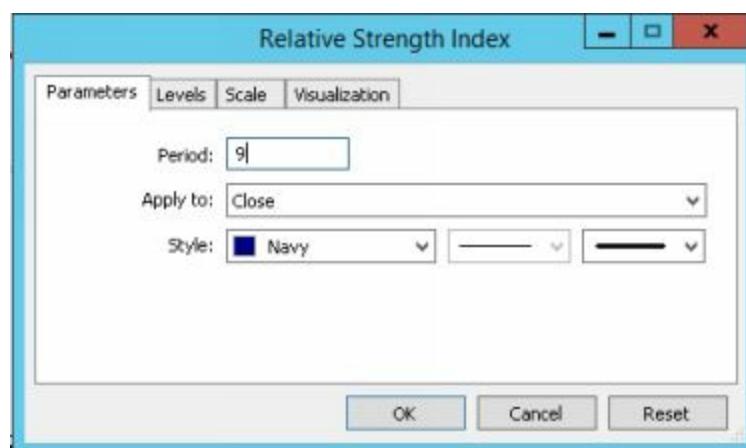
After clicking on ‘ok’, we have the immediate addition to the selected asset graph.



Let's now add the Relative Strength Index (RSI). This indicator is interesting as it is drawn in a separate window from the main candle price window (see figures below).



Let's leave a period of 9 applied to the closing price:



The result is:



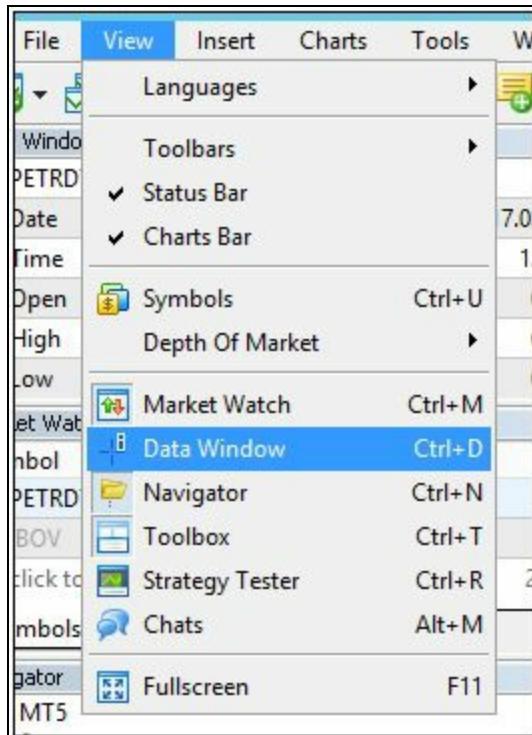
The RSI is an indicator used to warn about **overbought** and/or **oversold**. Basically, we have two reference levels that we must choose for the activation trigger of the overbought or oversold. We can choose the levels of 30 (oversold) and 70 (overbought).

For example, RSI values above 70 are indicating that the price is overbought, that is, at any moment in a consolidated market, we can begin to see a drop in prices. On the other hand, if the value of the IFR is below 30 we can say that the price is oversold and a sudden high in prices is eminent.

These two indicators were presented, since they will be used to create our investment robot. Going forward, we will use three strategies: one with the crossing of moving averages, another with only the RSI and one with both indicators.

.10. Data Window

A very important field, which greatly facilitates the visualization of prices and values of indicators, is the **Data Window**. To open it we can: go to the **View** menu -> **Data Window** or use the keyboard shortcut **Ctrl + D**.



This action will open, for us, a new field: **Data Window**. Now we can position the mouse cursor over any candle of the graph to obtain detailed information of prices and indicators in the place chosen.

See the example below, where we place the mouse on the highlighted candle in green and the Data Window gives us the detailed information:



Chapter 3

3. MQL5 Community

As discussed earlier, in Chapter 2, MQL5 is a programming language offered within MetaTrader 5. Programmers often share codes and questions around specific problems in a programming language.

It is very important to participate in communities of developers, because with this our learning and the motivational side are enhanced. The sharing of knowledge and doubts generates sustainability in our development and growth as a programmer and user of the language.

It is always good to have a place to exchange knowledge, ask questions and discuss project ideas. The more widely used and documented a programming language, the better the learning curve for first-time sailors. Therefore, a common environment where all are destined to the same end is quite valid.

In this sense, for the MQL5 language we are well covered, because we are facing a mature tool, with a lot of developers and a very active and generous community that offers articles and doubts with agility in the forums.

We can access the community from the site: <https://www.mql5.com/en>.

After reading this e-book and/or even now, the reader is invited to make an account and start adding their knowledge and ask questions from the portal.

MQL5 WebTerminal Documentation Calendar CodeBase Articles Freelance Market Signals VPS Forum [Login](#) [Sign up >>](#) English [Q](#)

Download MetaTrader 5

A PLATAFORMA #1 AGORA EM MINAS GERAIS! [BAIXE O METATRADER 5 >>](#)

[Log in](#) or [Register](#) to gain access to the benefits from all our services.

Traders Forum

Ask questions on technical analysis, discuss trading systems and improve your MQL5 programming skills to develop your own trading strategies.

Communicate and share your experience with traders from anywhere in the world, answer questions and help beginners – MQL5 community is developing along with you.

general discussions, trading systems, exchange, robots and expert advisors, technical indicators, trading articles, mql4 and metatrader 4

First Last Red Candle
19 Hi, Any one can help me with this situation as i'm new to mql4 programming. I would like to find the last red candle as it is showing in the attached photo. I know only one way to do that. But the problem is [What if the Red candle It is not in fifth bar back?] for Ex: If it In 4 or 3 or even In 7?...

Experts: Aaron JIN Scalper EA
Aaron JIN Scalper EA: Expert Advisor based on JIN Scalper Indicator. Author: Hemant Agarwal

Rubicon Expert Advisor - Based on Rubicon Indicator : Ready to use
1 I search on net Rubicon EA but didn't find.. so I develop my own... I did it with my little knowledge. If someone has good idea, please share. Steps To use: 1) Download Rubicon-Indicator.ex4 (I don't have mql4 file, I download .ex4 from internet) 2) Download Rubicon-Expert.ex4 3) Place...

Anyone could able to modify my code, so the alert can only be done right after the candle closed?
0 Hey guys, I have written a alert Indicator. I found out that, the Indicator send out alert before the candle closed. Kindly let me know which part of my code need to be amended. Thanks! #property Indicator_chart_window:#property indicator_buffers 2:#property indicator_color1 clrGreen...

Volume Indicator
50 Apparently, It is not impossible to have a volume indicator for Forex. <http://video.google.ca/videoplay?docid=-49473962417484315> Alternatively, the Indicator can count the number of ticks per bar to give an approximation of activity. Does anyone have a similar indicator?

Financial Trading Articles
Learn how to create your own technical indicators and trading

ZUP - Universal ZigZag with Pesavento patterns. Search for patterns
The ZUP indicator platform allows searching for multiple known patterns, parameters for which have already

This website uses cookies. Learn more about our [Cookies Policy](#).

Soon after the registration we will have access to a very interesting set of functionalities. For example, we can access the MetaTrader 5 web terminal, documentation that is extensive and up-to-date, key information on the global financial market, and other benefits.

The screenshot shows the official MQL5 Reference website. At the top, there's a navigation bar with links like 'WebTerminal', 'Documentation' (which is currently selected), 'Calendar', 'CodeBase', 'Articles', 'Freelance', 'Market', 'Signals', 'VPS', 'Forum', 'Login', 'Sign up >>', and language and search options. Below the header, there's a banner with the text 'DESCOBRI UMA MANEIRA FÁCIL DE OPERAR' and 'EM APENAS 3 CLIQUES'. The main content area is titled 'MQL5 Reference' and contains a sidebar with a list of topics such as Language Basics, Standard Constants, Enumerations and Structures, MQL5 programs, Predefined Variables, Common Functions, Array Functions, Conversion Functions, Math Functions, String Functions, Date and Time, Account Information, Checkup, Market Info, Timelines and Indicators Access, Custom Symbols, Chart Operations, Trade Functions, Trade Signals, Global Variables of the Terminal, File Functions, Custom Indicators, Object Functions, Technical Indicators, Working with Optimization Results, Working with Events, Working with OpenCL, and Standard Library. The main text area provides an overview of the MQL5 language, its development environment (MetaTrader IDE), and various resources available for traders and developers.

Something very important for anyone who is starting with the MQL5 language are the open source code samples available in **CodeBase**. In this location we will have access to codes for both the MQL4 and MQL5 languages. It is extremely valuable to spend time researching and studying these codes.

MQL5 WebTerminal Documentation Calendar CodeBase Articles Freelance Market Signals VPS Forum Login Sign up > English

Download MetaTrader 5

**FAÇA O QUE VOCÊ MAIS GOSTA
DEIXE ROBÔS NEGOCIAREM PARA VOCÊ!**

A MELHOR
RESCONTRIBUIÇÃO
AO MERCADO DA SÉCULA XIX

**EXECUTE
UM ROBÔ >**

MQL4 and MQL5 Source Code Library

You don't know where to start learning the MQL5 or MQL4 programming language? Various programs for your MetaTrader terminal are available here. Download and study published code examples, develop your own indicators and Expert Advisors. Publish your applications in the largest library of MQL5 and MQL4 codes, and they will be available in every MetaTrader terminal and MetaEditor.

Your code examples in MQL4 and MQL5 will be distributed worldwide, and thousands of traders will know about you!

MetaTrader 5

- Experts
- Indicators
- Scripts
- Libraries

MetaTrader 4

- Experts
- Indicators
- Scripts
- Libraries

watch how to download trading robots for free

Find us on Facebook! Join our fan page

Access the CodeBase from your MetaTrader 5 terminal.

Couldn't find the right code? Order it in the Freelance

MQL4 and MQL5 Source Code Library

Exp_AverageChangeCandle A trading system based on the signals of Indicator AverageChangeCandle.

XStdDevSpeed_Direction Indicator XStdDevSpeed_Direction shows the information on the market...

Exp_XRSIDeMarker_Histogram A trading system based on the XRSIDeMarker_Histogram Indicator signals.

Exp_ZOMA_Ichimoku_Oscillator A trading system based on the ZOMA_Ichimoku_Oscillator Indicator signals.

EMAVPS_StDev_HTF Indicator MAVPS_StDev with the timeframe selection option available...

Four_horizontal_lines The indicator plots four horizontal lines: Two basic ones (red by default...)

TST The Expert Advisor does not use any indicators. It only uses the current ...

EMAVPS_StDev

Coppock The Coppock Indicator of Bullish Signals.

Average_Modified_Moving_Average Average Modified Moving Average.

McClellan_Summation_Index Sherman and Marian McClellan Summation Index.

AD_Indicator Sherman and Marian McClellan Advances/Declines Indicator.

McClellan_Oscillator Sherman and Marian McClellan Oscillator.

SD_M_Swap A script for showing the swap value on a financial instrument. The texts...

AlliHeik Trading on indicator Heikin Ashi Smoothed Oscillator. Settings for plac...

Surefirething

We also have the section of **Articles** where tutorials with excellent didactics are added. This is one of the indispensable fields for anyone with the pretense of mastering the MQL5 language.

MQL5 WebTerminal Documentation Calendar CodeBase **Articles** Freelance Market Signals VPS Forum Login Sign up > English

Download MetaTrader 5

**FAÇA O DOWNLOAD DO METATRADER 5
E OPERE NUMA CONTA DEMO GRÁTIS**

DOWNLOAD

MQL4 and MQL5 Programming Articles

Study the MQL5 language for programming trading strategies in numerous published articles mostly written by you - the community members. The articles are grouped into categories to help you quicker find answers to any questions related to programming: Integration, Tester, Trading Strategies, etc.

Follow our new publications and discuss them on the Forum!

ZUP - Universal ZigZag with Posavento patterns. Search for patterns

The ZUP indicator platform allows searching for multiple known patterns, parameters for which have already been set. These parameters can be edited to suit your requirements.

Synchronizing several same-symbol charts on different timeframes

When making trading decisions, we often have to analyze charts on several timeframes. At the same time, these charts often contain graphical objects. Applying the same objects to

Multi-symbol balance graph in MetaTrader 5

The article provides an example of an MQL application with its graphical interface featuring multi-symbol balance and deposit drawdown graphs based on the last test results.

Deep Neural Networks (Part VI). Bayesian optimization of DNN hyperparameters

The article considers the possibility to apply Bayesian optimization to hyperparameters of deep neural networks, obtained by various training variants. The classification

Momentum Pinball trading strategy

In this article, we continue to consider writing the code to trading systems described in a book by Linda B. Raschke and Laurence A. Conner. "Street Smarts: High Probability Short-Term Trading..."

Creating a custom news feed for MetaTrader 5

In this article we look at the possibility of creating a flexible news feed that offers more options in terms of the type of news and also its source. The article will show how a web API can be...

NRTR

The NRTR indicator and trading modules based on HRTR for the MQL5 Wizard

In this article we are going to analyze the NRTR indicator and create a trading system based on this indicator. We are going to develop a module of trading signals that can be used in

Testing patterns that arise when trading currency pair baskets. Part II

We continue testing the patterns and trying the methods described in the articles about trading currency pair baskets. Let's consider in practice, whether it is possible to use the

Anyone who wants to build an EA with a more robust and complex strategy can hire MQL5 programmers from orders on the **Freelance** portal. Also, for those who have begun to master the tool and want extra income, they can accept projects and offer their robot production services, indicators and investment scripts.

The screenshot shows the MQL5 Freelance portal interface. At the top, there's a navigation bar with links like WebTerminal, Documentation, Calendar, CodeBase, Articles, Freelance (which is highlighted in yellow), Market, Signals, VPS, Forum, Login, Sign up >, and English. Below the navigation is a banner with the text "A PLATAFORMA #1 AGORA EM MACEIÓ!" and a small image of a person holding a globe. To the right of the banner is a button for "BAIXE O METATRADER 5". The main content area is titled "All orders" and shows a list of jobs. On the left, there's a sidebar with sections for "All orders" and "Rules", and a list of tips: "Watch how to order a trading robot", "Read the rules before you post an order or execute it", "How to order a trading robot and get the desired result", "Develop a robot in no time", and "Write trade operations easily". The main list of jobs includes one entry for "Jaffer khan" with a price range of "150 - 200 USD" and a description about experience in the trading market. The interface is in Portuguese.

In the field **Market** we can buy indicators and EA offered by individuals and specialized companies.

The screenshot shows the MQL5 website's Market section. On the left, there's a sidebar for MetaTrader 5 with 'Experts', 'Indicators', 'Libraries', and 'Utilities'. Below that is another sidebar for MetaTrader 4 with 'Experts', 'Indicators', 'Libraries' (which is highlighted), 'Utilities', 'Widgets', and 'Rules'. A link to 'Watch the Market tutorial video on YouTube' and a link to 'How to buy a trading robot or an indicator' are also present. The main content area features a banner with the text 'TESTE GRÁTIS O APLICATIVO ANTES DA COMPRA' and a button 'IR PARA O MERCADO'. Below the banner, there's a sub-section for 'MetaTrader 5' with a 'see all' link. Several items are listed, each with a thumbnail, name, price, and a 'details' link. These include: 'Synchronized Ch...' by Jiong Zhang (10 USD), 'CreateGridOrder...' by Konstantin Chernov (10 USD), 'Spread and Swap...' by Gennady Starinovych (12.95 USD), 'Profile Map MT5' by Dmitry Sapegin (180 USD), 'HedgeTerminalApi' by Vsevolod Solomin (50 USD), and 'Waves Of Fibonacci' by Roman Vashchenko (42 USD). There are also thumbnails for 'Fast Copy MT4', 'MARKET STATE', and 'Ats Infinity'.

We have a very interesting field that is Investment **Signals**. Through it we can follow the entry and exit of trader operations. We have the evolution of yields, the level of reliability, maximum profit, among other information, for each one of the traders who were willing to provide the signals of their operations. Some signal services can be contracted (paid for) and automatic execution can be synchronized with our accounts.

The screenshot shows the MQL5 website interface. At the top, there's a navigation bar with links like WebTerminal, Documentation, Calendar, CodeBase, Articles, Freelance, Market, Signals, VPS, Forum, Login, Sign up, and English. Below the navigation is a yellow banner with the text 'COMO ENCONTRAR TEMPO PARA O QUE REALMENTE IMPORTA? VOCÊ PODE AUTOMATIZAR A NEGOCIAÇÃO.' and a red button 'ESCOLHA UM ROBÔ'. On the left sidebar, there are buttons for Signals, Add widget, Rules, and Filter. A large blue banner on the left says 'COMPROU UM ROBÔ?' with an image of a hand holding a small wooden plaque with a robot icon. Below it is a green button 'ALUGUE UM VPS PARA NEGOCIAÇÃO'. The main content area features a search bar for 'MetaTrader 5' and 'Broker server', and a filter section with options like Maximum profit, Reliability, Profitable within a month, Intraday, For under 50 USD, Having reviews, and a sort icon. Below the search is a section titled 'Reliability' with three cards showing user profiles: Geraldo Palva (242% growth since 2016), YI XIONG (69% growth since 2017), and Pham Phong Son (50% growth since 2018). Each card includes a line chart and some small icons.

After finding a strategy and consistent setups and prove validity through testing the demo account, an important step is to hire a virtual machine service. This type of service will leave us more secure and protected with regard to lack of energy and internet.

Oscillations in the energy and internet network are problems that can compromise the proper functioning of our investment robots. An alternative to solving this problem is by hiring a Virtual Private Server (VPS) virtual machine. The MQL5 portal itself provides this kind of service.

MQL5 WebTerminal Documentation Calendar CodeBase Articles Freelance Market Signals **VPS** Forum **Login** Sign up > English

Download MetaTrader 5

TODOS OS INDICADORES MACROECONÔMICOS
EM TEMPO REAL

MetaTrader VPS

- Why you need hosting
- How to choose a server
- How VPS works
- Free hosting
- VPS plans and discounts
- How VPS pays off
- Rules

VOCÊ JÁ TEM SEU ROBÔ?

Forex VPS for MetaTrader 4/5

We will offer a server with the lowest ping and the best execution

Specify your broker's server →

Rent a Forex VPS for \$10 per month

Virtual hosting for MetaTrader 4/5 is the best VPS solution for Forex. It is cheap, it requires no configuration and it features minimum delays to the server.

For only \$10 per month, you obtain a virtual platform that works around the clock, saves your profit and pays off.

Create a remote copy of the application directly from the platform. Launch trading robots or activate a signal subscription on it in just a single click.

Traders choose our hosting calling it the best forex VPS solution for its unbeatable trade execution and high server availability.

There is the possibility of contracting the VPS virtual machine service by other companies, such as Amazon AWS (<https://aws.amazon.com/en/vpc/>) that even offers a free year of use.

Menu Entrar em contato com a equipe de vendas Produtos Soluções Mais Português Minha conta Crie uma conta gratuita

Amazon VPC Visão geral Recursos Definição de preços Conversões básicas Recursos Perguntas frequentes

Amazon Virtual Private Cloud

Provisione uma seção da nuvem da Amazon Web Services (AWS) isolada logicamente onde é possível executar recursos da AWS em uma rede virtual que você mesmo define.

