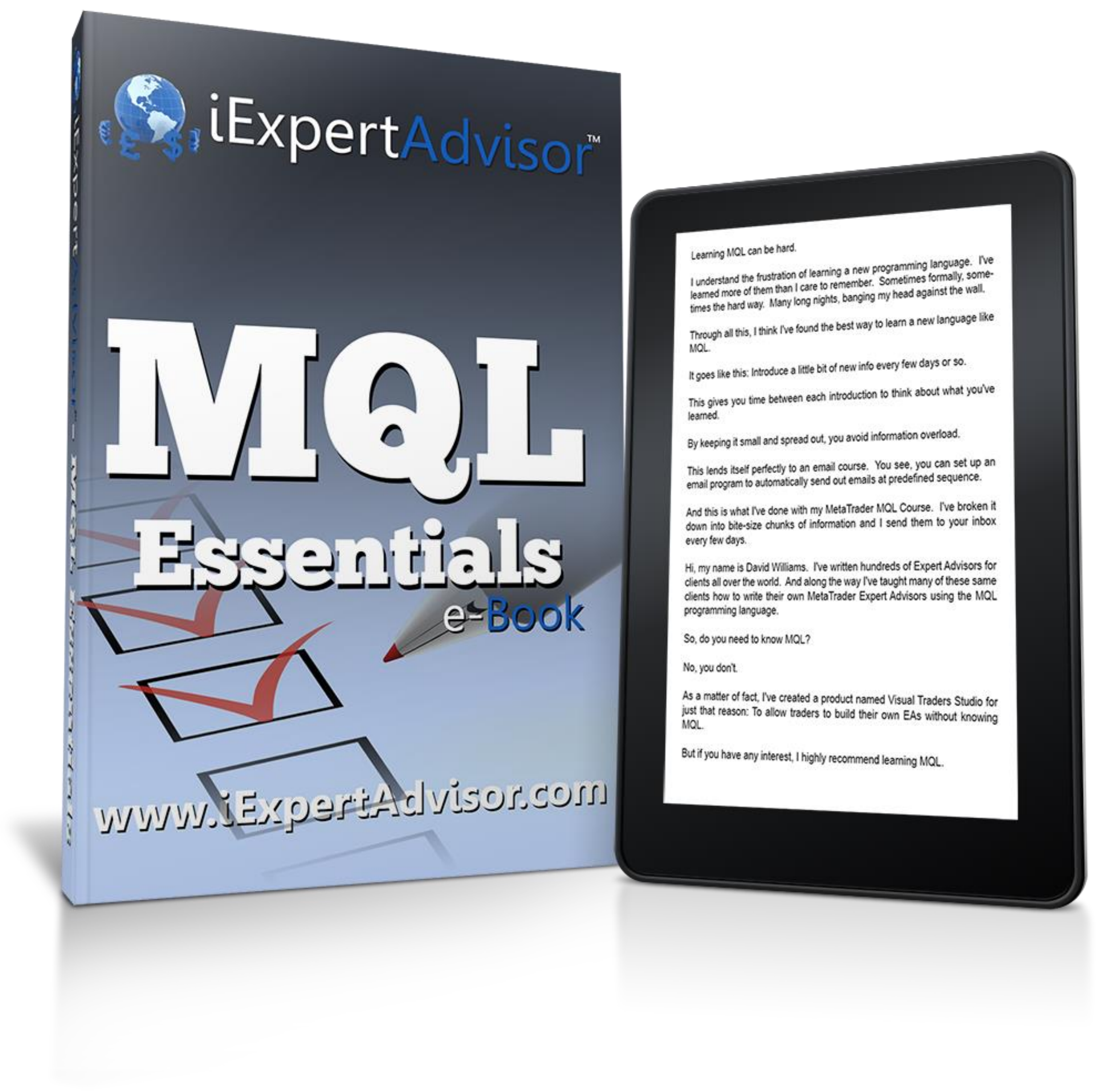


Learning MQL can be hard.

I understand the frustration of learning a new programming language. I've learned more of them than I care to remember. Sometimes formally, sometimes the hard way. Many long nights, banging my head against the wall.

Through all this, I think I've found the best way to learn a new language like MQL.

It goes like this: Introduce a little bit of new info every few days or so.

This gives you time between each introduction to think about what you've learned.

By keeping it small and spread out, you avoid information overload.

