Евгений Ждан
MQL4-программирование: торговый робот за один вечер

ISBN 9785448575594

Аннотация

Чтобы освободиться от рутины и сберечь нервы, каждый трейдер задумывается

об автоматизации своей торговли. Эта книга поможет освоить mql4-программирование любому человеку с любым уровнем образования. Книга написана простым языком без сложной терминологии. На протяжении всей книги автор и читатель вместе разрабатывают торговый советник для платформы MetaTrader4.

MQL4-программирование: торговый робот за один вечер

Евгений Ждан

ISBN 978-5-4485-7559-4
Создано в интеллектуальной издательской системе Ridero

Введение

Рынок FOREX, как и биржа ценных бумаг привлекает все новых и новых людей. Это и понятно – делать деньги из воздуха хочется всем. Но, не все так просто.
Данные о том, что именно **зарабатывающих** трейдеров не более 5% встречаются повсеместно. Деньги остальных 95% делятся между первыми 5% и брокерами/диллинговыми центрами.
Чтобы попасть в число успешных трейдеров необходимо иметь четкую стратегию работы и железные нервы. В принципе, прибыльных торговых стратегий и систем существует не мало. Основная проблема работы заключается в психологии трейдера. Как правило, почти все трейдеры стартуют стабильно успешно. Немногим далее – постепенный или внезапный крах.
Дело в том, что, начиная торговать, трейдер выполняет условия своей торговой стратегии. После череды успешных сделок, последний расслабляется, начинает думать, что поймал удачу за бороду и жизнь обеспечена. Появляется чрезмерная уверенность в своих действиях и трейдер начинает отклоняться от торговой стратегии. Открываются сделки не по системе, а по «наитию». Пребывая в эйфории от череды успешно закрытых ордеров, трейдер увеличивает торговый лот. И уже скоро этот человек отправляется в число тех теряющих деньги 95% трейдеров.
Решить проблему психологической стороны торговли может ее автоматизация – использование **торгового эксперта** (советника, торгового робота), который будет работать на счете трейдера без вмешательства человека – хозяина счета.
Торговый робот лишен эмоций и способен монотонно выполнять свой алгоритм с приходом каждого нового ценового значения. Конечно, иногда трейдеру придется запрещать ему работу, например, во время крайне-важных финансово-политических новостей, когда волатильность возрастает в разы. Например, такими событиями в недавнем прошлом являлись Britain Exit – «Брекзит» – кампания сторонников выхода Великобритании из ЕС, выборы президентов США и Франции, авария на АЭС Фукусима-1, спровоцировавшая обвал японской национальной валюты и т. п. Надеюсь, мысль вам понятна.
В этой книге мы научимся делать торговых роботов для для самого распространенного и самого удобного торгового терминала MetaTrader4 от компании MetaQuotes. Если быть точнее, в этой книге мы по шагам создадим советника (Expert Advisor), полностью готового «к употреблению». Естественно, прибыльность конечного продукта я не обещаю, нам важно другое – научиться их делать.
После изучения этой книги Вы сможете воплощать свои самые смелые торговые идеи самостоятельно, не прибегая к услугам mql-программистов. Также, вы сможете и сами зарабатывать, программируя советники на заказ.

Пожалуй, уже к середине изучения данной книги вы будете отклоняться от нее и вносить свои коррективы в создаваемый нами советник. Так и должно быть. Поехали.

Немного теории

Типы данных

Торговый эксперт оперирует данными. Он работает с поступающими ценами, ценовыми значениями индикаторов, ведет подсчеты открытых ордеров, что-то печатает в Журнал торгового терминала.

В mql4 существуют следующие типы данных:

Теперь весь наш код выглядит так:

Основные типы данных:
– целые числа (char, short, int, long, uchar, ushort, uint, ulong)
– логические (bool)
– литералы (ushort)
– строки (string)
– числа с плавающей точкой (double, float)
– цвет (color)
– дата и время (datetime)
– перечисления (enum)
Сложные типы данных:
– структуры;
– классы.

На первых порах вам не понадобится и 70% из вышеперечисленного. Рассмотрим только то, что нам будет нужно в рамках разработки нашего торгового эксперта.