Comece a usar a Amazon VPC

A Amazon Virtual Private Cloud (Amazon VPC) permite provisionar uma seção da Nuvem AWS isolada logicamente onde é possível executar recursos da AWS em uma rede virtual que você mesmo define. Você tem controle total sobre o ambiente de rede virtual, inclusive a seleção do seu próprio intervalo de endereços IP, a criação de sub-redes e a configuração de tabelas de rotas e gateway das redes. Você pode usar IPv4 e IPv6 na VPC para acessar recursos e aplicativos com segurança e facilidade.

E é possível personalizar facilmente a configuração da rede da Amazon VPC. Por exemplo, você pode criar uma sub-rede pública para os serviços web que têm acesso à Internet e clúster os sistemas de back-end, como bancos de dados ou servidores de aplicativos, em uma sub-rede privada sem acesso à Internet. Você pode aproveitar várias camadas de segurança, como security groups e listas de controle de acesso à rede, para ajudar a controlar o acesso às instâncias do Amazon EC2 em cada sub-rede.

Comece a usar a AWS gratuitamente

Criar conta gratuita Ou faça login na Conta

Receba doze meses de acesso ao nível gratuito da AWS e aproveite os recursos do AWS Basic Support, como atendimento ao cliente 24 horas por dia, 7 dias por semana, 365 dias por ano e flutuações de suporte, entre outros recursos.

We have the **Forum** field which is where we will probably spend a good part

of our learning journey answering and putting our doubts about the MQL5 codes. This is a good place to exchange experiences, make friendships and partnerships.

The screenshot shows the MQL5 forum homepage. At the top, there's a navigation bar with links for WebTerminal, Documentation, Calendar, CodeBase, Articles, Freelance, Market, Signals, VPS, Forum, Login, Sign up >, and English. Below the navigation is a banner for "EXECUÇÃO RÁPIDA DE OPERAÇÕES". A woman's face is on the left, and a green button says "Ping: 147 ms". To the right, there's a link to "ALUGUE UM VPS >". A message at the top right says "To post a new topic, please log in or register".

Below the banner, there are several posts listed:

- Volume Indicator** by moldem: Apparently, it is not impossible to have a volume indicator for Forus. <http://video.google.ca/videoplay?docid=-4894739832417484315> Alternatively, the indicator can count the number of ticks per bar to give an approximation of activity. Does anyone have a similar indicator?
- Where to download MT4?** by simotos: Hello! I'm new to this and I'm testing my Robot inventions to make changes, etc. The problem arises when testing the Robots, is SUPER, but SUPER difficult to test them in an orderly way, since never, but I have never been able to test a robot in the range of dates that I want, always the graph does...
- Any real traders here making a profit?** by EA-trader: Why is the commercial section in forex factory having 80 % of the threads? They failed and can't make money from trading. Why are there more signal sellers and ea /Indicator sellers , than I assume real traders? am talking about all forums and okers forums , and all the regulars on them. Any real...
- Update of MetaTrader 5 terminals for compatibility with Windows 10 April 2018 Update** by MetaQuotes Software Corp.: Update of MetaTrader 5 terminals for compatibility with Windows 10 April 2018 Update Today, we are releasing an update for desktop terminals, which fixes compatibility issues on the latest Windows version. We strongly recommend that you install this update. If you have already installed Windows 10...
- Elliot Wave** by winstondeonite123: Hi does any one know of a good free Elliot wave indicator I've found a few but would really like a recommendation cheers.
- who_has_tzmt4apl.zip** by halfbre: Hello, I have been searching tzmt4apl.dll/.zip for days, could anyone provide me with this file or an not expired link? thanks so much!
- USD/CAD close to v1line resistance** by Alexander London: The USD/CAD keeps its bullish trend, the next resistance may be the 1.3458 level and from there the price may try to stall or even break to the downside, but if oil keeps dropping, then we may see a breakout of that level. A breakdown of the bullish trendline may indicate a change in direction, but...
- VPS doesn't gain execution speed... why** by Volcanbleu: Hi guys, I just started recently test VPS solution (names removed) and I don't see any gain in final speed execution of the orders, even with a 1ms latency with the server. Compare to my...

Chapter 4

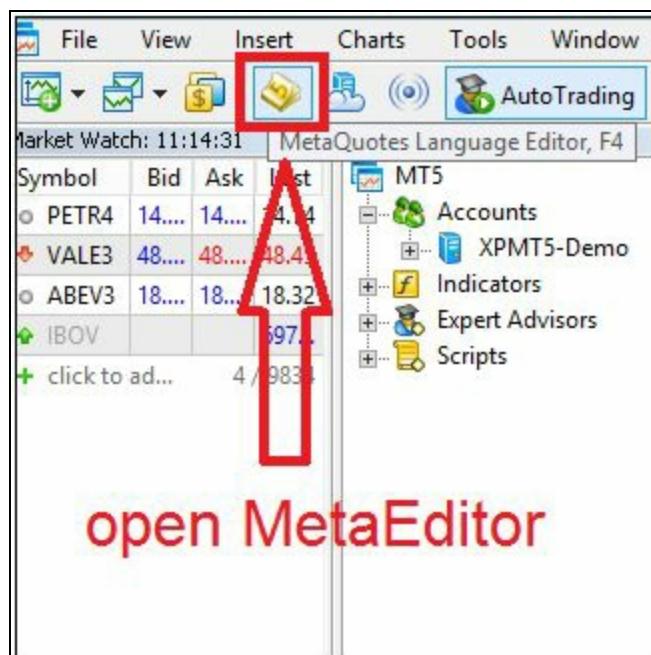
I. MetaEditor MQL5

The language **MQL5** (MetaQuotes Language 5) has a large number of useful functions to analyze the prices of financial assets. We need to know these functions so that we do not recreate the wheel. But before it, is important to know as much as possible of our language programming environment (MetaEditor).

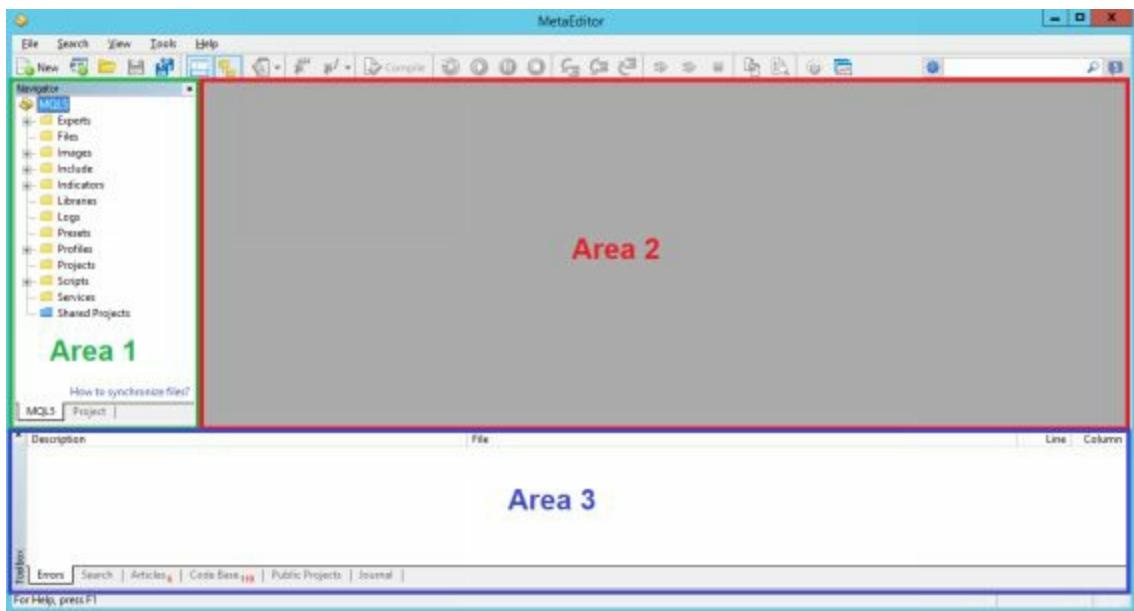
1.1. MetaEditor

MetaEditor is an Integrated Development Environment (**IDE**) for the MQL5 language that is included in MetaTrader 5. Let us now know the main features of developments offered by MetaEditor.

To open MetaEditor we must click the highlighted icon in the image below or press the **F4** shortcut.



After clicking on the icon for MetaEditor we will have a development environment with three main areas that will assist us in the development of our codes.



Area 1: **Navigator** - where we will manage our code library archives and libraries in development.

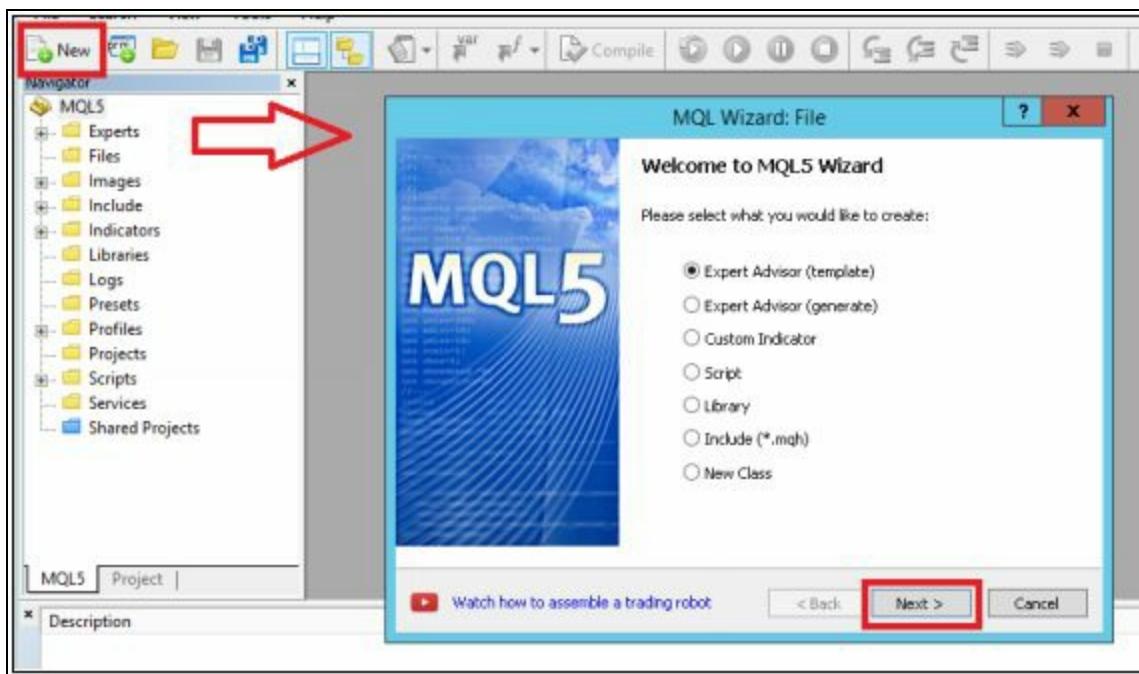
Area 2: Window where the codes are properly typed.

Area 3: **Toolbox** - here is where we will follow the development of our compilation and debugging tests. We have 6 tabs (Errors, Search, Articles, Code Base, Public Projects and Journal). Basically, for the creation of our algorithms we will only use the **Errors** and **Journal** tabs.



1.2. Creating a new Project

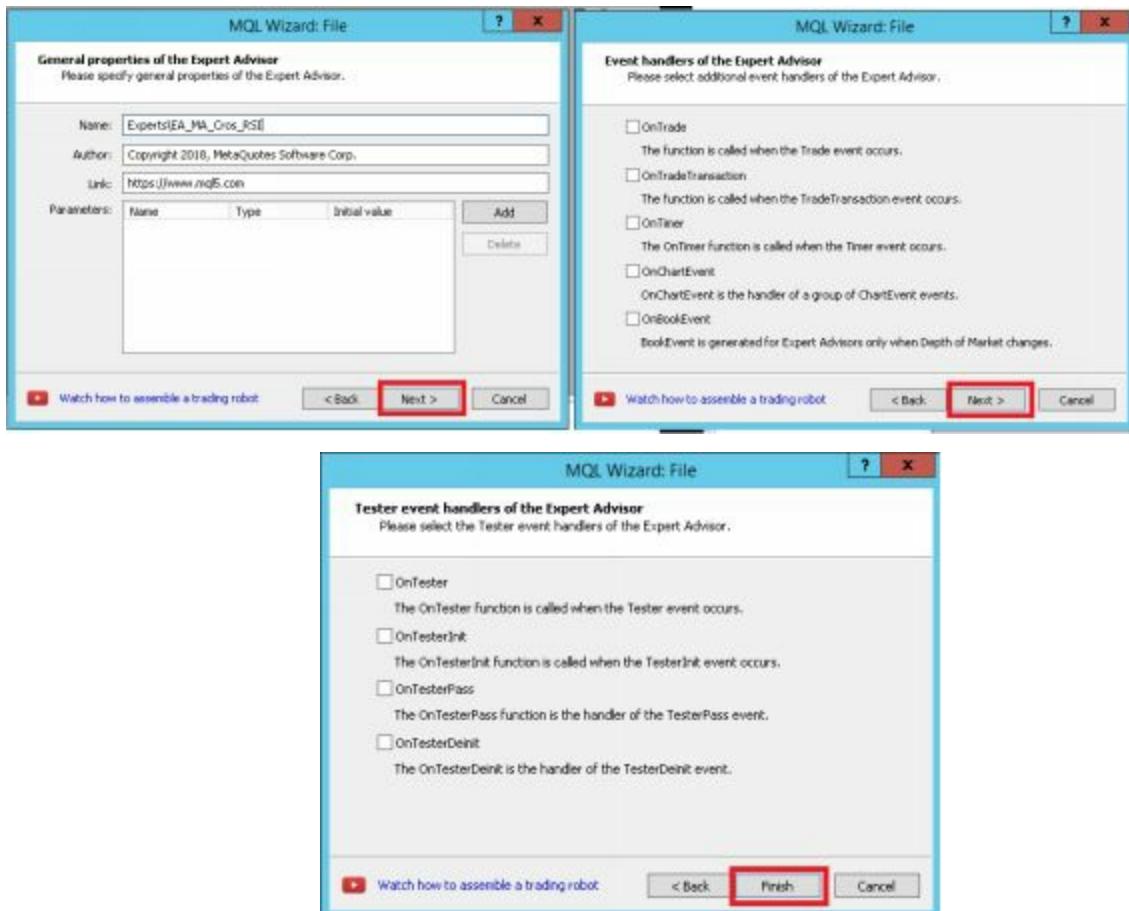
Let's click the **New** icon and then choose from the MQL5 wizard the type of project that we want to create.



We can notice that we have several design options. This **Wizard** even allows us to create investment robots without programming knowledge. Of course in this case we will be blind to most of the steps and we will hardly do anything with confidence and due security.

As this ebook intends to provide the first steps with the MQL5 tool we will only have the first option **Expert Advisor** (model). There are two^[3] other e-books from the same author that show you how to develop Indicator and Scripts codes.

Let's first name our investment robot (Expert Advisor). We'll call it **EA_MA_Cros_RSI** (Expert Advisor of Moving Moving Crosses with the RSI Dindicator). This is just a temporary name for MetaEditor presentation.



After clicking **Finish** the MQL Wizard creates for us the basic body required for the development of any investment robot. Therefore, we already have a divided structure with the main methods for the operation of an EA.

```

1 //+
2 //|
3 //| Copyright 2018, MetaQuotes Software Corp. |
4 //| https://www.mql5.com |
5 //+
6 #property copyright "Copyright 2018, MetaQuotes Software Corp."
7 #property link     "https://www.mql5.com"
8 #property version  "1.00"
9 //+
10//| Expert initialization function
11//+
12int OnInit()
13{
14//---
15
16//---
17    return(INIT_SUCCEEDED);
18}
19//+
20//| Expert deinitialization function
21//+
22void OnDeinit(const int reason)
23{
24//---
25
26}
27//+
28//| Expert tick function
29//+
30void OnTick()
31{
32//---
33
34}

```

Initialization of Variables

removal of functions after EA closure

data capture function

The structure of an EA, for its proper functioning, will always contain at least these three functions or methods: **OnInit ()**, **OnDeinit ()** and **OnTick ()**.

1.3. **OnInit()**

The first function executed in an EA is **OnInit ()**. It is triggered only when we initialize the robot. We use this function to initialize pointers (Handles) of indicators, variables, as well as make previous configurations and loads of variables and templates.

1.4. **OnDeinit()**

The **OnDeinit ()** function is called when the robot is removed from MetaTrader. For example, we use this function to clean indicator handles, variables, remove graphic and textual objects from the screen, and so on.

I.5. OnTick()

One of the most important is the **OnTick()** function because it is where a lot of our programming logic is running. This function is one of the most requested by our robot, since it is called every time a market operation happens. At each new sell, buy, release and order cancellation this function is called.

It is worth mentioning that this function is only called when we have EA within trading hours. That's because only with the trading in progress is that we have changes in the market.

If we are working on the development of a robot and we are not in the trading hours we will not see any call of this function. Alternatively, we can use the **OnTimer()** function to minimize these problems and to make tests and debugs through it (using the **Print()** function). Later in Chapter 5 we will see an example of using the **OnTimer()** function.

It is worth highlighting that there are several other predefined functions like **OnChartEvent()**, **OnTrader()**, **OnTester()**, and others. However, we will only study the main ones. Further details regarding these other functions can be found in the MetaTrader 5 documentation.

I.6. Programming Accessories

We have some elements (functions) that help us in the readability and organization of our robot codes.

I.6.1. Comments

Commenting on our algorithms is of the utmost importance for readability and understanding of key steps.

For example, we have the comment element we can represent by // or / * * /

```
1 //+-----+  
2 //| EA_MA_Cros_IFR.mq5 |  
3 //| Copyright 2018. |  
4 //| https://www.mql5.com |  
5 //+-----+  
6 #property copyright "Copyright 2018."  
7 #property link      "https://www.mql5.com"  
8 #property version   "1.00"
```

This type of structure (//) causes the compiler not to interpret the characters they saw immediately. So we can make small reminders and explanatory comments to make the code clear and readable for editing by other developers and even us.