This lends itself perfectly to an email course. You see, you can set up an email program to automatically send out emails at predefined sequence.

And this is what I've done with my MetaTrader MQL Course. I've broken it down into bite-size chunks of information and I send them to your inbox every few days.

Hi, my name is David Williams. I've written hundreds of Expert Advisors for clients all over the world. And along the way I've taught many of these same clients how to write their own MetaTrader Expert Advisors using the MQL programming language.

So, do you need to know MQL?

No, you don't.

As a matter of fact, I've created a product named Visual Traders Studio for just that reason: To allow traders to build their own EAs without knowing MQL.

But if you have any interest, I highly recommend learning MQL.

Contents

Essential #1: Expert Advisors Background Knowledge

- An Expert Advisor is an executable program that runs on the MetaTrader platform.
  - An Expert Advisor is built by editing an MQL file and then compiling the MQL file into executable Expert Advisor code.
  - For MetaTrader4, the MQL file has an extension of *mq4*.
  - The *mq4* file is a human-readable text file. This means you can read and edit it with any text editor, like ***Notepad*** or ***Microsoft Word***.
  - For MetaTrader4, the Expert Advisor file extension is *ex4*.
  - If the Alpari version of MetaTrader is installed on your PC, the folder:

    "C:\Program Files\ MetaTrader - Alpari UK\experts"
    contains both the mq4 and ex4 files.

- If you double click on an mq4 file, the *MetaEditor* application will start and load the mq4 file.
  - The *MetaEditor* application is part of the MetaTrader platform. It is automatically installed when the MetaTrader platform is installed.
  - The *MetaEditor* application is used to edit MQL code. It has many useful features found in most code editors, including intelli-sense, context help and an online library.
  - While editing an mq4 file from within the *MetaEditor*, an Expert Advisor is built by clicking the *Compile* button. If there are no syntax errors, the *MetaEditor* application will create the ex4 file.
  - The ex4 file is not a human readable file. It is a binary file interpreted and executed by the MetaTrader platform.
  - By compiling the mq4 file, you are converting the MQL language, which is human readable text, into a binary file that can be read by a machine.

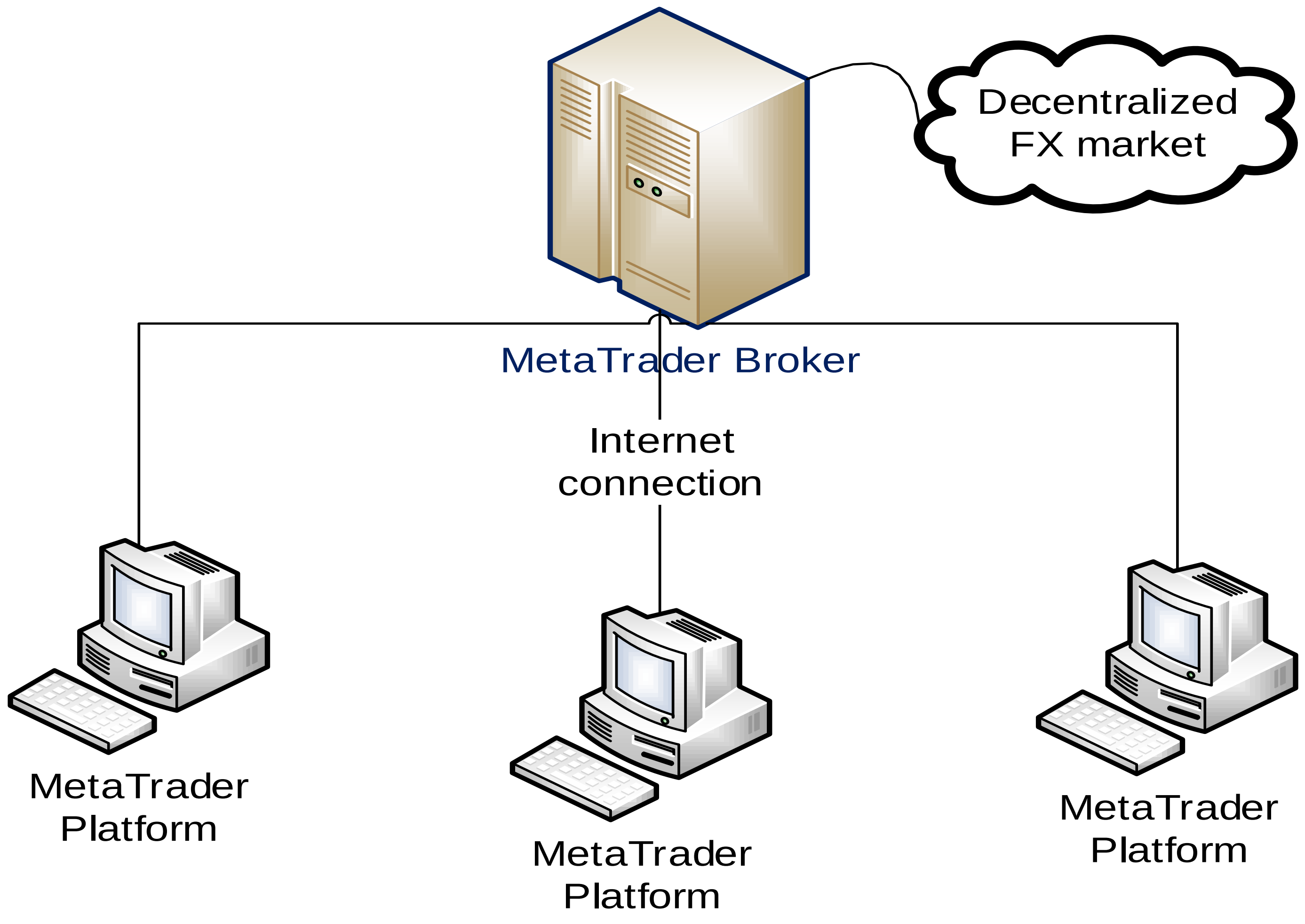
- When the MetaTrader platform is started, all ex4 files in the "experts" folder are available under the "Expert Advisors" folder in the Navigator window of the MetaTrader platform.