The second type of comment / * * / allows you to write entire paragraphs between them.

```
12 /*  
13  
14 Here I can put a comment as a paragraph. I can skip the line  
15 between these special characters  
16  
17 */  
18
```

We have some shortcuts to the comments. For example, if we want to

comment several lines we can make the selection of them and press (Ctrl + ~). To remove the comments, simply select and press (Ctrl + C).

```
5 //+---  
6 //#property copyright "Copyright 2018."  
7 //#property link      "https://www.mql5.com"  
8 //#property version   "1.0"  
9 //---  
10
```

The shortcut (Ctrl +.) Lets you create the following annotated string. This is useful for separating large sections of code.

```
10  
11  
12 //+---  
13 //|  
14 //+---  
15  
16
```

The shortcut (Ctrl +;) lets you create a small separation with the following strings:

```
11  
12 //---  
13  
14
```



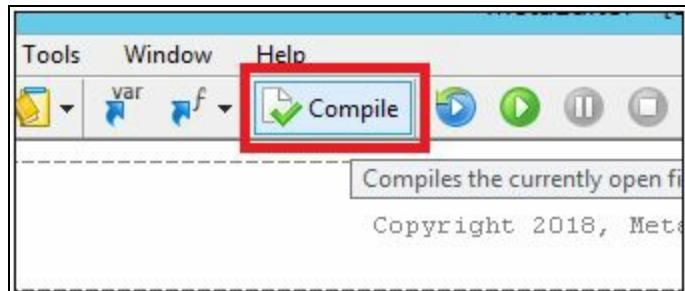
Ctr + ;

1.7. EA Properties

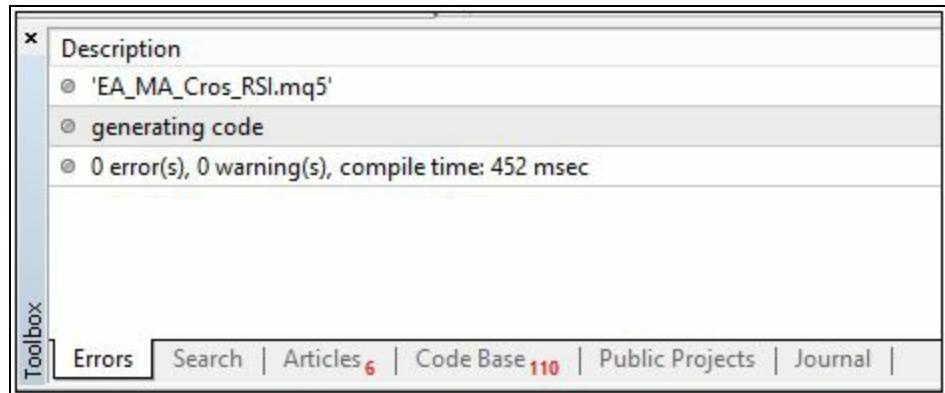
It is always good to inform details of robot development. Thus, directive **#property**: serves to describe basic information about EA. In the example below, where we have the property directive, we can observe information regarding copyright, link directed to some website and version of the robot.

```
1 //+
2 //|                               EA_MA_Cros_IFR.mq5 |
3 //|                               Copyright 2018. |
4 //|                               https://www.mql5.com |
5 //+
6 #property copyright "Copyright 2018."
7 #property link      "https://www.mql5.com"
8 #property version   "1.00"
```

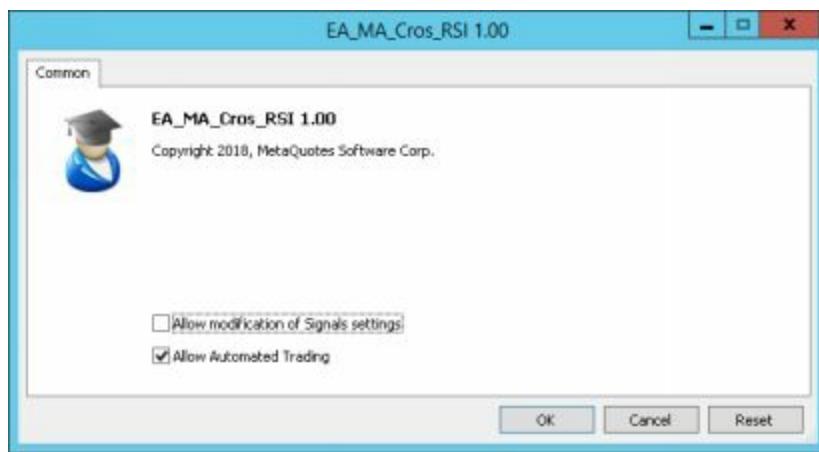
After pressing the Compile button:



We note that in the Toolbox field on the **Errors** tab the compiler did not raise any problems.



So automatically in the MetaTrader Platform in the field **Navigator** -> clicking on the more of Expert consultant we can find our **EA compiled** (.exe5). You can now click, hold and drag to the candle chart and observe the **#property** information present.



As we still have practically nothing programmed from the robot these information in the **Common** tab above, are the only available so far. Then, with each new modification made in the code with MetaEditor we can observe the changes in the field **Navigator** in the part of experts.

1.8. MetaTrader and MetaEditor File Types

The file type generated with MetaEditor has the extension '.mql'. We can verify this in the MetaEditor Navigator field where we have our robot edition file called EA_MA_Cros_RSI.mql5.

After clicking on the button **Compile** a file with the extension 'EA_MA_Cros_RSI.exe5' is automatically created in the field **Navigator -> Expert Advisor** of MetaTrader.

It is important to note the difference between MetaTrader and MetaEditor. In **MetaEditor** we have files with extension '**.mql5**' and in **MetaTrader** we have '**.exe5**' files.

I.9. Adding Libraries

We can add special libraries provided by MQL5. For this we just need to know the name of the library with the following code structure:

```
11  
12 //--- Includes  
13 #include <ChartObjects/ChartObjectsFibo.mqh>  
14 #include <Trade/Trade.mqh>  
15
```

The above example adds a set of classes and methods ready to generate Fibonacci figures in the first include <ChartObjects/ChartObjectsFibo.mqh>, and work with sending and managing sales orders in the second include <Trade/Trade.mqh>.

Note that the extension of these include files is of type '.mqh'. MetaEditor also allows you to create this type of file. From this we can create complex algorithms that work with the graphic part and even visual animations.

We can research and study the main MQL5 libraries from the site:

The screenshot shows the MQL5 website interface. The top navigation bar includes links for Terminal Web, Documentação, Calendário, CodeBase (which is highlighted in yellow), Artigos, Freelance, Mercado, Sinais, VPS, Fórum, and a Portuguese language switcher. A prominent red banner in the center says "FINALMENTE, POSSO VER PREVISÕES EM TEMPO REAL!" with a "VER" button. On the left, there's a sidebar with categories for MetaTrader 5 (Especialistas, Indicadores, Scripts, Bibliotecas), MetaTrader 4 (Especialistas, Indicadores, Scripts, Bibliotecas), and Acessos (Acessar o CodeBase do seu terminal MetaTrader 5, Não foi possível encontrar o código adequado). The main content area displays a list of user-voted best libraries for MetaTrader 5, such as EasyAndFastGUI, ALGLIB, IncGUI, CCalendar, HDS Hash, Funções Estatísticas, IncAMDArray, and Modulo de sinais de negociação baseado no Indicador HorLag.

In Chapter 5 we will know and study better MetaEditor functionality from the study of the MQL5 syntax.

Chapter 5

i. Basic Programming Logic with MQL5

As mentioned before, the MQL5 language has great similarities with the C++, Java, and C # languages. Therefore, anyone who has any knowledge of these languages will have great ease in understanding the MQL5 operating syntax.

Every algorithm, every recipe for cake, begins with the description of the ingredients. So let's make an analogy and start studying the ingredients of our EA, our cake.

Let's first understand the main types of variables present in MQL5. We also need to know how this language declares and assigns values to variables.

1.1. Types of Variables

The variable is the basic unit of data storage in any programming language. Almost every algorithm needs variables. In MQL5 we need to declare a variable always with some **type**.

The main types of variables used to create robots are: Integer (int), floating point - decimal values (float and double), alphanumeric characters (string), logical (bool) and dates (datetime).

Unlike MQL5 there are some languages that have automatic typing with, for example, Python.

Having to explicitly declare the variable type has practical security advantages and makes it easy to find compilation errors. When we are limiting the type of the variable, we protect it from possible arithmetic errors, assignment errors, and programming logic errors.

1.2. Declaration of Variables

Every MQL5 variable must have a type and a name. The type of this variable must always be specified before the name. Then we can assign values corresponding to the variable type. Let's study the most used and that will be part of the structure of our EA.

1.2.1. *Integer Type*

We have some peculiarities regarding the memory consumption required to store a variable. In most EAs this is not a problem, but we will discuss these issues in case you want to leave your robot optimized for the memory consumption requirement of processing.

For example, we have the type variables **char**, **short**, **int**, **uchar**, **ushort**, **uint** that serve for integer variables.

char - uses only 1 byte of memory. You can allow values in the range of -128 to 127.

short - uses 2 bytes of memory. Can support values in the range of -32,768 to 32,767.

int - uses 4 bytes of memory. You can allow values in the range of -2,147,48,648 through 2,147,483,647.

uchar - uses 1 byte of memory. The range of values is 0 to 255.

ushort - uses 2 bytes of memory. The range of values is 0 to 65,535.

uint - uses 4 bytes of memory. The range of values is 0 to 4,294,967,295.

See the example below:

```
13  
14 int movingAveragePeriod = 24;  
15
```

In this case we have a variable with the name ‘movingAveragePeriod, which accepts integer values (int). We can see a value assignment equal to 24. If no value is assigned to the variable the MQL5 immediately assigns zero.

2.2. Type Double

When we are interested in storing variables with decimal numbers the most appropriate is to use the double type. The double type consumes 8 bytes of memory. More economical, we have the float type that consumes half, 4 bytes.

The main difference between these two types (float and double) is that the accuracy of significant digits is larger for the double type (15 significant digits) while the float has only 7 significant digits. As the difference in memory consumption for our robots will be irrelevant we will use only the double type that is more accurate.

For example, the division, in the figure below, does not involve many numeric digits. Therefore, the double or float type makes no difference. On the other hand, if we had a problem where the result was some periodic titling multiplied by some number or other decimal, cumulative rounding errors could interfere with the accuracy of the results.

```
14 int movingAveragePeriod = 24;
15
16 double valueRSI = 1.611;
17
18 double a = 1.0;
19 double b = 0.5;
20
21 double c = a / b;
```

To the present moment do not worry about not observing the results of operations. We will do this later with the use of the **Print()** function. For the time being, let's take a look at the structure of the statement of variables.

.2.3. Type String

Textual character variables are typed with string. The MQL5 language allows string concatenation in a very simplified way as shown below.

```
22
23 string s_1 = "My first ";
24 string s_2 = "Expert Advisor!";
25
26 string s_t = s_1 + s_2; // return: My first Expert Advisor!
27
```

For a string if no value is assigned, an empty string (NULL) is placed.

.2.4. Type bool

When the interest is to have a variable with boolean characteristics, that is, that only allow true values (true) or false (false) we can use the type bool:

```
28 int x = 10;
29 int y = 20;
30
31 bool result; //without initial assignment we have (false)
32
33 result = x < y; // return (true)
34
```

It should be noted that in the case the variable 'bool response' was declared without initial value, so the value of (false) is immediately assigned. Only then, see figure above, that we have a value change to true, because **x** is less than **y**.

.2.5. Type Datetime

This is one of the most widely used types, since asset price data are collected

over some time scale. We will always be able to make evaluations of times and days for operations. Therefore, we should be well aware of the use of this type of variable.

The date and time format for MQL5 is defined as follows: {D'yyyy.mm.dd hh: mm: ss'} ie {year.mon.day hours: minutes: seconds}

The capitalized D letter placed at the beginning of the date causes the variable to have the datetime constant.

```
36 datetime today = D'2018.07.17 00:00:00'; // full description
37
38 datetime today2 = D'2018.07.17 00';        // just hour
39
40 datetime today3 = D'2018.07.17';           // just date
```

3.3. Declaring Constants

We have a way to create constants from global declarations at the beginning of our code from the **#define** directive.

```
| 10 #define PI  3.1415  
|  
11  
12 #define NOME_EMPRESA = "1° EA LTDA."  
13
```

Of course, the values of the constants can not be changed. If there is any attempt to change the value, we will have a compilation error.

It is standard to consider the names of constants with upper-case characters.

4. Vector Variables: Arrays

An array is a variable that supports multiple values. We may think that an array is something like a list of values with a certain type. Within this list we can do iterations by traversing each of its values.

We will often use variable arrays to store mainly financial asset price information and indicator variables.

Here is an example of a three-position array declaration for storing integer numbers:

```
42 int myArray[3]; // creates a variable and allocates three int positions
43
44 myArray[0] = 10; // saves the value 10 in the zero position
45
46 myArray[1] = 20; // saves the value 10 in the one position
47
48 myArray[2] = 30; // saves the value 10 in the two position
```

If we try to do:

```
49
50 myArray[3] = 40; // we will have a compilation error!
51
```

This compilation error happens because we made an initial statement (myArray [3]) to allocate only 3 memory locations and we are trying to add a fourth position.

We can make our array a list of the type of variable we want. However, in advance, MQL5 needs to know the size of the list of values we need for the array. So putting some value in brackets soon after the variable name is required.

Another way to declare arrays is to directly assign values to it. We should do this in the following way:

```
52  
53 double myArray[4] = {3.14, 2.22, 1.61, 17.21};  
54
```

1.5. For loop

We will now study a very important repetition loop used in the development of investment robots. When we want the computer to perform repetitive activities within certain predetermined limits, a practical way of informing the machine is through the **for loop**.

When we type the word **for** in MetaEditor we will have a small set as shown below:

```
62
63 for|
64
65
```

When we press **Tab**, the editor will autocomplete the whole structure of the **for loop**.

```
62
63 for(int i=0;i<total;i++)
64 {
65
66 }
67
```

This structure means that we will iterate from the integer 0 (zero) to the integer (total-1) with an iteration increment ‘i’ of a unit ($i++$ is the same as $i = i + 1$, as well as $i--$ is the same as $i = i - 1$).

So if we want to make a scan inside the elements of an array we can do from the **for loop** as follows:

```
53 double myArray[4] = {3.14, 2.22, 1.61, 17.21};  
54  
55 for(i i=0;i<4;i++)  
56 {  
57     Print("myArray["+i+"] = ", myArray[i]);  
58 }  
59  
60 // out: myArray[0] = 3.14  
61 //       myArray[1] = 2.22  
62 //       myArray[2] = 1.61  
63 //       myArray[3] = 17.21
```

1.6. Enum

The **enum** structure is a special type of integer where we can define, through a list, assignment constants. With enum we can customize the types of variables that best suit our problem.

For example, let's create an enum to represent the seasons of the year. We know that there are only 4 seasons of the year: spring, summer, autumn and winter.

When you type the word **enum** we have a small arrow pointing to the right, as shown below:

```
18
19 enum →
20
```

Another Tab, we have autocomplete. Our enum should look like this:

```
13 enum SEASONS
14 {
15     spring,
16     summer,
17     fall,
18     winter,
19 };
```

We can now use ‘SEASONS’ as a type of variable to be declared. In many situations this is very useful to make our code more intuitive and organized.

```
21 SEASONS season;
22
23 season == summer;
24
25 Print(season); // out: 1
```

The value of output was 1 (one) because enum considers the ordering of

integers from zero. In our case we have spring (0), summer (1), fall (2), winter (3). Interesting and quite useful the **enum**, right?

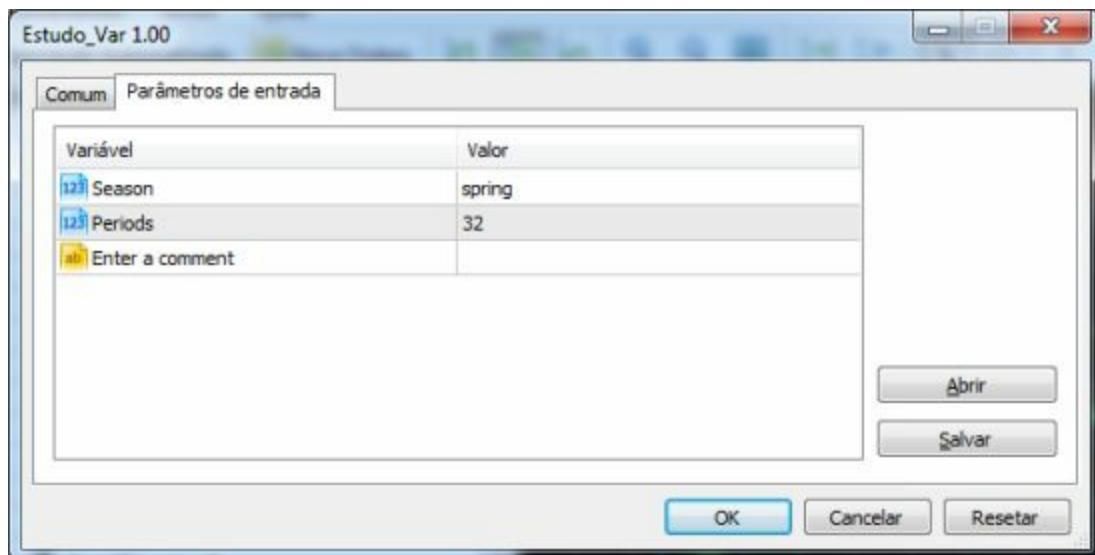
1.7. Input Type Variables

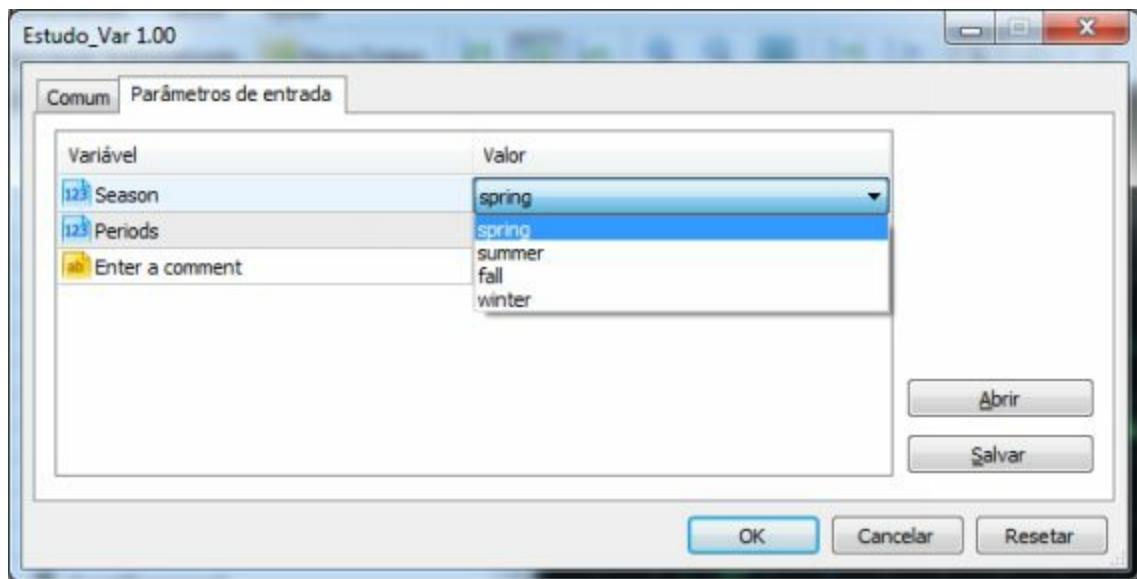
Input variables are the only ones that allow the user of the indicators or EA to assign values to them. These variables are used to provide the user with the possibility of changing the setups of the indicators, stop loss, take profit, number of lots etc.

Input variables can be of any type including enum types. Let's take an example to make it easier to understand.

```
13 enum SEASONS
14 {
15     spring,
16     summer,
17     fall,
18     winter,
19 };
20
21 input SEASONS season = spring; // Season
22
23 input int numPeriod = 32;      // Periods
24
25 input string comment = " ";   // Enter a comment
```

After compiling the above code we will have in MetaTrader the following possibility of setting the EA:





Therefore, the input directive allows us to dialogue with the user on the **Input** tab of our EA.

8. Local and Global Variables

It is very important to know the distinction between local variables and global variables. The confusion between these two types of variables leads to many EA compilation and execution errors. Therefore, we must be very careful and pay close attention to where we are making the declaration of variables.

8.1. Local variables

Let's create a function to show an example of local variables. When variables are declared within a function/methods they with their assigned values will only be used within functions. So we have local variables because they are declared within a method or function.

```
28 // test function with void return
29 void funTest()
30 {
31
32     int movingAveragePeriod = 7;
33
34     int multiplier = 2;
35
36     Print(movingAveragePeriod * multiplier); // out: 14
37
38 }
```

8.2. Global Variables

When we want a variable to be used by any function/method anywhere in our algorithm, we must declare it globally as shown in the example below. Pay close attention to this example as it is very important to understand the difference between the variables.

```

28 int varGlobal_1 = 3;
29
30 int varGlobal_2 = 4;
31
32 // test function with void return
33 void funTest()
34 {
35
36     int movingAveragePeriod = 7;
37
38     int multiplier = 2;
39
40     Print(movingAveragePeriod * multiplier);    // out: 14
41
42     Print(movingAveragePeriod * varGlobal_1); // out: 21
43
44     Print(varGlobal_1 + varGlobal_2);           // out: 7
45
46 }
47
48 // 'movingAveragePeriod' is local
49 Print(movingAveragePeriod * multiplier); // out: compile error!
50
51 Print(varGlobal_1 * varGlobal_2);         // out: 12

```

Let's take another important example, now with the chaining of a structure with the if-conditional Boolean. Carefully evaluate this example.

```

23 int varLoc_0 = 11;
24
25 void secondFunTest()
26 {
27     int varLoc_1 = 3;
28
29     int varLoc_2 = 7;
30
31     if(varLoc_2 > varGlobal_1)
32     {
33         int varLoc_3 = 10;
34
35         int varLoc_4 = 20;
36
37         Print(varLoc_0+varLoc_1+varLoc_2+varLoc_3+varLoc_4); // out: 51
38     }
39
40     Print(varLoc_0+varLoc_1+varLoc_2+varLoc_3+varLoc_4); // out: compile error!
41     // varLoc_3 and varLoc_4 are local inside 'if'
42
43     Print(varLoc_0+varLoc_1+varLoc_2); // out: 21
44 }
45
46 Print(varLoc_0+varLoc_1+varLoc_3); // out: compile error!
47 // varLoc_1 and varLoc_2 are local variables

```

1.9. Variables Predefined by MQL5

The MQL5 language has several predefined routinely used variables for strategy development. These variables have their own notation and are preceded by an underscore (_).

The main variables most used and that we will apply to create our first EA are:

_Symbol: represents the financial asset symbol present on the current chart shown on the screen.

_Period: refers to the period, in minutes, of the current candle chart.

_Point: represents the size of points that characterize the financial asset.

_Digits: represents the number of decimal point digits of the current asset.

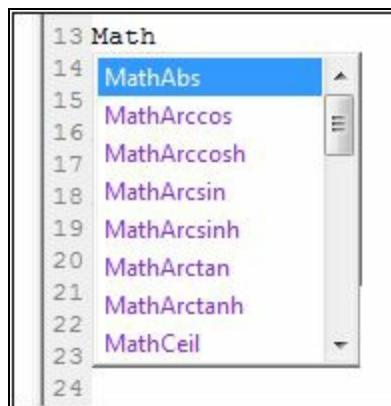
The use of these variables will become clearer when we are developing our EA.

10. Math Operations

We have already seen several mathematical operations. Let's study some more of them. The MQL5 language provides us with a very intuitive and simple structure for working with mathematical operations.

```
14
15 // Adição e subtração
16 int varA = 2 + 3;           // Resultado: 5
17
18 double varB = 2.5 + varA; // Resultado: 7.5
19
20 double varC = varA - varB; // Resultado: 2.5
21
22
23 // Multiplicação
24 int varD = 2 * 3;           // Resultado: 6
25
26 double varE = 2.5 * varD; // Resultado: 15
27
28 // Divisão
29 double varF = 5 / 2;         // Resultado: 2.5
30 int     varG = 5 / 2;         // Resultado: 2
31
```

The language itself already comes with a set of mathematical and statistical functions to work connected with the most diverse operations. We have the functions with radical MathXXX quite useful:



11. Logical and Conditional Relationships

In the structure of our algorithms we commonly use logical relations that return values of true or false. The main relationships we will be working on are represented as follows by MQL5:

```
54 A > B // greater than
55
56 A < B // less than
57
58 A >= B // greater than or equal
59
60 A <= B // less than or equal
61
62 A == B // equal
63
64 A != B // different
```

See the use of the **if** conditional plus the logical connectives **&&** (AND) and **||** (OR):

```
67 int A = 7;
68 int B = 7;
69 int C = 10;
70
71 if(A == B && A+B == C) // && is the boolean AND operation
72 {
73     Print(true); // Out: true
74 }
75
76 if(A == B || A == C) // || is the boolean OR operation
77 {
78     Print(true); // Out: true
79 }
```

Let's now go to the conditional structure of the **else** type. The **else** is activated if the '**if**' conditional is not true. Therefore, if the '**if**' is false automatically the **else** becomes true and everything inside the keys of this structure is executed.

```
61
62 int var1 = 1;
63 int var2 = 4;
64 int var3 = 7;
65
66 if(var1 >= var3)
67 {
68     Print("Maior Var1 >= Var3");
69 }else
70 {
71     Print("Var1 não é >= Var3");
72 }
73
```

The **else if** structure is very useful in many cases. When the chain of conditions is extensive we can use this structure. If no **else if** is true, **else** executes automatically.

```
76
77 int var1 = 1;
78 int var2 = 4;
79 int var3 = 7;
80
81 if(var3 == 1)
82     Print("Var3 = 1");
83
84 else if(var3 == 4)
85     Print("Var3 = 1");
86
87 else
88     Print("Var3 = ",var3);
89
```

.12. Ternary Operator

The ternary operator exists in several high-level programming languages and in MQL5 it is no different. This is a very useful operator to simplify conditional structures that could use **if** and **else**.

The ternary operator has the following operating structure:

(conditional)? (alternative 1 if true): (alternative 2 if false)

Let's take an example:

```
102 bool condicional = true;
103
104 bool resultado = condicional ? true : false;
105
106 Print(resultado); // Out: true
107
```

In the place where we use the conditional variable we could do any structure using the logical evaluators ($>=$, $<=$, $\&\&$, $||$).

.13. Methods or Functions

When we are working with object-oriented programming (OOP) we make constant use of classes. Classes are the master plans of the objects architecture. Do not worry if this seems complicated because we will not use OOP.

Among the classes we usually find methods and attributes. The term function is most commonly used when we are working with procedural programming. The MQL5 language allows us to program both in the object-oriented and procedural paradigm. This book is about procedural programming, so we will talk about methods and functions as if they were the same thing.

But what are methods or functions? They are code blocks that allow you to perform a specific task. Usually the functions have input and return parameters. In other words, the functions process the parameter entries according to a well-defined logic and returns a result.

Here is an example to further clarify all this:

Return type	input parameters
80 ↴ 81 82 bool makeBuy(string symbol, double TK, double SL, int lots) 83 { 84 // 85 // function logic 86 // 87 return false; 88 } 89	↳

In a function we can have several types of returns and input parameters. The processing performed depends on the logic of the problem.

Let's take another example where we have functions used to calculate the sum, multiplication, and division of two numbers:

```
90 double sum(double a, double b)
91     {
92         double s = a + b
93         return s;
94     }
95
96 double multiplication(double a, double b)
97     {
98         double m = a * b
99         return m;
100    }
101
102 double division(double a, double b)
103    {
104        double d = a / b
105        return d;
106    }
```

Functions that do not return values have a **void** type declaration. However, if the function has some kind of return we are **required to return something** relative to the return type.

```
108 void sendMessage(string msg)
109     {
110
111         Print("This is your message : ", msg);
112
113     }
```

14. Candles and Tick Variables

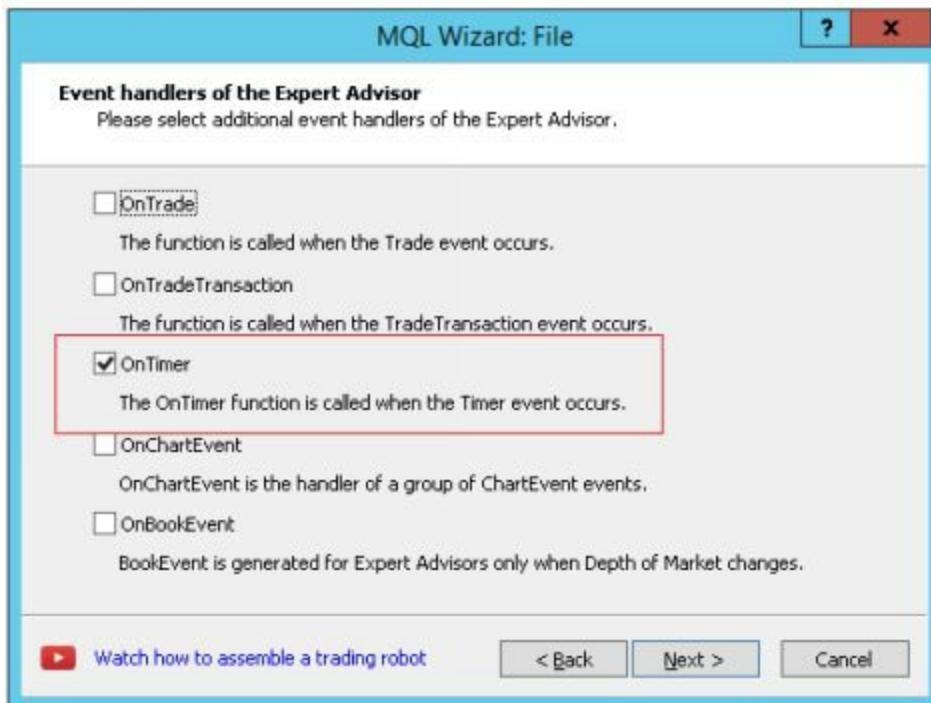
The MQL5 language has a set of libraries prepared to work with the acquisition and manipulation of **candle** prices. We have the **MqlRates** structure that stores the price information of opening, closing, maximum, minimum, volume, spread and time.

Tick information refers to those collected from the book of offers such as ask (bid), bid (sale), last business etc. This information can be accessed from the **MqlTick** structure.

Let's see how to declare a candle variable and tick. We must beware, because we must first declare the variable after doing the proper loading of the data in the variable declared from other specific functions.

In order to load, we must specify the financial asset (which can be the current one of the graph using **_Symbol**) and the period of the candles (to use the current chart we put **_Period**).

For an example, we will create a new Expert Advisor with the name of Study_Var (study of variables). In the wizard now we will add the **OnTimer** option.



We decided to add the **OnTimer()** function, because with it we can work and view our debug codes outside the trading hours. As we saw in Chapter 4, the **OnTick()** function is called every time a new trading operation is performed, and as we are probably making the codes outside trading hours no call to this function will be made.

So we'll use the **OnTimer()** function to make timed calls and be able to see the development of our codes. Let's ask it to be called every **2 seconds** from the **EventSetTimer(2)** function (see example below).

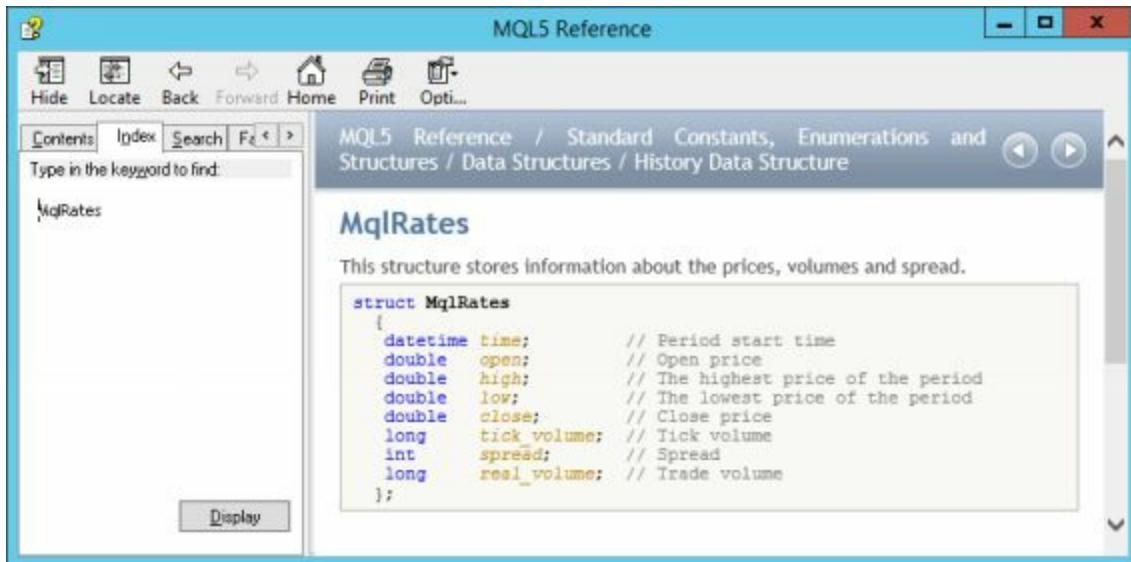
```
13 MqlRates candle[]; // global variable candles
14
15 MqlTick tick;      // global variable tick
16
17 int OnInit()
18 {
19 //--- create timer
20 EventSetTimer(2); //every 2 seconds the OnTimer () function will be called
21
22 // Loads the variable candles 10 positions with prices:
23 CopyRates(_Symbol,_Period,0,10,candle);
24
25 // Sort the vector of 10 positions, position 0 is the current price
26 ArraySetAsSeries(candle,true);
27
28 // returns current prices of a specified symbol in a variable of the MqlTick type.
29 SymbolInfoTick(_Symbol,tick);
30
31 //---
32 return(INIT_SUCCEEDED);
33 }
```

The **CopyRates()** and **SymbolInfoTick()** functions have been completed following the following structures:

```
int CopyRates(
    string          symbol_name,           // symbol name
    ENUM_TIMEFRAMES timeframe,            // period
    int             start_pos,             // start position
    int             count,                 // data count to copy
    MqlRates        rates_array[]         // target array to copy
);
```

```
bool SymbolInfoTick(
    string symbol,           // symbol name
    MqlTick& tick            // reference to a structure
);
```

We can better understand the structures and main functions of the MQL5 library by using the **F1** key. To do this, just place the mouse cursor on the structure of interest and press F1. For example, let's do this for **MqlRates**:



From the F1 key we have access to a very complete documentation of the MQL5 language. So we should always use it when a doubt arises.

As already discussed, we are not always developing EA in the trading hours and even then we need to make some **debugs** in the code, so we are using **OnTimer()**.

However, within the OnTimer() function we will add the following code:

```

51 //+-----
52 //| Timer function
53 //+-----
54 void OnTimer()
55 {
56 //---
57
58 Print("Open price = ", candle[0].open);
59 Print("Max price= ", candle[0].high);
60 Print("Close price = ", candle[0].close);
61 Print("Min price = ", candle[0].low);
62 Print("=====");
63
64 }
65 //+-----

```

Let's compile. Now let's go to MetaTrader and drag EA 'Study_Var.exe5' to the chart. Then in the Toolbox (*if it is not showing go to the display menu and ask to open*) and enable the Experts tab.

Once the robot has been added to the graph, we can notice that on this tab every 2 seconds we have an update of the **OnTimer()** function and what is inside it is executed. In our case we see the prices of open, max, close and min.

Horário	Fonte	Mensagem
o 2018.06.05 12:42:18.522	Estudo_Var (BRF53,D1)	Freio Max = 24.93
o 2018.06.05 12:42:18.522	Estudo_Var (BRF53,D1)	Freio close = 24.43
o 2018.06.05 12:42:18.522	Estudo_Var (BRF53,D1)	Freio Min = 24.03
o 2018.06.05 12:42:18.522	Estudo_Var (BRF53,D1)	=====
o 2018.06.05 12:42:20.537	Estudo_Var (BRF53,D1)	Freio Open = 24.35
o 2018.06.05 12:42:20.537	Estudo_Var (BRF53,D1)	Freio Max = 24.93
o 2018.06.05 12:42:20.537	Estudo_Var (BRF53,D1)	Freio close = 24.43
o 2018.06.05 12:42:20.537	Estudo_Var (BRF53,D1)	Freio Min = 24.03
o 2018.06.05 12:42:20.537	Estudo_Var (BRF53,D1)	=====

We are taking the prices of the candle in position '0', that is, collecting the values of the most current candle. The candle can represent any time chart, what really matters is your position. We can choose any desired position, all we need is to understand the ordering of the vector. Below is an illustration to facilitate this understanding (it is worth noting that this illustration is valid only after using the **ArraySetAsSeries()** that orders the vector as shown).



Our candle vector has only 10 positions, because we use the function **CopyRates(_Symbol, _Period, 0,10, candles)**.

We can make our code more easily able to choose the position of the candles and, in addition, add **tick** information, see below:

```

54 void OnTimer()
55 {
56 //---
57     int pos = 0; // choose candle position
58
59     Print("Open price = ", candle[pos].open);
60     Print("Max price= ", candle[pos].high);
61     Print("Close price = ", candle[pos].close);
62     Print("Min price = ", candle[pos].low);
63     Print("Last tick = ", tick.last);
64     Print("Time = ", tick.time);
65     Print("=====");
66
67 }
```

Let's now test the case of the variable (pos = 10). We chose for candles a vector with only 10 price positions, but we are trying to access the position of number 11 (remembering that the count starts with zero). So **the code will compile normally**, however when executed in MetaTrader we will get the following error message:

Horário	Fonte	Mensagem
2018.06.05 13:01:47.517	Estudo_Var (ABEV3,D1)	array out of range in 'Estudo_Var.mq5' (59,32)

Negociação | Exposição | Histórico | Notícias [99](#) | Caixa de E-mail [5](#) | Calendário | Empresa | Mercado | Alertas | Sinais | Biblioteca | [Experts](#) | Diário |

We have an array out of range error. MetaTrader also informs the line of code in the **.mql file** where we find the violation (line 59). We must be alert because this type of error is quite frequent in our robot creations. So be very careful with the sizes and tracking positions of the arrays.

.15. Functions Comment() and Alert()

By using some useful functions we can create warning elements and alerts to provide greater security of use and better understanding of what is happening with our EAs.