- VTS converts its drawings into MQL code.
  - VTS automatically creates an mq4 file and places it in the MetaTrader platform's "experts" directory. The mq4 file is then compiled into an ex4 file by VTS.
  - Both the MetaEditor and Terminal applications can be run from within VTS when VTS is configured correctly.

Essential #2: The Tick

- The MetaTrader platform installed on your computer communicates with your broker's server. Your broker's server in turn is connected to the decentralized FX market.
- Each time the price changes, for any currency pair supported by the broker, the new price information is sent to your MetaTrader platform. This new price information is called a ***tick***.
- Any Expert Advisor built by VTS displays a tick count on the price chart when the EA is running. The tick count begins at zero and is incremented on each ***tick***. This is useful to check if your EA is alive and running.

Tip: An Expert Advisor is invoked, or executed, on each incoming ***tick***. If no ticks arrive, the Expert Advisor will not run. In slow markets, an Expert Advisor may be idle for long periods of time.

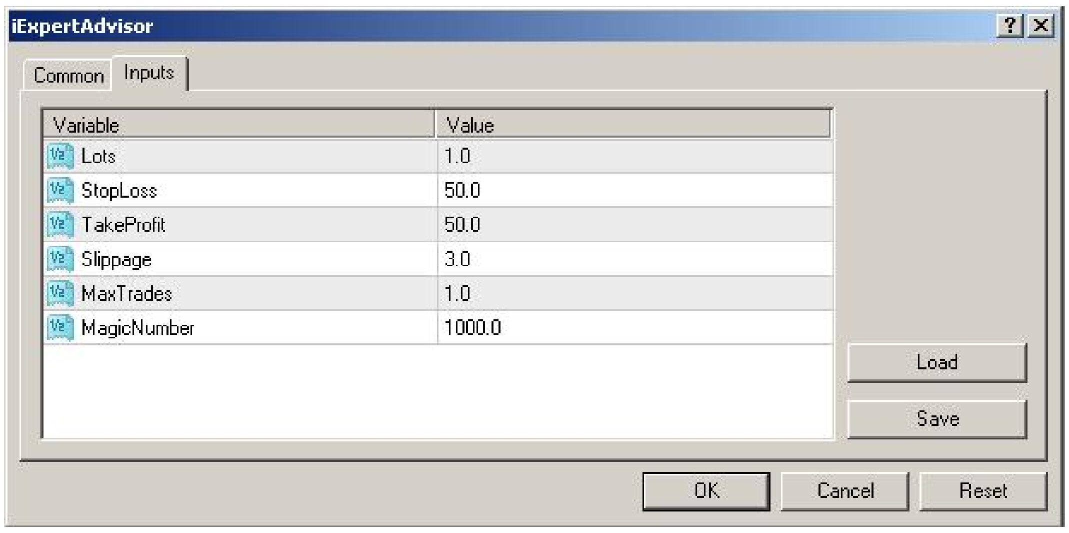


Essential #3: Variables and Data Types

- In a computer language a Variable is used to store data.
- The type of data stored by a Variable is determined by the variable's Data Type.
- The Data Types in MQL are:

MQL Data Types			
Name	**MQL keyword**	**Description**	**Example**
Integer	int	Used to store a whole number	0, 1, 2, …
Double	double	Used to store a decimal number	5.75
Boolean	bool	Used to store one of two values	true or false
String	string	Used to store one or more characters	"a" or "hello world"
DateTime	datetime	Used to store a time value. It's actually an integer value, where the current time is the number of seconds since January 1, 1970!	
Color	color	Used to store a color value	Red, Blue, etc.

- The syntax for declaring a variable is simple: the Data Type, the name and a semicolon (That's right, lines in MQL end with a semicolon)
  - For example: double myDouble;
- Any Variable in MQL can be used as an input parameter to your EA.
  - When you attach your EA to a chart, you can specify the values of these ***Inputs***.



Variable	Value
Lots	1.0
StopLoss	50.0
TakeProfit	50.0
Slippage	3.0
MaxTrades	1.0
MagicNumber	1000.0

  - When the keyword ***extern*** is used to declare a Variable, it will appear on the ***Inputs*** tab. (The name ***extern*** is likely short for the word external).
  - For example:
    - extern double myDouble;

Tip: There is support for *Global* variables in MQL. Normally a *Global* variable is a variable that can be accessed anywhere in the code. *The meaning is different in MQL!*

  - In MQL, a normal variable is already global. It can accessed from anywhere in the Expert Advisor.
  - A *Global* variable in MQL retains its value between EA starts and stops. The value is actually stored in a file in your MetaTrader platform.
  - MQL *Global* variables are not needed to develop most EAs.

Essential #4: Built-in MQL Variables

There are a number of variables built into MQL. These variables are predefined and are always available to be accessed.

Common MQL Built-in Variables			
Name	**Type**	**Description**	
Bid	double	The latest buyer's price (offer price, bid price) of the current symbol	
Ask	double	The latest seller's price (ask price) for the current symbol	
Close	double	The latest close price for the current symbol	
Open	double	The latest open price for the current symbol	
High	double	The latest high price for the current symbol	
Low	double	The latest low price for the current symbol	
Point	double	The value of one point for the current currency	

- The Point value for the EURUSD currency pair is: 0.000010
- The Point value for the USDJPY currency pair is: 0.001000

The Point value is often needed when converting between integer numbers and price values. For example, suppose you need to define an offset for your EA: a number of points above the highest price. It's better to use a value like 20 instead of 0.00020. So the common technique is to take the integer number and multiply is by the Point value to get the correct value:

20 * Point

Note: This technique can be done in VTS for any variable by simply using the ***Apply an Offset*** checkbox.

Tip:

The values in the above table are for the current chart on which the EA is running. It's possible to get any of these values for any chart using the MQL function ***MarketInfo***.

Essential #5: Functions

- In a programing language a *Function* is a section of code that performs a specific task.
  - A function may return a value.
    - For example, a function named *GetHighest* may return the highest value.
  - A function may *not* return a value.
    - For example, a function named *PrintDate* may simply print the date on the price chart.
  - A function may accept parameters.
    - For example, a function named *Add* may accept two numbers and add them and return the result.
- Functions that return values actually return a variable, so the variable has a Data Type, such as Integer, Double, etc.
- Function parameters are variables, so they also have a Data Type.
- In MQL, there are two types of functions: user-created and platform functions.
- A drawing in VTS is equivalent to a user-function.
- MQL has many, many built-in functions. They are all available from the VTS Toolbox.
- When a function element is added to a VTS drawing, VTS automatically creates a variable to hold the return value of the function. The variable name is the name of the function with a preceding underscore (_). For example, for a function named iADX0, a variable named _iADX0 is created to hold the value calculated by the iADX0 function.

Tip:

Functions are the foundation of modern programming. The reason is because once a function has been completely tested, it can be used over and over again without any real effort. Use VTS to create your own user-created functions whenever possible.