.15.1. Comment()

Comments are always welcome. Let's take an illustrative example:

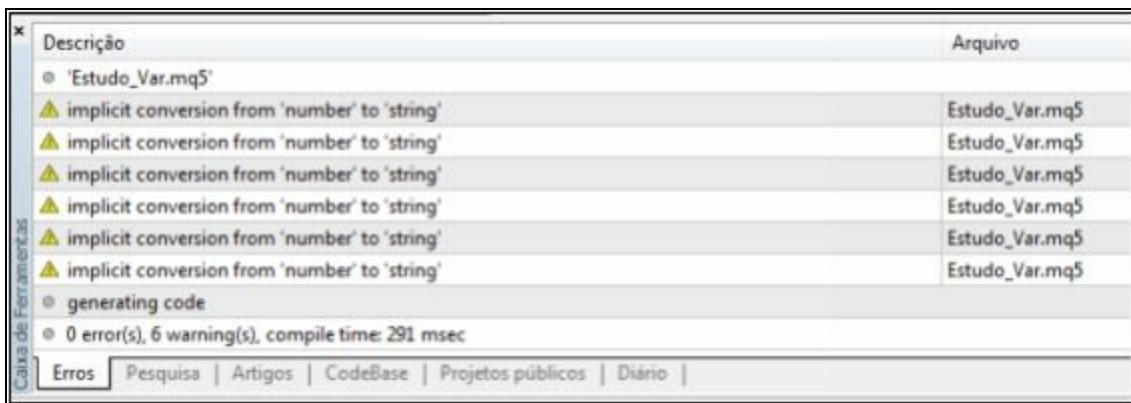
```
54 void OnTimer()
55 {
56 //---
57     int pos = 0; // choose candle position
58
59     string legend = "Open price = "+DoubleToString(candle[pos].open,2)+"\n"+
60                 "Max price = "+ DoubleToString(candle[pos].high,2)+"\n"+
61                 "Close price = "+ DoubleToString(candle[pos].close,2)+"\n"+
62                 "Min price = "+ DoubleToString(candle[pos].low,2)+"\n"+
63                 "Last tick = "+ DoubleToString(tick.last,2)+"\n"+
64                 "Time = "+ tick.time;
65
66     Comment(legend);
67
68 }
```



We are using the **DoubleToString()** function to make the change from double to string. If we are not careful to use this function we will not have a compilation error, but in the **Errors** tab of MetaEditor **Toolbox** we will

observe the following composition of warnings:

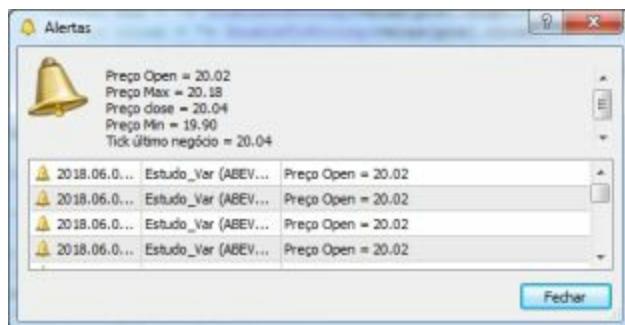
```
54 void OnTimer()
55 {
56 //---
57     int pos = 0; // choose candle position
58
59     string legend = "Open price = "+candle[pos].open+"\n"+
60                 "Max price= "+ candle[pos].high+"\n"+
61                 "Close price = "+ candle[pos].close+"\n"+
62                 "Min price = "+ candle[pos].low+"\n"+
63                 "Last tick = "+tick.last+"\n"+
64                 "Time = "+ tick.time;
65
66     Comment(legend);
67
68 }
```



.15.2. Alert()

The **Alert()** function displays an alert box in the center of the screen with the requested information. Those who are interested can search for audible alerts, because it is also possible to add them to our EAs.

```
54 void OnTimer()
55 {
56 //---
57     int pos = 0; // choose candle position
58
59     string legend = "Open price = "+DoubleToString(candle[pos].open,2)+"\n"+
60             "Max price = "+ DoubleToString(candle[pos].high,2)+"\n"+
61             "Close price = "+ DoubleToString(candle[pos].close,2)+"\n"+
62             "Min price = "+ DoubleToString(candle[pos].low,2)+"\n"+
63             "Last tick = "+ DoubleToString(tick.last,2)+"\n"+
64             "Time = " + tick.time;
65
66     Alert(legend);    ←
67 }
68 }
```



16. Adding MQL5 Indicators

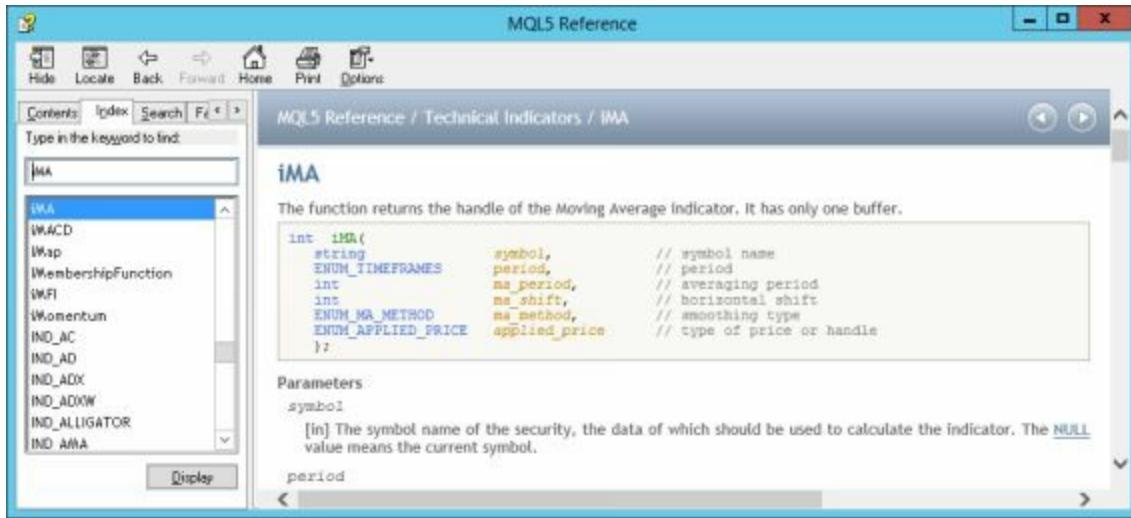
Let us now turn to one of the most common and important practices when the goal is to plan an investment strategy. It is very likely that your investment strategy will make use of some indicator, so let's learn how to call indicators from the MQL5 language.

Initially, we need to declare the variables for our indicator. How we will declare variables will depend greatly on the type and functioning of the indicator.

There are indicators that provide only one output of information, such as a moving average. From the moving average we have only one output of information which is the value of the moving average calculated for a specific point in the graph. In this case we say that the moving average has only one memory buffer. Every indicator needs at least one Handle and it must be an integer type.

```
12 //--- Moving Averages
13 int ma_Handle;           // Handle Moving Average
14
15 double ma_Buffer[]; // Buffer Moving Average
16
17 //+
18 //| Expert initialization function
19 //+
20 int OnInit()
21 {
22 //---
23   ma_Handle = iMA(_Symbol,_Period,7,0,MODE_EMA,PRICE_CLOSE);
24
25
26   if(ma_Handle<0)
27   {
28     Alert("Error trying to create Handles for indicator - error: ",GetLastError(),"!");
29     return(-1);
30   }
31
32   // To add the indicator to the chart:
33   ChartIndicatorAdd(0,0,ma_Handle);
34 //---
35   return(INIT_SUCCEEDED);
36 }
```

The declaration of the moving average indicator is given by a function of MQL5 (**iMA**) with a F1 in this function we obtain:



In our case we are using a moving average period of 7 with the simple smoothing type (MODE_SMA) applied to the closing price (PRICE_CLOSE). It is good to point out, once again, that each indicator has its own structure of parameters necessary for its proper functioning.

It is always good to put codes to evaluate possible variable loading errors. In our case we added an **Alert()** if we have problems with the variable 'mm_Handle'. We did this because the correct functioning of this variable is extremely important to our robot strategy. However, the practice of evaluating the loading of variables is always welcome.

The same procedure can be done for the **iRSI()** indicator.

We made the initial statement and addition of the indicator on the chart with the function **ChartIndicatorAdd(0,0,mm_Handle)**. Now we should point 'mm_Handle' to the vector of 'mm_Buffer[]' which is where we will access the values of our moving average.

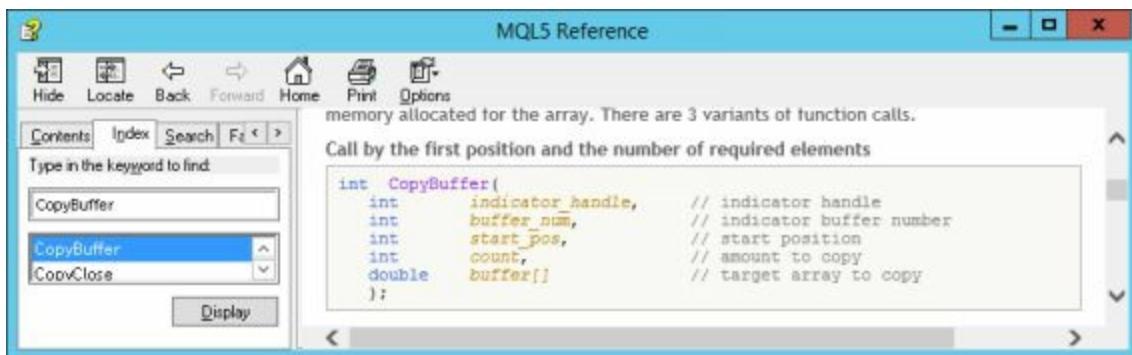
To do this within the **OnTick()** function we must call the **CopyBuffer()** function.

```

46 //+-----+
47 //| Expert tick function
48 //+-----+
49 void OnTick()
50 {
51 //---
52     // Copy a three-dimensional data vector to Buffer
53     CopyBuffer(ma_Handle,0,0,3,ma_Buffer);
54
55     // Sort the data vector:
56     ArraySetAsSeries(ma_fast_Buffer,true);
57
58 }

```

We can understand the structure of **CopyBuffer()** as follows:



Now let's add the **RSI** oscillator indicator. This indicator also has only a single Buffer. If you are in doubt regarding the number of buffers of each indicator simply go to the documentation from the **F1** key and look for **#property indicator_buffers**:

```

#property description "Todos os outros pa
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- desenhando iRSI
#property indicator_label1 "iRSI"
#property indicator_type1 DRAW_LINE

```

It follows the code with the addition of the two indicators (moving average and RSI). Note that the (iRSI ()) function of the RSI indicator has input parameters that are very different from the iMA () function:

```
12 //--- Moving Averages
13 int ma_Handle;           // Handle Moving Average
14
15 double ma_Buffer[];     // Buffer Moving Average
16
17 //--- RSI
18 int rsi_Handle;         // Handle for RSI
19
20 double rsi_Buffer[];    // Buffer for RSI
21
22 //+-----+
23 //| Expert initialization function
24 //+-----+
25 int OnInit()
26 {
27 //---
28     ma_Handle = iMA(_Symbol,_Period,7,0,MODE_EMA,PRICE_CLOSE);
29
30     rsi_Handle = iRSI(_Symbol,_Period,3,PRICE_CLOSE);
31
32     if(ma_Handle<0 || rsi_Handle<0)
33     {
34         Alert("Error trying to create Handles for indicator - error: ",GetLastError(),"!");
35         return(-1);
36     }
37
38     // To add the indicator to the chart:
39     ChartIndicatorAdd(0,0,ma_Handle);
40     ChartIndicatorAdd(0,1,rsi_Handle);
41 //---
42     return(INIT_SUCCEEDED);
43 }
```

```
44 //+-----+
45 //| Expert deinitialization function
46 //+-----+
47 void OnDeinit(const int reason)
48 {
49 //---
50     IndicatorRelease(ma_Handle);
51     IndicatorRelease(rsi_Handle);
52 }
```

```
53 //+
54 //| Expert tick function
55 //+
56 void OnTick()
57 {
58 //---
59     // Copy a three-dimensional data vector to Buffer
60     CopyBuffer(ma_Handle,0,0,3,ma_Buffer);
61     CopyBuffer(rsi_Handle,0,0,3,rsi_Buffer);
62
63     // Sort the data vector:
64     ArraySetAsSeries(ma_fast_Buffer,true);
65     ArraySetAsSeries(rsi_Buffer,true);
66
67     Print("ma_Buffer = ", ma_Buffer[0]);
68     Print("rsi_Buffer = ", rsi_Buffer[0]);
69     Print("-----");
70
71 }
```

Finally, we are fully equipped with the minimum information necessary to start programming our investment robot. Let's go to the next chapter!

Chapter 6

3. Programming the EA

Now that we know the main functionality of MetaTrader, we have already understood the basics of the features of the MQL5 language and the use of MetaEditor we are minimally prepared to start developing our investment robot (Expert Advisor - EA).

Before starting to program, let us know and understand the details of our EA strategy.

1.1. Strategy overview

Our EA will have two moving averages and the RSI indicator. The crossing of two Moving Averages works well when the market is in a trend. On the other hand, in parallel markets or in consolidation, the RSI indicator performs better.

There are several indicators that offer indicative when the market is in trend as the ADX indicator. We decided to choose the RSI for reasons of making an alternative between the possibility of choosing the market in the trend and/or lateralized market.

For the buy or sell operations, our EA will have the possibility to choose only one of the indicators (crossing of averages or RSI) or both at the same time. Thus, we will have an EA that will track the input triggers based on two indicators: moving averages and RSI.

Let us understand how the input triggers in operations will be given for each of these indicators.

1.1.1. Moving Averages

First we go to the trigger for the crossing of moving averages. We have two moving averages and the type, whether exponential or arithmetic, of these averages can be changed as we will see next.

One of the moving averages will be called the **fast average** and the other **slow average**. The fast average is the one that has the shortest period, on the other hand the slow average should always have the longest period.

We give these names to averages because fast averages are those with a shorter period of time and are more adherent to price, so they move faster than those of longer periods where they have slower directional behavior.

Let's look at an illustrative diagram of how these triggers work in trading:

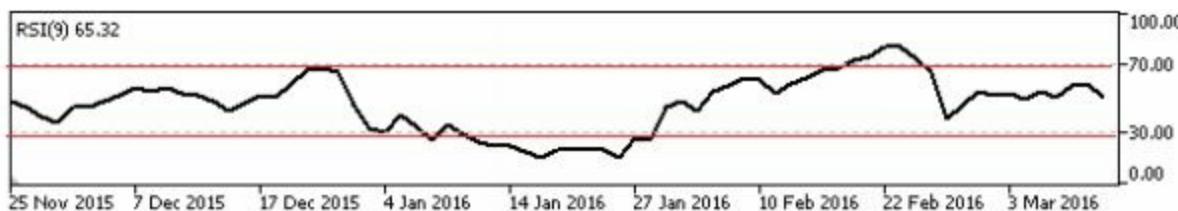


1.1.2. Relative Strength Index (RSI)

O indicador RSI (Relative Strength Index) no MetaTrader se refere a um Oscilador, ou seja, é útil para indicar posições de sobrecompra e sobrevenda em mercado lateralizado (consolidado).

We will not go into detail about the calculations behind the RSI because it is not the goal at the moment. But let's understand the kittens buy and sell this indicator

The figure below shows the RSI and two main levels (solid lines in red) at 30 and 70. Each of these lines represents the levels to allow **overbought** (line 70) and **oversold** (line 30). In other words, in a consolidated market, when the RSI is equal to or greater than line 70, it can be a reversal warning, warning an overbought, that is, it may be a good time to make a **sale**. On the other hand, if the RSI, once again in a consolidated market, is less than or equal to line 30, we are in an oversold region indicating a good time to **buy**.



In our EA we will leave those levels of 70 (overbought) and 30 (oversold) possible to be changed by the user. So you can choose more loose or restrictive levels for the kittens to buy and sell.

Let's now set an example where the trigger levels to buy are at **line 30** and the trigger to sell is at **line 70**:



Let us, therefore, to a summary of the strategy with the elements that we will need to program for our EA:

1 - We must add two moving averages (fast and slow) both with the possibility of choosing the period, the average application price and the average method used.

2 - We must add the RSI indicator with possibility of period choice, applied

price, and oversold and oversold levels.

3 - The user can choose among the options to enter the traders if they come from the moving averages plus the RSI, only moving averages or only the RSI.

Now that we know the operation to buy and sell for our two indicators (crossing of averages and RSI) we are ready to start developing the algorithm with the presented strategy.

1.2. Creating EA

Let's open MetaEditor and ask the wizard to create a new Expert Advisor. Let's give it the following name: MM_CROS_RSI.mql5.

If you want to obtain the complete source code of the robot you can access the following page in github <https://github.com/rafaelfvcs> or send an email to rafaelfvcs@gmail.com. However, it is strongly recommended that you enter the codes presented to gradually memorize the use of MetaEditor functions and features.

Here we have the skeleton of our EA:

```
1 //+-----+
2 //| MM_CROS_IFR.mq5 |
3 //| Copyright 2018. |
4 //| https://www.mql5.com |||
5 //+-----+
6 #property copyright "Copyright 2018."
7 #property link      "https://www.mql5.com"
8 #property version   "1.00"
9 //+-----+
10 //| Expert initialization function
11 //+-----+
12 int OnInit()
13 {
14 //---
15
16 //---
17     return(INIT_SUCCEEDED);
18 }
19 //+-----+
20 //| Expert deinitialization function
21 //+-----+
22 void OnDeinit(const int reason)
23 {
24 //---
25
26 }
27 //+-----+
28 //| Expert tick function
29 //+-----+
30 void OnTick()
31 {
32 //---
33
34 }
35 //+-----+
36
```

3. Declaration of Global Variables

Let's start by declaring our global variables. We will separate those variables that the user will have access to modify and those that will be used for the structure of the EA algorithm.

i.3.1. Variables for the User

We will provide the following options (input parameters): profit targets, stop loss, number of traded lots, chart time, average periods, RSI period, application of methods in prices (opening, closing, max, min) for both the averages and the RSI, possibility of choice in the strategy of entry triggers in operations (Moving averages plus RSI, only crosses, RSI only) and time limit to close the open operations.