Essential #6: The 3 Special MQL Functions

Every Expert Advisor has 3 special functions: ***init, start*** and ***deinit.***

init is short for initialization. The init function is only called once - the first time the Expert Advisor is attached to a chart.

start is the entry point of the Expert Advisor. Each time the Expert Advisor is run, program execution begins at the start function.

deinit is short for de-initialization. The deinit function is called once when the Expert Advisor is *removed from the chart, or for some other reason that causes the Expert Advisor to stop* executing.

These functions are not required to build an EA. Of course if there is not a ***start*** function, the EA will do nothing!

Tip:

In my 8+ years of developing EAs, I have never needed to use the ***deinit*** function. And the ***init*** function I've needed maybe 1% of the time.

So, my advice: don't worry too much about the ***init*** and ***deinit*** functions. Save your brain power for something more important.

Essential #7: Indicator Functions Parameters (including Shift)

Each indicator function in MQL may offer different parameters depending on the exact indicator, but they all support 3 primary parameters: Symbol, Timeframe and Shift.

- The Symbol parameter defines the symbol the indicator will be calculated for. Note: The EA does not need to be attached to a chart of the same symbol.
- The Timeframe parameter defines the timeframe the indicator will be calculated for (from one minute to one month). Note: The EA does not need to be attached to a chart of the same timeframe.
- The Shift parameter defines what candle is used to calculate the indictor value.

The SHIFT parameter is used to identify a candle (or bar) on a MetaTrader price chart.

- The currently forming candle has a shift value of 0.
- The last fully formed candle has a shift value of 1.
- The candle formed before that one has a shift value of 2, and so and so on

So when you look at a MetaTrader price chart, the candle-shift-values read from 0 to N, from right to left. Where the candle with a shift of 0 (zero) is the candle at the far right edge of the chart.

This concept of a candle being identified by its shift-value runs deep in the MQL language.

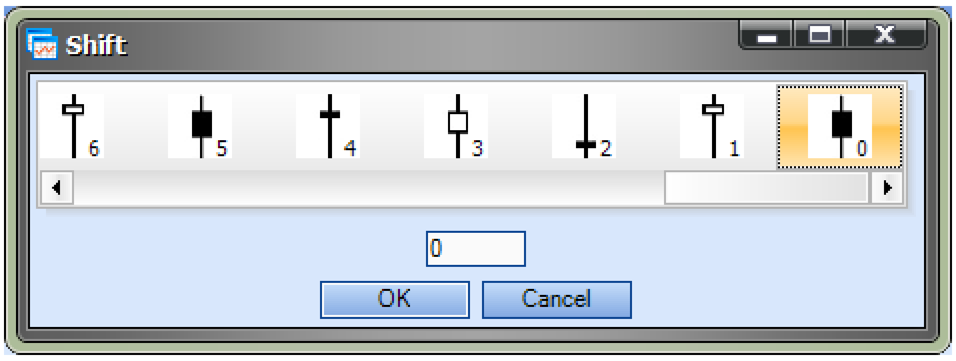
Any of the 4 price values (High, Low, Open, Close) can be obtained using a shift value as an index to the built-in price series that are available in MQL.

For example:

- High[0] : This is the high value of the currently forming candle.
- Close[1]: This is the close value of the last fully formed candle.
- Open[5]: This is the open price from 5 candles ago

Every technical indicator provides a shift value. For example, here is the RSI function:
double iRSI(string symbol, int timeframe, int period, int applied_price, int shift)
To get the RSI value for the last candle, set the shift parameter to 1.

In the VTS-MetaTrader EA builder, every shift parameter offers a visual selection menu. To find it, just scroll up in the parameter pull-down menu and select the first item “choose…“.



The bar or candle that is currently forming (furthest to the right on a candlestick chart) is numbered zero. Each bar to the left increases in number. A series, or array, of Indicator values also follow this concept of number 0 starting at the far right with each number increasing to the left. The Shift of an indicator value is the index in the array of the value, where shift=0 is the latest value and shift=1 is the value from the last period.

Tip:

Using a shift parameter of zero may force the indicator calculation to be updated on each incoming tick. This is because the Close, High and Low values of the current bar (shift=0) are not finalized until the bar is actual closed. This provides the latest value but can lead to unexpected results when testing the value of the indicator against a threshold value: the indicator value can quickly move above or below the threshold on each tick.

Essential #8: Built-in Series

Similar to the built-in variables in Essential #4, MQL also supports ***lists*** of variables. These are called ***series arrays***, but you can just think of them as a ***list*** of values.

Series array elements are indexed in the reverse order, i.e., from the last one to the first one. The current bar in the array is indexed as 0.

Common MQL Built-in Series			
Name	**Type**	**Description**	
Close[]	double	Series array that contains close price of each bar of the current chart	
Open[]	double	Series array that contains the open price of each bar of the current chart	
High[]	double	Series array that contains open the high price of each bar of the current chart	
Low[]	double	Series array that contains the low price of each bar of the current chart	
Time[]	Integer	Series array that contains the open time of each bar of the current chart	
Volume[]	double	Series array that contains tick volumes of each bar of the current chart	

Examples:

- The high value of the current bar: High[0]
- The high value of last fully formed bar: High[1]
- On a 1-hour chart, the low value from 5 hours ago: Low[5]

Tip:

These series return the value for the symbol and time frame of the current chart. To get the values for any time frame or any symbol, use these functions:

- iClose
- iHigh
- iLow
- iOPen
- iTime
- iVolume

Essential #9: Writing Data out of an Expert Advisor

At times, it's very helpful to view the actual values of indicators or other variables while your EA runs.

There are two basic methods for getting information out of an Expert Advisor. The first is the MQL function ***Print*** and the second is the MQL function ***Comment***.

The MQL function ***Print*** writes information to a log file and it also appears in the Experts tab of the Terminal and the Journal tab of the strategy tester.

The ***Print*** function is able to easily write any data type except for arrays. (An array must be written by addressing each index.)

The ***Comment*** function is used similarly to the ***Print*** function, however its data is written to the top left corner of price chart to which the Expert Advisor is attached.

Tip:

Each call to the Comment function clears the previous information that was on the chart. In order to add to the data on the price chart, the data must be managed by the Expert Advisor. This is usually done by adding information to a single string variable throughout the Expert Advisor using the MQL function ***StringConcatenate***. The Comment function is then called once before exiting the Expert Advisor.

Essential #10: Using Custom Indicators

A Custom Indicator draws lines, arrows or some other objects directly on a price chart or in a window below a price chart. For an Expert Advisor to interpret the output of a Custom Indicator, it must ***read the value programmatically***.

The MQL function ***iCustom*** is used to read the values of a Custom Indicator.

Here is the MQL help information for the ***iCustom*** function (don't worry; I'll explain the important details below):

double iCustom(string symbol, int timeframe, string name, ..., int mode, int shift)
You are probably familiar with the parameters ***symbol, timeframe*** and ***shift***.

The name parameter is simply the name of the Custom Indicator, but it must be in the "experts\indicators" folder of your MT platform. This is the same location that VTS looks for Custom Indicators to make them available in the VTS Toolbox. So if you are able to drag your Custom Indicator onto a drawing pad, it is probably in the correct folder.

The ellipses (...) parameter represents any number of input parameters used by the Custom Indicator. These are the values you can modify when you attach your Custom Indicator to a price chart.

In order for your Expert Advisor to "read" your Custom Indicator correctly, the parameters passed in programmatically must be equivalent to the values that are used when you attach your Custom Indicator to a chart.

Usually you can copy the parameters from the Custom Indicator configuration screen. This video shows you how to do this in VTS:
    http://www.youtube.com/watch?v=OvH6mjuDG8k

I say usually, because this is not always reliable. If you don't have access to the MQL code, you may need to experiment to get the right values.

That's part 1 - defining the parameters. Once you have the parameters correct, the second part is to determine what values are being written to what ***buffer***.

Generally speaking, a buffer is used to draw lines or objects on the chart.

A MetaTrader 4 Custom Indicator supports up to 8 buffers. So this means a Custom Indicator can draw up to 8 different lines on a chart. These lines can be different colors and different styles (solid, dash, dots).

But not all of the buffers are always used to draw lines. Often, some of them are used for internal calculations - especially for complex Custom Indicators.

The output buffers are accessed using the mode parameter of the ***iCustom*** function.

- When the mode parameter is 0, the ***iCustom*** function will return the value for buffer 0.
- When the mode parameter is 1, the ***iCustom*** function will return the value for buffer 1.
- And so on, to the maximum, which is 7 (0 through 7 buffers equates to 8 buffers).