Below is the code of all this:

```
19 // Variables Input
20 sinput string s0; //-----Strategy-----
21 input STRATEGY_IN strategy      = ONLY_MA;      // Trader Entry Strategy
22
23 sinput string s1; //-----Moving Averages-----
24 input int ma_fast_period      = 12;           // Fast Moving Average Period
25 input int ma_slow_period      = 32;           // Slow Moving Average Period
26 input ENUM_TIMEFRAMES ma_time_graphic = PERIOD_CURRENT; // Graphic Time
27 input ENUM_MA_METHOD ma_method     = MODE_EMA;      // Method
28 input ENUM_APPLIED_PRICE ma_price    = PRICE_CLOSE;   // Price Applied
29
30 sinput string s2; //-----RSI-----
31 input int rsi_period          = 5;            // RSI Period
32 input ENUM_TIMEFRAMES rsi_time_graphic = PERIOD_CURRENT; // Graphic Time
33 input ENUM_APPLIED_PRICE rsi_price     = PRICE_CLOSE;   // Price Applied
34
35 input int rsi_overbought      = 70;           // Level Overbought
36 input int rsi_oversold        = 30;           // Level Oversold
37
38 sinput string s3; //-----
39 input int num_lots            = 100;          // Number of lots
40 input double TK                = 60;           // Take Profit
41 input double SL                = 30;           // Stop Loss
42
43 sinput string s4; //-----
44 input string limit_close_op    = "17:40";       // Time Limit Close Position
```

For the 'strategy' variable it was chosen to create an enum (this should be

declared at the top of the algorithm above the **Input** variables):

```
11 enum STRATEGY_IN
12 {
13     ONLY_MA,      // Only moving averages
14     ONLY_RSI,     // Only RSI
15     MA_AND_RSI   // moving averages plus RSI
16 };
```

3.3.2. Global Variables

We must now declare the global variables that will be used by the functions of our algorithm. See below:

```
46 //+-----
47 //|  Variables for indicators
48 //+-----
49 //--- Moving Averages
50 // FAST - shorter period
51 int ma_fast_Handle;           // Handle Fast Moving Average
52 double ma_fast_Buffer[];      // Buffer Fast Moving Average
53
54 // SLOW - longer period
55 int ma_slow_Handle;          // Handle Slow Moving Average
56 double ma_slow_Buffer[];      // Buffer Slow Moving Average
57
58 //--- RSI
59 int rsi_Handle;              // Handle for RSI
60 double rsi_Buffer[];          // Buffer for RSI
61
62 //+-----
63 //|  Variables for functions
64 //+-----
65
66 int magic_number = 123456;    // Magic Number
67
68 MqlRates candle[];           // Variable for storing candles
69 MqlTick tick;                // Variable for storing ticks
70
```

4. OnInit () Function

In the **OnInit()** function we will add the initialization variables to the indicators and add them to the chart with the **ChartIndicatorAdd()** function. Follow the code:

```
71 //+
72 //| Expert initialization function
73 //+-----+
74 int OnInit()
75 {
76 //---
77     ma_fast_Handle = iMA(_Symbol,ma_time_graphic,ma_fast_period,0,ma_method,ma_price);
78     ma_slow_Handle = iMA(_Symbol,ma_time_graphic,ma_slow_period,0,ma_method,ma_price);
79
80     rsi_Handle = iRSI(_Symbol,rsi_time_graphic,rsi_period,rsi_price);
81
82     if(ma_fast_Handle<0 || ma_slow_Handle<0 || rsi_Handle<0)
83     {
84         Alert("Error trying to create Handles for indicator - error: ",GetLastError(),"!");
85         return(-1);
86     }
87
88     CopyRates(_Symbol,_Period,0,4,candle);
89     ArraySetAsSeries(candle,true);
90
91     // To add the indicator to the chart:
92     ChartIndicatorAdd(0,0,ma_fast_Handle);
93     ChartIndicatorAdd(0,0,ma_slow_Handle);
94     ChartIndicatorAdd(0,1,rsi_Handle);
95     //---
96
97 //---
98     return(INIT_SUCCEEDED);
99 }
```

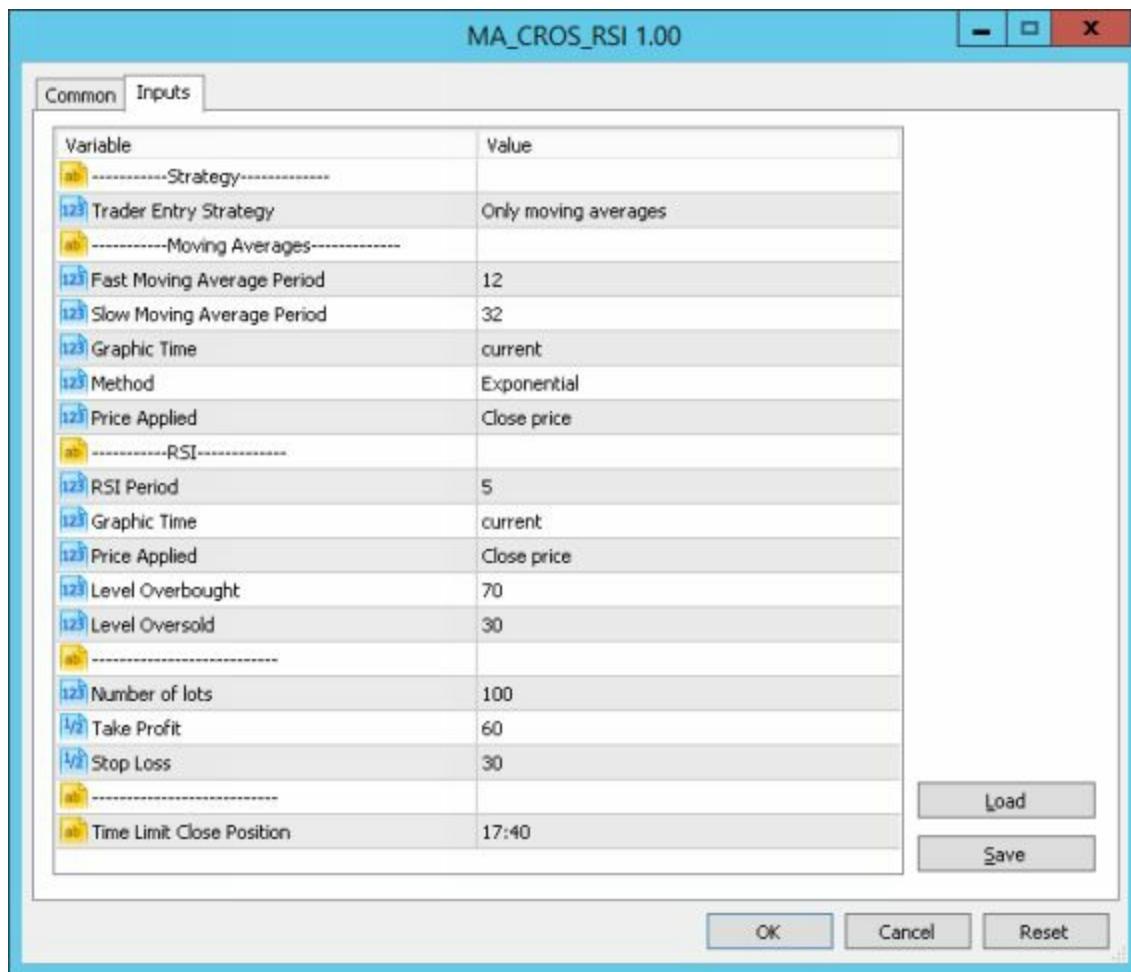
Note that **ChartIndicatorAdd(0,0,mm_fast_Handle)** and **ChartIndicatorAdd(0,0, mm_slow_Handle)** has the same graphical addition reference (0 zero), which means that both indicators will be on the main screen where the candles . However, **ChartIndicatorAdd(0,1,ifr_Handle)** terms the reference (1 one) ie the bookmark will be added to another chart box.

5.5. Function OnDeinit()

When the EA is removed we need to remove the indicators and if necessary clear variables and parameters. The **OnDeinit()** function is called when the robot is removed. Thus, within this function we can write our disinitialization activities of the parameters:

```
100 //+-----+
101 //| Expert deinitialization function
102 //+-----+
103 void OnDeinit(const int reason)
104 {
105 //---
106     IndicatorRelease(ma_fast_Handle);
107     IndicatorRelease(ma_slow_Handle);
108     IndicatorRelease(rsi_Handle);
109 }
```

As we write these codes it is advisable to compile and test EA in MetaTrader. So if we drag the file: MM_CROS_RSI.exe5 in MetaTrader for some financial asset we will have the following:



Clicking on ok will add the indicators to the chart and the EA will appear in the upper right corner or not if the automated trading button is on.

See the example below:



6. Expert Advisor Functions

Let's create some useful and important functions for the operation of our EA. We can use the shortcut (Ctrl +.) To create sections for a set of functions that we will create for our EA.

```
15 //+-----+
16 //| FUNCTIONS TO ASSIST IN THE VISUALIZATION OF THE STRATEGY |
17 //+-----+
18
19 //+-----+
20 //| FUNCTIONS FOR SENDING ORDERS |
21 //+-----+
22
23 //+-----+
24 //| USEFUL FUNCTIONS |
25 //+-----+
```

First, we will need a function that informs the appearance of a new candle regardless of the graphical time. Still, we do not have a native function in MQL5 that allows us to accurately assess changing candle and so we must create it.

The function should return true if a new candle appears on the chart. Here is the code used in the tutorials in the MetaTrader article portal:

```

354 //+
355 //| USEFUL FUNCTIONS
356 //+
357 //--- for bar change
358 bool isNewBar()
359 {
360 //--- memorize the time of opening of the last bar in the static variable
361     static datetime last_time=0;
362 //--- current time
363     datetime lastbar_time= (datetime) SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DATE);
364
365 //--- if it is the first call of the function
366     if(last_time==0)
367     {
368         //--- set the time and exit
369         last_time=lastbar_time;
370         return(false);
371     }
372
373 //--- if the time differs
374     if(last_time!=lastbar_time)
375     {
376         //--- memorize the time and return true
377         last_time=lastbar_time;
378         return(true);
379     }
380 //--- if we passed to this line, then the bar is not new; return false
381     return(false);
382 }

```

Let's do a function to help you visualize the entry points of our strategies. This function basically will construct a graphical object (**vertical line**). Every time our robot presents logical conditions (the trigger is activated) a vertical line will be drawn on the graph with a name and color referring to the type of warning.

```

207 //+
208 //| FUNCTIONS TO ASSIST IN THE VISUALIZATION OF THE STRATEGY
209 //+
210
211 void drawVerticalLine(string name, datetime dt, color cor = clrAliceBlue)
212 {
213     ObjectDelete(0,name);
214     ObjectCreate(0,name,OBJ_VLINE,0,dt,0);
215     ObjectSetInteger(0,name,OBJPROP_COLOR,cor);
216 }

```

The function **drawsLineVertical()** has four input variables. MQL5 allows the last input variables of a function to already come with a default value. See the case of the variable 'color = clrAliceBlue'. This means that when the user does not set a specific color the 'clrAliceBlue' will automatically be assigned to the variable. Therefore, the user has no obligation to assign values to this type of variables. However it is mandatory to enter values for the other

variables.

Now let's create one of the most important functions for our EA. They are the functions of sending orders.

We need to send orders to buy or sell every time a trigger is started. In MQL5 we can do this type of sending in different ways. We can use a specific class called **CTrade** to work with the handling and sending of orders. However, here we choose to use the requisite assembly structures (**MqlTradeRequest**) and send response (**MqlTradeResult**).

MqlTradeRequest is used to perform all trade requests for a trade operation. It contains all the fields necessary for performing a trade deal.

```
struct MqlTradeRequest
{
    ENUM_TRADE_REQUEST_ACTIONS      action;           // Trade operation type
    ulong                           magic;            // Expert Advisor ID (magic number)
    ulong                           order;             // Order ticket
    string                          symbol;            // Trade symbol
    double                          volume;            // Requested volume for a deal in lots
    double                          price;             // Price
    double                          stoplimit;         // StopLimit level of the order
    double                          sl;                // Stop Loss level of the order
    double                          tp;                // Take Profit level of the order
    ulong                           deviation;         // Maximal possible deviation from the requested price
    ENUM_ORDER_TYPE                 type;              // Order type
    ENUM_ORDER_TYPE_FILLING         type_filling;     // Order execution type
    ENUM_ORDER_TYPE_TIME            type_time;        // Order execution time
    datetime                        expiration;       // Order expiration time (for the orders of ORDER_TIME_SPECIFIED type)
    string                          comment;           // Order comment
};
```

MqlTradeResult is used for any trade operation. The variable declared to be of this type will be able to access the trade request results.

```
struct MqlTradeResult
{
    uint    retcode;          // Operation return code
    ulong   deal;             // Deal ticket, if it is performed
    ulong   order;            // Order ticket, if it is placed
    double  volume;           // Deal volume, confirmed by broker
    double  price;            // Deal price, confirmed by broker
    double  bid;              // Current Bid price
    double  ask;              // Current Ask price
    string  comment;          // Broker comment to operation (by default it is filled by the operation description)
};
```

From these structures we can delimit the type of order that we are interested in, specifying details of it. The request and response variables must be used in an order send function called **OrderSend()**.

Below is a function created to send an order to buy to market with type of fill FOK (fill or kill - or fills the lot size of the order or does not execute). This order, executed to market, already enters with the stop loss and target of gains (take profit) defined.

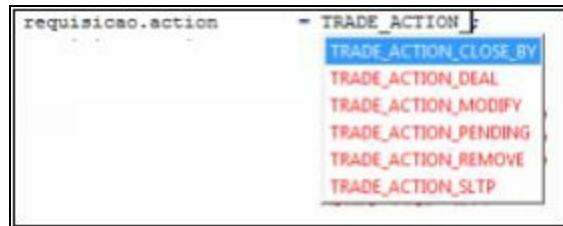
In every market there are offers from the best buyers and best sellers. Therefore, we need to access the book of offers to know these prices. As we have seen, in Chapter 5, we can access book information from the **MqlTick** structure. So we created a variable called 'tick' to store this information. With it we can access the best seller 'tick.bid' and the best buyer 'tick.ask'.

We have the bid = (sell, offer) and ask = (buy, demand).

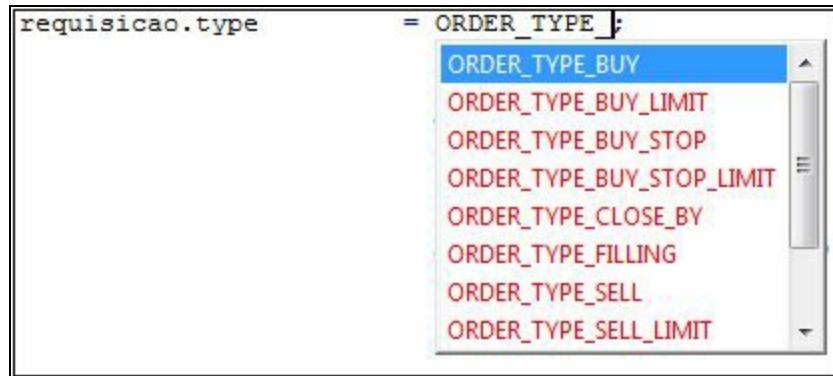
```

222 // BUY TO MARKET
223 void BuyAtMarket()
224 {
225     MqlTradeRequest    request;          // request
226     MqlTradeResult     response;         // response
227
228     ZeroMemory(request);
229     ZeroMemory(response);
230
231     //--- For Buy Order
232     request.action      = TRADE_ACTION_DEAL;           // Trade operation type
233     request.magic       = magic_number;                // Magic number
234     request.symbol      = _Symbol;                     // Trade symbol
235     request.volume      = num_lots;                   // Lots number
236     request.price       = NormalizeDouble(tick.ask,_Digits); // Price to buy
237     request.sl          = NormalizeDouble(tick.ask - SL*_Point,_Digits); // Stop Loss Price
238     request.tp          = NormalizeDouble(tick.ask + TK*_Point,_Digits); // Take Profit
239     request.deviation   = 0;                          // Maximal possible deviation
240     request.type        = ORDER_TYPE_BUY;              // Order type
241     request.type_filling = ORDER_FILLING_FOK;        // Order execution type
242
243     //---
244     OrderSend(request,response);
245     //---
246     if(response.retcode == 10008 || response.retcode == 10009)
247     {
248         Print("Order Buy executed successfully!!!");
249     }
250     else
251     {
252         Print("Error sending Order to Buy. Error = ", GetLastError());
253         ResetLastError();
254     }
255 }
```

The type of action to be chosen is the first requisition that we should assign:



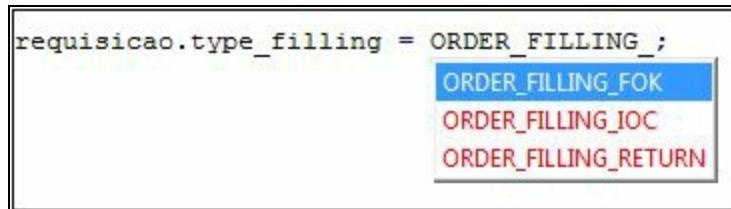
We have several types of orders. We can choose from the structure type:



The **NormalizeDouble()** function is used for rounding variables. We need to use it because some financial assets have the number of different price digits of others. We have assets that are quantified by points (as is the case of index and dollar futures contracts), others are quantified by price and even pips (Forex market). In Chapter 5 we discussed the use of **_Point** and **_Digits**.

The values of the SL and TK variables should be chosen by the user in the form of points. Notice that for the stop loss price we did: **NormalizeDouble(tick.ask - SL * _Point, _Digits)** as we are buying and our stop loss point needs to be below the (ask). Already the target of the take (take profit) naturally for a purchase should be above TK ask points.

It is with the request `request.type_filling = ORDER_FILLING_FOK'` that we choose the order fulfillment type. We basically have three types of order fulfillment:



Identifier	Description
ORDER_FILLING_FOK	This filling policy means that an order can be filled only in the specified amount. If the necessary amount of a financial instrument is currently unavailable in the market, the order will not be executed. The required volume can be filled using several offers available on the market at the moment.
ORDER_FILLING_IOC	This mode means that a trader agrees to execute a deal with the volume maximally available in the market within that indicated in the order. In case the the entire volume of an order cannot be filled, the available volume of it will be filled, and the remaining volume will be canceled.
ORDER_FILLING_RETURN	This policy is used only for market orders (ORDER_TYPE_BUY and ORDER_TYPE_SELL), limit and stop limit orders (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT and ORDER_TYPE_SELL_STOP_LIMIT) and only for the symbols with Market or Exchange execution. In case of partial filling a market or limit order with remaining volume is not canceled but processed further. For the activation of the ORDER_TYPE_BUY_STOP_LIMIT and ORDER_TYPE_SELL_STOP_LIMIT orders, a corresponding limit order ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT with the ORDER_FILLING_RETURN execution type is created.

Let's see the function to sell:

```

257 // SELL TO MARKET
258 void SellAtMarket()
259 {
260     MqlTradeRequest    request;      // request
261     MqlTradeResult     response;     // response
262
263     ZeroMemory(request);
264     ZeroMemory(response);
265
266     //--- For Sell Order
267     request.action      = TRADE_ACTION_SELL;           // Trade operation type
268     request.magic        = magic_number;                // Magic number
269     request.symbol       = _Symbol;                     // Trade symbol
270     request.volume       = num_lots;                    // Lots number
271     request.price        = NormalizeDouble(tick.bid,_Digits); // Price to sell
272     request.sl           = NormalizeDouble(tick.bid + SL*_Point,_Digits); // Stop Loss Price
273     request.tp           = NormalizeDouble(tick.bid - TK*_Point,_Digits); // Take Profit
274     request.deviation   = 0;                          // Maximal possible deviation
275     request.type         = ORDER_TYPE_SELL;            // Order type
276     request.type_filling = ORDER_FILLING_FOK;          // Order execution type
277
278     OrderSend(request,response);
279
280     if(response.retcode == 10008 || response.retcode == 10009)
281     {
282         Print("Order to Sell executed successfully!");
283     }
284     else
285     {
286         Print("Error sending Order to Sell. Error =", GetLastError());
287         ResetLastError();
288     }
289 }
```

In this case, we have the stop loss level above the bid price **NormalizeDouble**(`tick.bid + SL * _Point, _Digits`) and the gain target naturally below **NormalizeDouble** (`tick.bid - TK * _Point, _Digits`).