Tip:
Custom indicators that draw solid lines are easy to use programmatically. Custom indicators that draw lines that change color pose a challenge, but with enough persistence can be used programmatically. Custom indicators that draw arrows, thumbs and other objects are very challenging; if possible, the MQL code of the custom indicator should be referenced when using these kinds of custom indicators.

Essential #11: Using the MetaEditor

The MetaEditor is the application used to build Expert Advisors, Scripts and Custom Indicators. It is both a text editor and a compiler.

The text editor is useful for writing MQL code because it displays variables and functions in specific colors. Also, when entering an MQL function, the MetaEditor displays the parameters of the function. (Functions and parameters are covered in Module 2).

A compiler is a program that translates the human readable text of a programming language into a file that can be read and executed by a computer.

The human readable text must follow the compiler's rules of syntax.

The syntax specifies the human readable text of the programming language that is understood by the complier. The syntax also specifies the special meaning of characters such as parenthesis, semi colons, etc.

In addition to the file editing window, the MetaEditor contains a Navigator and a Toolbox window.

The Errors tab is an important window: when compiling, the MetaEditor writes messages to the Error tab of the Toolbox. Any syntax errors found during compilation are listed in this window. Double clicking the error message will focus the cursor on the offending line of MQL. Warning messages are also listed in the Errors window.

If there are any syntax errors present, the Expert Advisor's ex4 file will not be built (created). However, warning messages alone will not prevent an Expert Advisor from building.

Tip:

The Navigator window provides MQL help via three tabs: Files, Dictionary and Search. The help is very good. All MQL functions are fully defined and there are very few errors in the documentation. Even when I use VTS to build and EA, I often refer to the MetaEditor help for detailed info about how to use some MQL functions.

Essential #12: The Order Functions

There are three main functions in MQL for working with orders (or trades): ***OrderSend, OrderModify and OrderClose***.

MQL Order Functions			
Name	**Returns**	**Description**	
OrderSend	The ticket of the open order or -1	Used to open a new order.	
OrderModify	True or false	Used to modify the stoploss or takeprofit of an existing order	
OrderClose	True or false	Used to fully or partially close a trade	

OrderSend is tricky because the StopLoss and Takeprofit values must be price values. That means if you want a stoploss of 20, you need to perform the calculations to find the price value that is 20 points below the current price (for a BUY order) or 20 points above the current price for a SELL order. The same is true when using the OrderModify function.

The ***OrderClose*** is tricky because you need to pass in the order's ticket number. The easiest way to find a ticket number for an order is to loop through all of the open orders.

OrderClose is used to close an entire order or just part of it. For example, if there is an open order of 2 lots, ***OrderClose*** can be used to close 1 lot and the remaining lot will remain open.

Note: There is no "adding to an open position". To add, you need to open another order for the desired lot amount.

Tip:

- Ask is the preferred price used for ***opening*** BUY orders.
- Bid is the preferred price used for ***opening*** SELL orders.
- Bid is the preferred used for ***closing*** BUY orders.
- Ask is the preferred price for ***closing*** SELL orders.

If these values are not used, the function will likely fail to open or close the order.

Essential #13: Order Selection and the Magic Number

As noted in Essential 11, an order is closed using its ticket number and a ticket number is usually found by looping through all of the open orders for an account.

Basically there are 4 ways to select orders:

Order Selection			
Type	**Description**	**Usage**	
By Account	All orders for the account	Used alone	
By Order Type	All BUY or SELL orders for the account	Can be used with others	
By Symbol	All orders of a currency pair (Symbol)	Can be used with others	
By Magic Number	All orders with the matching magic number	Can be used with others	
By Ticket	The single order with the matching ticket	Used alone	

The magic number is a unique number that is assigned to an order when it is opened. For example, when an Expert Advisor opens an order on the EURUSD, it can assign a magic number of 3000. Later when the Expert Advisor requests information about all of the open orders for the account, it recognizes the orders it has opened by their magic number - any order whose magic number is 3000. The magic number allows an Expert Advisor a method to tag and identify orders.

The most common trade set for an Expert Advisor to manage is the set all positions for a currency symbol, followed closely by all positions for a currency symbol and magic number.

The MQL code for selecting orders is simple, but tedious. All VTS functions that required order selection display a screen for easily selecting the criteria.

Tip:

The magic number can only be set programmatically. The magic number of a manually opened trade is always 0. Also, the magic number of an order cannot be changed after the order has been opened.