We also need to create two functions to close open orders if arrive at the **time limit** chosen by the user. That's because our robot is **daytrader**, we do not want to sleep positioned. Remembering that to guarantee the closing of open orders we have the variable 'Limit Closing Closing Position' chosen by the

user.

So to close a sale we must buy and to close a purchase we must sell.

```
292 void CloseBuy()
293 {
294     MqlTradeRequest    request;      // request
295     MqlTradeResult     response;     // response
296
297     ZeroMemory(request);
298     ZeroMemory(response);
299
300     //--- For Sell Order
301     request.action      = TRADE_ACTION_DEAL;
302     request.magic        = magic_number;
303     request.symbol       = _Symbol;
304     request.volume       = num_lots;
305     request.price        = 0;
306     request.type         = ORDER_TYPE_SELL;
307     request.type_filling = ORDER_FILLING_FOK;
308
309     //---
310     OrderSend(request,response);
311     //---
312     if(response.retcode == 10008 || response.retcode == 10009)
313     {
314         Print("Order to Sell executed successfully!");
315     }
316     else
317     {
318         Print("Error sending Order to Sell. Error =", GetLastError());
319         ResetLastError();
320     }
321 }
```

We can note that for the **CloseBuy()** function the `request.type_filling = ORDER_FILLING_RETURN`, this is because we want to close the order independent of the best buy or sell. As we know this type of fill (ORDER_FILLING_RETURN) executes the order until it reaches the limit of the number of lots.

```

323 void CloseSell()
324 {
325     MqlTradeRequest    request;          // request
326     MqlTradeResult     response;         // response
327
328     ZeroMemory(request);
329     ZeroMemory(response);
330
331     //--- For Buy Order
332     request.action      = TRADE_ACTION_DEAL;
333     request.magic       = magic_number;
334     request.symbol      = _Symbol;
335     request.volume      = num_lots;
336     request.price       = 0;
337     request.type        = ORDER_TYPE_BUY;
338     request.type_filling = ORDER_FILLING_RETURN;
339
340     //---
341     OrderSend(request,response);
342
343     //---
344     if(response.retcode == 10008 || response.retcode == 10009)
345     {
346         Print("Order Buy executed successfully!!!");
347     }
348     else
349     {
350         Print("Error sending Order to Buy. Error = ", GetLastError());
351         ResetLastError();
352     }
353 }

```

6.1.Error Handling

It is always good to treat the returns of the orders sent. This is because we can minimize problems and evaluate execution errors. In our functions we did this from answer.retcode which returns a code (with 5 digits) coming from the brokerage server. Below is a window showing details of these codes. You can get to this window by pressing the F1 key on code number 10008 or 10009.

MQL5 Reference

Hide Locate Back Forward Home Print Options

Contents Index Search F4 < >

Type in the keyword to find:

10008

10008
10009
10010
10011
10012
10013
10014
10015
10016
10017
10018
10019
10020
10021
10022

Display

MQ5 Reference / Standard Constants, Enumerations and Structures / Codes of Errors and Warnings / Trade Server Return Codes

Return Codes of the Trade Server

All requests to execute trade operations are sent as a structure of a trade request `MqlTradeRequest` using function `OrderSend()`. The function execution result is placed to structure `MqlTradeResult`, whose `retcode` field contains the trade server return code.

Code	Constant	Description
10004	TRADE_RETCODE_QUOTE	Requote
10006	TRADE_RETCODE_REJECT	Request rejected
10007	TRADE_RETCODE_CANCEL	Request canceled by trader
10008	TRADE_RETCODE_PLACED	Order placed
10009	TRADE_RETCODE_DONE	Request completed
10010	TRADE_RETCODE_DONE_PARTIAL	Only part of the request was completed
10011	TRADE_RETCODE_ERROR	Request processing error
10012	TRADE_RETCODE_TIMEOUT	Request canceled by timeout
10013	TRADE_RETCODE_INVALID	Invalid request

1.7. Function: OnTick()

The goal now is to create the whole strategy logic for our EA. We already have the necessary functions ready for its operation. It is within the **OnTick()** function, called each new business carried out on the stock exchange, that we will write the strategic body.

First, we must feed our candle, tick, and indicator variables with data. Also, let's not forget to sort them appropriately with the **ArraySetAsSeries()** function:

```
116 // Copy a three-dimensional data vector to Buffer
117 CopyBuffer(ma_fast_Handle,0,0,4,ma_fast_Buffer);
118 CopyBuffer(ma_slow_Handle,0,0,4,ma_slow_Buffer);
119
120 CopyBuffer(rsi_Handle,0,0,4,rsi_Buffer);
121
122 //--- Feed candle buffers with data:
123 CopyRates(_Symbol,_Period,0,4,candle);
124 ArraySetAsSeries(candle,true);
125
126 // Sort the data vector:
127 ArraySetAsSeries(ma_fast_Buffer,true);
128 ArraySetAsSeries(ma_slow_Buffer,true);
129 ArraySetAsSeries(rsi_Buffer,true);
130 //---
131
132 // Feed with tick variable data
133 SymbolInfoTick(_Symbol,tick);
134
```

After this we can make use of these variables to build the logic of the buy and sell action of our EA. Follow the conditions of crossings:

Crossings conditions

R — Fast average

L — Slow average



$R[0] > L[0] \&& R[2] < L[2]$



$L[0] > R[0] \&& L[2] < R[2]$

In the MQL5 language it is:

```

135 // LOGIC TO ACTIVATE PURCHASE
136 bool buy_ma_cros = ma_fast_Buffer[0] > ma_slow_Buffer[0] &&
137             ma_fast_Buffer[2] < ma_slow_Buffer[2] ;
138
139 bool buy_rsi = rsi_Buffer[0] <= rsi_oversold;
140
141 // LOGIC TO ACTIVATE SALE
142 bool sell_ma_cros = ma_slow_Buffer[0] > ma_fast_Buffer[0] &&
143             ma_slow_Buffer[2] < ma_fast_Buffer[2];
144
145 bool sell_rsi = rsi_Buffer[0] >= rsi_overbought;
```

We have to create a logic to use the enum (STRATEGY_IN). We need the sales and purchase triggers to be oriented with the options of: ONLY_MM (mobile averages only), ONLY_RSI (RSI only) and MA_AND_RSI (Moving averages plus RSI) described in this enum. Here's a possible solution to this problem:

```
148     bool Buy = false; // Can Buy?
149     bool Sell = false; // Can Sell?
150
151     if(strategy == ONLY_MA)
152     {
153         Buy = buy_ma_cros;
154         Sell = sell_ma_cros;
155     }
156     else if(strategy == ONLY_RSI)
157     {
158         Buy = buy_rsi;
159         Sell = sell_rsi;
160     }
161     else
162     {
163         Buy = buy_ma_cros && buy_rsi;
164         Sell = sell_ma_cros && sell_rsi;
165     }
166 }
```

Now we can organize our functions so EA can make buying and selling decisions. It is important to know when a new bar (candle) is created so that the algorithm can track, without much computational effort, a possible crossing of averages and RSI values in the desired region. Let's draw a vertical line when the buy/sell triggers are activated.

```

168     // returns true if we have a new candle
169     bool newBar = isNewBar();
170
171     // Every time there is a new candle enter this 'if'
172     if(newBar)
173     {
174
175         // Buy Condition:
176         if(Buy && PositionSelect(_Symbol)==false)
177         {
178             drawVerticalLine("Buy",candle[1].time,clrBlue);
179             BuyAtMarket();
180         }
181
182         // Sell Condition:
183         if(Sell && PositionSelect(_Symbol)==false)
184         {
185             drawVerticalLine("Sell",candle[1].time,clrRed);
186             SellAtMarket();
187         }
188
189     }

```

The '**PositionSelect(_Symbol)**' function returns a boolean. If there is any position in progress it returns true, otherwise false. So we just want to get into an operation if the EA has no open position.

Finally, to end our long-awaited EA we need a code to close some open position at the time limit, i.e., that position that did not reach the winning target or the stop loss.

```

191     if(newBar && TimeToString(TimeCurrent(),TIME_MINUTES) == limit_Close_OP && PositionSelect(_Symbol)==true)
192     {
193         Print("-----> End of Operating Time: End Open Positions!");
194         drawVerticalLine("Limit_OP",candle[0].time,clrYellow);
195
196         if(PositionGetInteger(POSITION_TYPE) == POSITION_TYPE_BUY)
197         {
198             CloseBuy();
199         }
200         else if(PositionGetInteger(POSITION_TYPE) == POSITION_TYPE_SELL)
201         {
202             CloseSell();
203         }
204     }

```

The **PositionGetInteger(POSITION_TYPE)** function returns the open position type. If it is a buy position we have **POSITION_TYPE_BUY** in the case of a sale we have **POSITION_TYPE_SELL**.

Once again if you want to have access to the complete code of this EA can be downloaded at: <https://github.com/rafaelfvcs>. Or if you prefer, you can send an email to rafaelfvcs@gmail.com and I'll be happy to send it to you!

Let's now do backtests with our newest multi-strategy investment EA.

Chapter 7

7. Backtests

Our main interest with Expert Advisor (EA) is to have a winning strategy over the long term. There is that famous phrase: that nobody in the financial market has a crystal ball. It is no use trying to predict the future of the market, because unfortunately, it is impossible until the present moment.

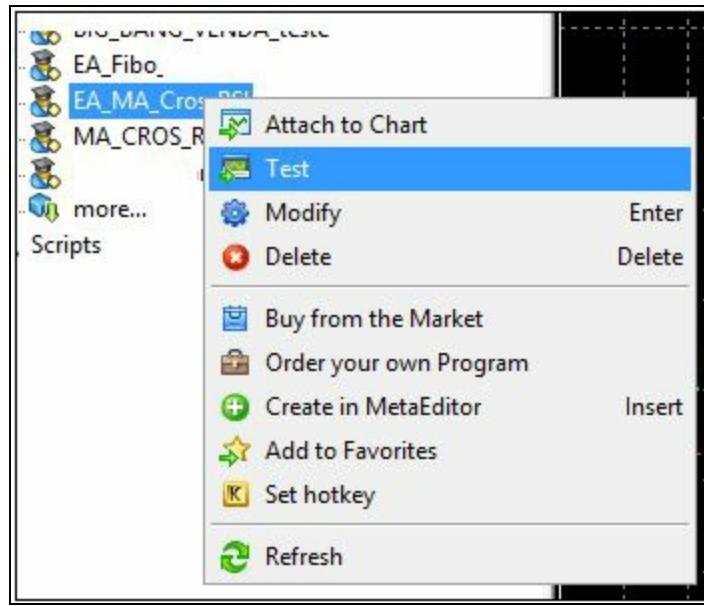
We need to be realistic to accept that our strategy will have ups and downs regarding financial returns. The 100% winning strategy is a utopia. However, it is important to know how good our strategy is. There is nothing better than using past data to gauge how it performed under various market conditions. So testing our strategy at past time periods is called a backtest.

Fortunately, when it comes to backtesting MetaTrader 5 provides a collection of simple and useful features to extract as much information as possible from the performance of our EA.

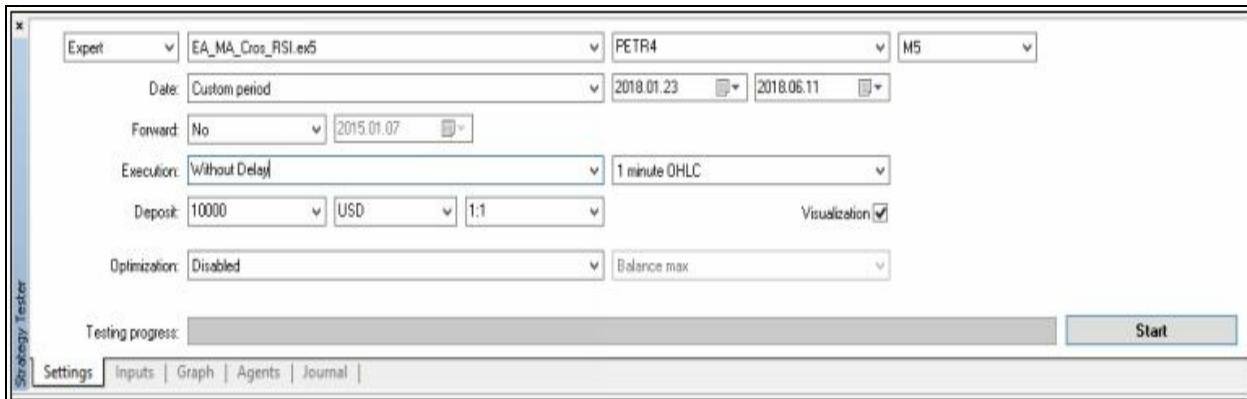
Now that we already have an EA (MM_CROS_RSI.exe5) multi-strategy programmed and working properly, let's do the backtests.

'1. Backtest in MetaTrader 5

Let's open MetaTrader and go to the field **Navigator -> Expert Advisor** consultant and look for our newest EA. Once found we should right click on it and then on **Test**.

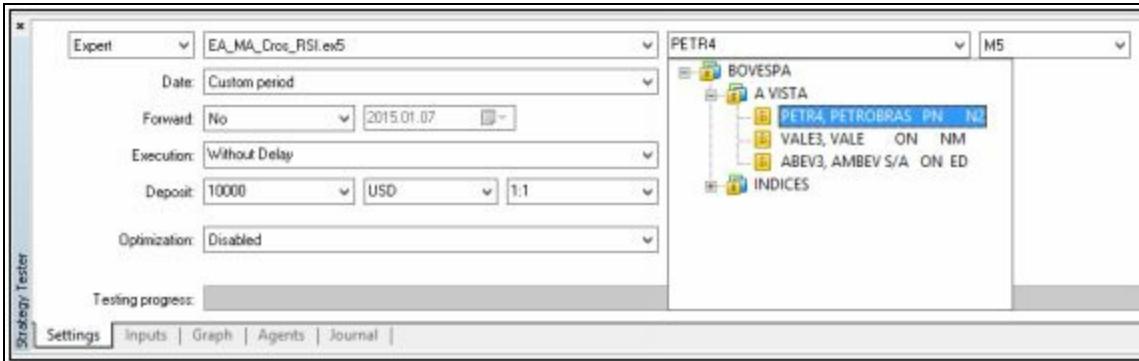


This will give us a new field in MetaTrader called **Strategy Tester**, in the settings tab we have the following panel of possibilities:

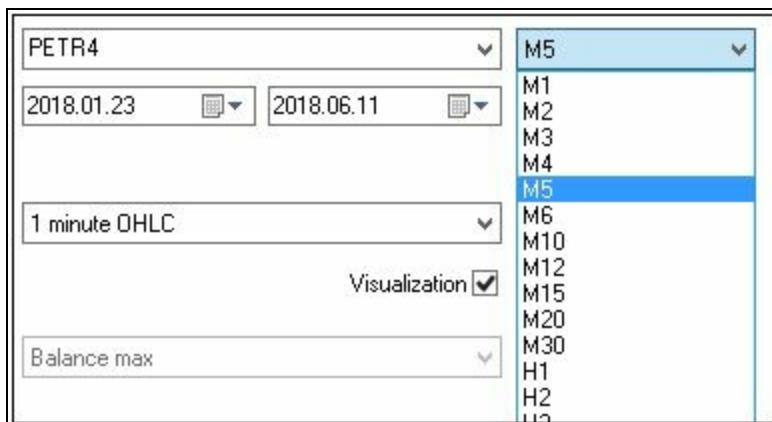


With these features we will be able to choose the financial asset for the

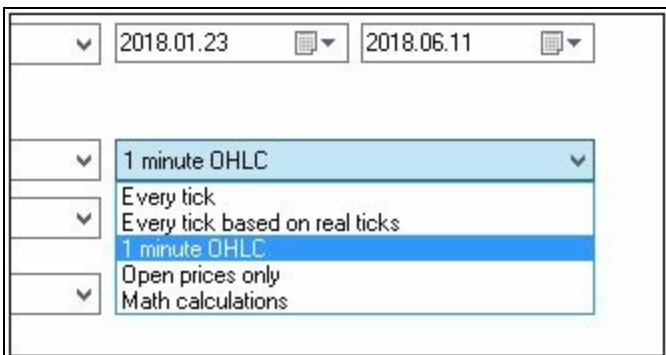
backtest:



The graphical time:



We have the option to choose how the **formation of our candles** are simulated by MetaTrader.



For this way of generating prices in candles, let's go to the main options most

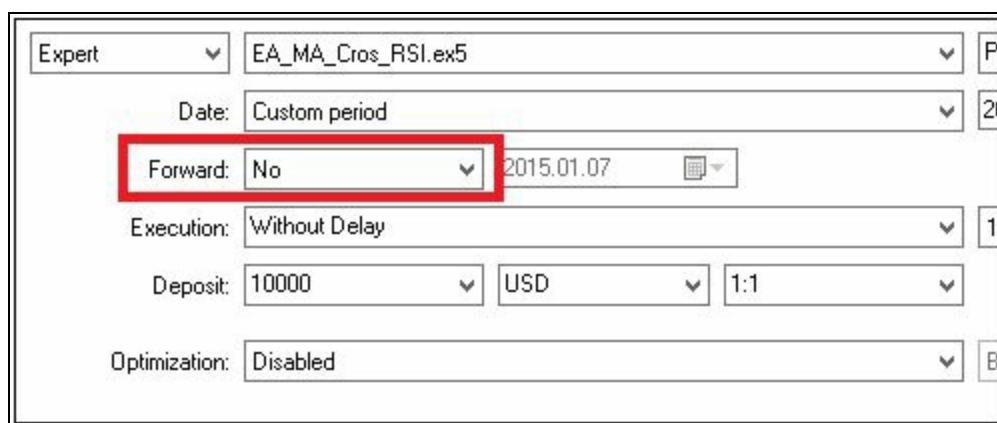
used:

Every tick or Every tick based on real ticks: MetaTrader tries to simulate real price ticks in the formation of candles. This option leaves the backtest more accurate, but requires more computational resources once inside a candle the MetaTrader tries to reproduce the formation of real ticks. It is worth noting that ticks are not real are just a simulation (a possible representation of reality). This article clarifies the functioning of this simulation:<https://www.mql5.com/en/articles/75>.

1 minute OHLC: In this type of candle formation MetaTrader generates only 4 price values: **O** - open, **H** - High, **L** - Low, **C** - Close. This option requires much fewer computational resources compared to the tick option.

Open price only: This is the least costly option computationally, because it only uses a price to represent a candle.

In our backtests we will always leave the option **Forward** marked **No** (see the figure below). This option is useful when we are doing strategy optimizations and want to use optimization methods such as **Walk Forward Analysis** (WFA). This type of study is beyond the scope of this book. If you want to know more about it, you can buy the other e-book ([How to Properly Optimize Investment Robots in MetaTrader 5](#)^[4]).



The type of **execution** with or without **Delay** is interesting so we can simulate the conditions in which the servers are and the speed of our internet. Latency refers to the time required for routing orders to be properly executed.

This screenshot shows a backtesting configuration window. The 'Execution' field is set to 'Without Delay'. Below it, the 'Deposit' field is set to 27 ms (last ping to your server is 27.19 ms). The 'Optimization' dropdown menu is open, showing options like 1 ms, 5 ms, 10 ms, etc., with '1:1' selected. The 'Testing progress' bar is at 0%.

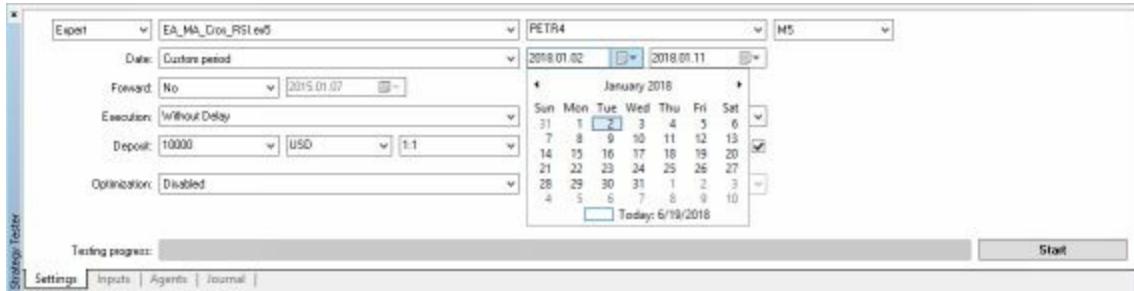
We have the possibility to choose the financial from the initial deposit as well as the possibility to leverage investments (in the case of the figure below 1: 1 - without leverage).

This screenshot shows a backtesting configuration window. The 'Execution' field is set to 'Without Delay'. The 'Deposit' field is set to 10000 USD. The 'Optimization' dropdown menu is open, showing options like 1:1, 1:50, 1:100, etc., with '1:1' selected. The 'Testing progress' bar is at 0%.

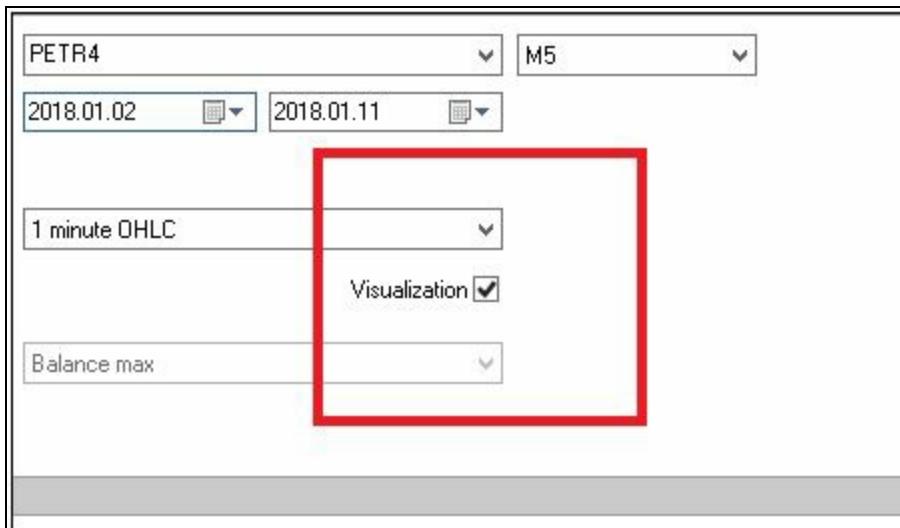
For our backtests the **Optimization** field should be disabled:

This screenshot shows a backtesting configuration window. The 'Execution' field is set to 'Without Delay'. The 'Deposit' field is set to 10000 USD. The 'Optimization' dropdown menu is open, showing options like 1:1, 1:50, 1:100, etc., with 'Disabled' selected. The 'Testing progress' bar is at 0%.

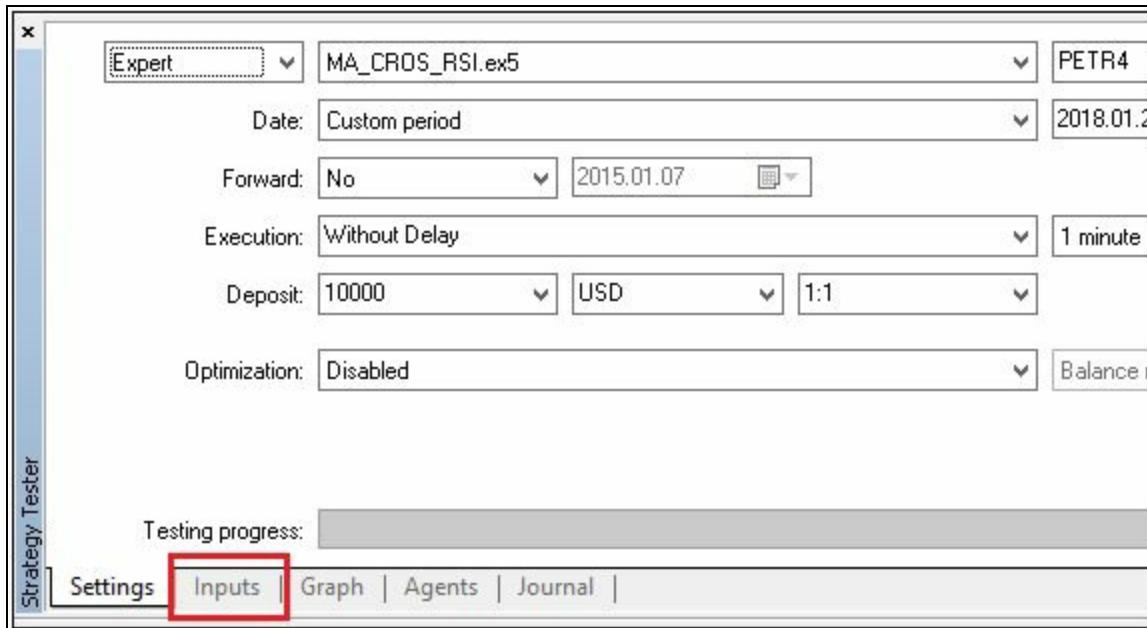
We should choose the data period for the backtest.



And check the option to view the backtest:



Let's now go to the **Input** (Parameters) tab of the Strategy Tester.



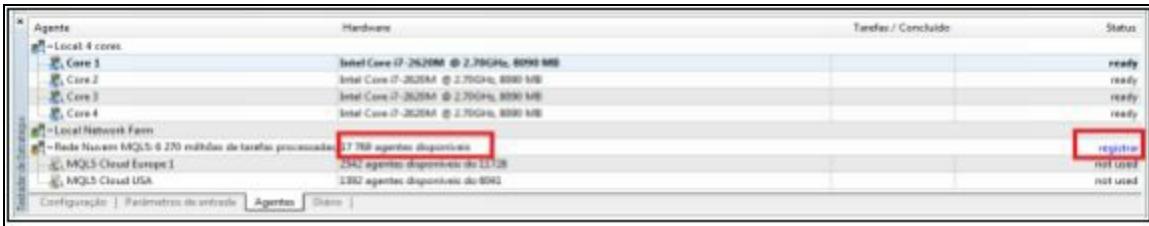
In this field we can put the setup necessary for the realization of our backtests. At the moment, what interests us is just the Value column, highlighted in low red. The other columns (highlighted in green) are used as parameters for EA optimization.

The screenshot shows the 'Inputs' tab with a table of parameters. The 'Value' column is highlighted in red, and the 'Start', 'Step', and 'Stop' columns are highlighted in green. The table includes rows for Moving Averages and RSI parameters.

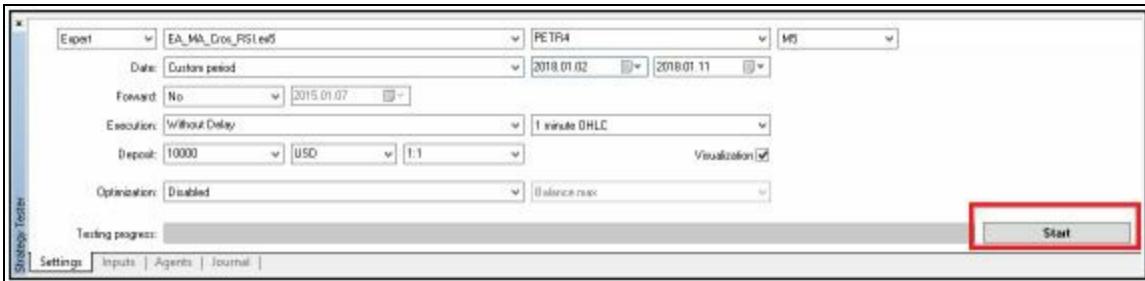
Variable	Value	Start	Step	Stop	Steps
Strategy	Only moving averages	Only moving averages		moving averages plus ..	
Trader Entry Strategy					
Moving Averages					
Fast Moving Average Period	12	12	1	120	
Slow Moving Average Period	32	32	1	320	
Graphic Time	current	current		1 Month	
Method	Exponential	Simple		Linear weighted	
Price Applied	Close price	Close price		Weighted price	
RSI					
RSI Period	5	5	1	50	
Graphic Time	current	current		1 Month	
Price Applied	Close price	Close price		Weighted price	
Level Overbought	70	70	1	700	
Level Oversold	30	30	1	300	

In the **Agents** tab we have information regarding the computational power of the machine that we are using to do the backtests and optimizations. A backtest is usually computationally inexpensive. However, when we are doing optimizations (many backtests are required) we will need a lot of resources and therefore this **Agents** tab will be important to us.

We can make a log to use multiple processing cores in the cloud and thus increase our computing power. The use of these extra cores is very affordable. With a few dollars we can make hundreds of thousands of optimizations.



After setting the parameters correctly for the backtest, simply go back to the **Settings** tab and hit the **Start** button.



'2. Analyzing the Backtest

As soon as you press the **Start** button, the following window appears:



So we have a screen to visualize what is happening with the backtest. We can control the speed with which new candles (candles) appear, from the highlighted buttons (red) of the figure above.

You can also have the prices of each candle and indicator (green area highlighted), just place the mouse over the desired element on the screen.

In the Toolbox we have four tabs (Trading, History, Operations and Journal) where we can track records regarding our trades operations of our EA.



In the **Toolbox** in the **Operations** tab we can follow the records with each of the sales and purchase operations. Something very useful is that we can give two clicks on some of this operation and the graph will automatically show the operation in question.



Ticket	Ativo	Ação	Tipo	Volume	Preço	S / L	T / P	Comentário
2018.01.08 11:19:00	2	petr4	exchange	buy	180.00	16.82	16.77	17.00
2018.01.08 17:49:00	3	petr4	exchange	sell	180.00	16.88		
2018.01.11 12:29:00	4	petr4	exchange	buy	180.00	17.01	16.96	17.21
2018.01.11 10:29:00	6	petr4	exchange	buy	180.00	16.81	16.86	17.11
2018.01.11 12:45:00	8	petr4	exchange	buy	180.00	16.86	16.83	17.08
2018.01.12 17:25:00	10	petr4	exchange	buy	180.00	17.27	17.22	17.47
2018.01.13 17:45:00	12	petr4	exchange	sell	180.00	17.38		

The Toolbox History tab shows a bit more detail. We can quickly see profitable (green) and lossy (red) operations from colorful highlights.

Tipo	Orden	Año	Tipo	Volumen	Precio	G/I.	T/P	Término	Estado	Comentarios
2018.01.18 11:06:40	25	petrol	sell	300.00 / 300.00	18.38			2018.01.18 12:06:40	Final	
2018.01.18 11:07:00	24	petrol	buy	300.00 / 300.00	18.38	18.31	18.30	2018.01.18 12:05:00	Final	
2018.01.17 13:14:49	23	petrol	sell	300.00 / 300.00	18.38			2018.01.17 13:56:49	Final	sp 18.01
2018.01.17 18:15:00	22	petrol	buy	300.00 / 300.00	17.67	17.70	18.02	2018.01.17 19:15:00	Final	
2018.01.17 18:15:45	21	petrol	sell	300.00 / 300.00	17.68			2018.01.17 19:11:45	Final	st 17.65
2018.01.17 18:16:00	20	petrol	buy	300.00 / 300.00	17.68	17.80	18.08	2018.01.17 19:05:00	Final	
2018.01.17 18:09:40	19	petrol	sell	300.00 / 300.00	17.65			2018.01.17 19:00:40	Final	st 17.65
2018.01.17 18:09:00	18	petrol	buy	300.00 / 300.00	17.68	17.80	18.10	2018.01.17 19:05:00	Final	
2018.01.16 18:53:40	17	petrol	buy	300.00 / 300.00	17.65			2018.01.16 19:53:40	Final	st 17.65
2018.01.16 18:40:00	16	petrol	buy	300.00 / 300.00	17.72	17.67	17.82	2018.01.18 18:40:00	Final	
2018.01.16 18:56:20	15	petrol	sell	300.00 / 300.00	17.65			2018.01.18 19:56:20	Final	sp 17.57
2018.01.16 19:00:00	14	petrol	buy	300.00 / 300.00	17.37	17.30	17.37	2018.01.18 19:25:00	Final	
2018.01.15 20:00:40	13	petrol	sell	300.00 / 300.00	17.28			2018.01.15 20:30:40	Final	
2018.01.15 20:00:00	12	petrol	buy	300.00 / 300.00	17.36	17.20	17.54	2018.01.15 21:00:00	Final	

Finally, the **Operation** tab shows the progress of each particular negotiation. We have the information regarding the entry price, our available financial margin and how the trade is progressing, whether with gain or loss.

.3. Performance Charts

During the backtest or after the end we can follow the evolution of our capital in the **Graph** tab that appears in the **Strategy Tester**.



However, after the end of the backtest, a new tab called Backtest appears. This tab provides a summary of the operational statistics of our strategy. In it we can see the profit Factor of the operation (ratio between gross profit and gross loss), recovery factor, drawdown or retraction, total trading, number of short positions and bought with gains and losses, sharp index among others.

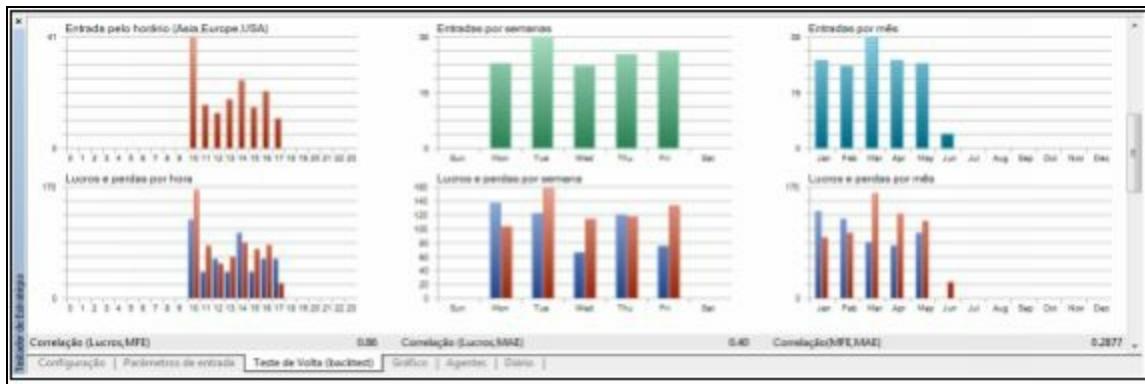
The figure displays a table of operational statistics for the strategy. The table has two main sections: "Qualidade do histórico" (Historical Quality) and "Operações" (Operations). The "Operações" section is expanded to show detailed metrics for buy and sell operations. The table includes columns for metric name, value, and description. The "Operações" section also includes a "Total" row and a "Maior" (Max) row. At the bottom of the table, there is a summary of consecutive wins and losses. The table is part of a larger interface with a navigation bar at the bottom: "Configuração" (Configuration), "Parâmetros de entrada" (Input Parameters), "Teste de Volta (backtest)" (Backtest Test) [highlighted with a red box], "Gráfico" (Graph), "Agentes" (Agents), and "Diário" (Daily).

Qualidade do histórico	93%	Ativos		
Barras	8733	Ticks	173527	1
Depósito Inicial	100 000,00			
Lucro Líquido Total	-105,00	Rebaixamento Absoluto do Saldo	105,00	Rebaixamento Absoluto do Capital Líquido
Lucro Bruto	520,00	Rebaixamento Máximo do Saldo	181,00 (0,18%)	Rebaixamento Máximo do Capital Líquido
Perda Bruta	-625,00	Rebaixamento Relativo do Saldo	0,18% (181,00)	Rebaixamento Relativo do Capital Líquido
Fator de Lucro	0,83	Retorno Esperado (Payoff)	-0,66	Nível de Margem
Fator de Recuperação	-0,53	Índice de Sharpe	-0,07	Z-Pontuação
AHPR	1.0000 (-0,00%)	Correlação LR	-0,89	Resultado OnTester
GHPR	1.0000 (-0,00%)	Erro Padrão LR	26,86	
Total de Negociações	160	Posições Vendidas (% de ganhos)	0 (0,00%)	Posições Compradas (% de ganhos)
Ofertas Total	320	Negociações com Lucro (% de total)	33 (20,63%)	Negociações com Perda (% de total)
	Maior	lucro da negociação	20,00	perda na Negociação
	Média	lucro da negociação	15,76	perda na Negociação
	Máximo	ganhos consecutivos (\$)	3 (41,00)	perdas consecutivas (\$)
	Máxima	lucro consecutivo (contagem)	41,00 (3)	perda consecutiva (contagem)
	Média	ganhos consecutivos	1	perdas consecutivas

We can visualize bar charts informing about the schedules, the days of the week and the month where we have more entries in the traders.

You can track the frequency of profits and losses: by EA operating hours, days of the week, and months of the year. We also have information regarding the maximum time set and the minimum duration of a position.

However, as we can see, MetaTrader 5 gives us a very thorough review of our backtest.



--=*= *=-=--

Congratulations on getting here!! Thank you for your confidence in reading this e-book. Sorry for the problems with the translation. Feel free to submit suggestions, criticisms and errors in the texts and subjects covered in the e-book.

Now that you've been able to create your first EA and make the backtests, it means you have a new journey of study towards more sophisticated strategies. It is worth remembering that the EA taught in this book is purely demonstrative and offers no means to use as a suggestion of real investments.

If you want to know other books, with similar didactics, it is worth checking the following:

[How to create indicators and scripts in MQL5 \(step by step\).](#)

[How to Properly Optimize Investment Robots in MetaTrader 5](#)

If you are interested in any of these books please contact me by e-mail:rafaelfvcs@gmail.com

This material was written thinking of you dear reader. It is good to remember that we are all motivated by criticism or praise. If this book was helpful to you in any way, whether or not you liked the didactics, please leave a comment on the **Amazon** store. This kind of attitude is of extreme importance so that future editions are even better.

Thank you very much!

[1] **Obs:** The information contained in this e-book does not represent or offer any suggestion and / or expectation of investment.

[2] [https://en.wikipedia.org/wiki/Bill_Williams_\(trader\)](https://en.wikipedia.org/wiki/Bill_Williams_(trader))

[3] More information send an email to: rafaelfvcs@gmail.com .

[4] By the same author (rafaelfvcs@gmail.com)