– **Тип int** – целые числа, т. е. 1, 2, 5,100, 1425…
– **Тип double** – числа с дробной частью (с запятой): 1,0254, 0,0547…
– **Тип bool** – имеет только 2 значения – true (правда) и false (ложь).
– **Тип string** – строковые значения, т. е. слова: «слово», «предложение из четырех слов»…

Переменные

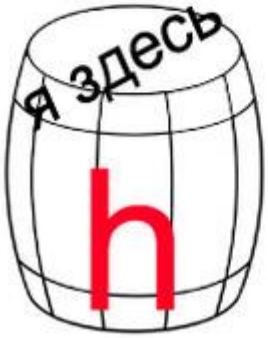
Переменные – это буквенные символы, содержащие в себе значения какого-либо типа.

Переменные – это бочонки, в которых что-то лежит.

С типом bool все также, например переменная bool b = true, означает, что бочонок с именем b содержит в себе true.

int x = 100 означает, что в бочонке по имени x лежит целое число 100;

double z = 0.03254 означает, что в бочонке по имени z лежит число 0,03254



string h = “я здесь” означает, что в бочонке лежит эта строка - “я здесь”

Перед тем как создавать переменную для последующей работы с ней, нужно обязательно объявить ее тип, чтобы компилятор редактора MetaEditor (в нем мы будем создавать нашего робота) знал, что в этой переменной будет храниться. Названия переменных не могут начинаться с цифры.
Объявлять переменную нужно только 1 раз. Позднее мы поговорим о том, где их можно объявлять и как это влияет на последующую работу.

Условные операторы if-else

Условные операторы if-else применяются всегда и везде. If – означает «если», else – «если нет, то».
Например:

```
if (x &lt;y) // Если содержимое бочонка x меньше содержимого бочонка y

Что-то делаем, например, открываем ордер. Или закрываем другой ордер, да все что угодно!

else // А если x не меньше y, делаем то, что ниже, в фигурных скобках

Делаем что-то здесь.
```

использование оператора else не обязательно, все зависит от конкретной задачи.
Два слеша (косые черты) – // , то что после них в коде советника означают **комментарии.** При компиляции вашего советника (превращения вашего кода в машинный код, понятный компьютеру), комментарии игнорируются. Комментарии желательно писать для себя, чтобы не забыть что куда и зачем сделано.
Блоки комментариев делаются так:

```
/* это
блок
комментария */
```

Все, что между символами **/* и */** также компилятором игнорируется.
Циклы
В mql4 существуют циклы for и while. Чаще используется for, но, нередко и while.

```
for (int i=0; i &lt;100; i++)

что-то считаем 100 раз.
```

int i = 0 – объявляем переменную, которая будет работать в пределах данного цикла; i <100 – цикл прокрутится 100 раз, от 0 до 99; i++ (инкремент) означает, что при каждой прокрутке (итерации) цикла, переменная i будет увеличена на единицу.

```
bool x = false; //присваиваем переменной x типа bool значение false
while (x==false) //пока x равен false. Два символа равно "==" означают сравнение

/*
здесь будут выполняться какие-то условия.
Как только x станет true, цикл прекратится.
*/
//например
x = true;//после первого же прохода делаем x равным true
//и цикл прекращается
```

В процессе написания советника мы будем использовать оба этих цикла, и вы без труда с ними разберетесь.

Техническое задание

Опишем, что и когда должен делать наш будущий советник:

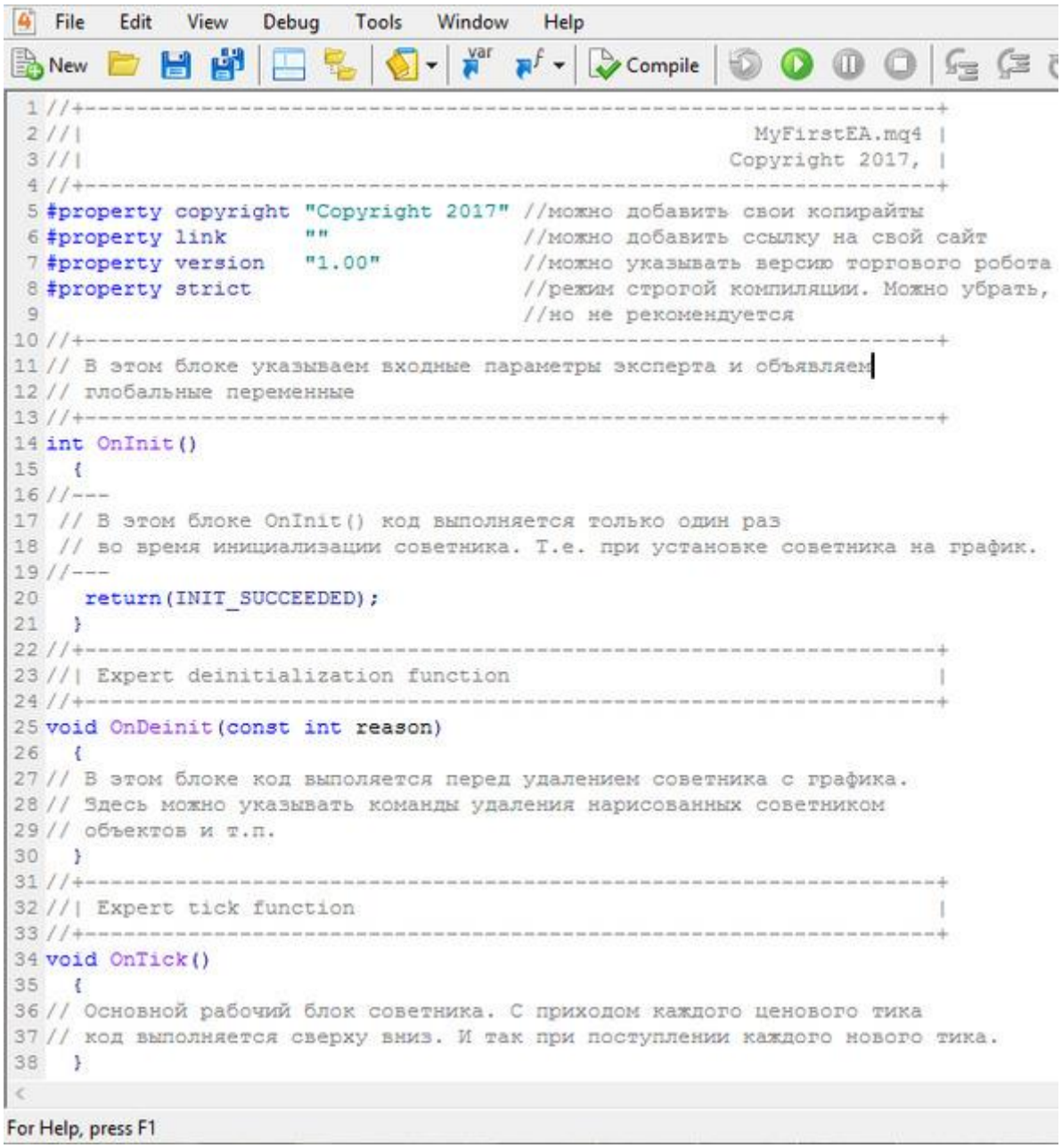
Торговые сигналы будут формировать два стандартных индикатора Envelopes и ZigZag. Эти индикаторы встроены в MetaTrader4 и дополнительно скачивать их не нужно. Я выбрал два именно этих индикатора, т. к. их значения вызываются разными способами. Для Envelopes – с помощью стандартной функции iEnvelopes, а ZigZag вызывается функцией iCustom – ее вам необходимо изучить (хотя, это громко сказано), чтобы в дальнейшем вы умели вызывать данные почти **любых не стандартных** индикаторов для MetaTrader4.

Составим краткое техническое задание:
– Если верхний пик индикатора ZigZag (далее – ZZ) сформировался выше верхней линии индикатора Envelopes (с параметром Shift = 10, остальные – стандартные), выставляем ордер на продажу фиксированным лотом, определенным в настройках советника.
– Если нижний пик ZZ сформировался ниже нижней Envelopes – сигнал на покупку (т. е. наоборот от buy-сигнала).
– Путем модификации (почему модификации, а не сразу при установке ордера – позже, когда будем писать этот код) советник должен устанавливать у ордеров Стоп-Лосс и Тейк-Профит.
– Добавить возможность закрывать ордера при касании ценой противоположной линии Envelopes. Эту функцию можно выключать в настройках.
Если вы читаете эту книгу, я надеюсь, на вашем компьютере уже установлен торговый терминал MetaTrader4 и вы умеете открывать демо-счет. Если нет, нужно этот терминал установить, предварительно зарегистрировавшись у любого брокера, поддерживающего работу MetaTrader4.
А теперь, **переведите свой терминал на английский язык!** Если уж вы решили заняться программированием, привыкайте к английскому, без этого никуда! Сам редактор кода MetaEditor лучше оставить на русском, т. к. при переводе его на английский язык, Справка (F1) получается тоже на английском языке. Не всем это удобно.

Получаем данные индикаторов

Открываем свой MetaTrader4 и нажимаем кнопку F4 на клавиатуре, или левой кнопкой мыши здесь:
В открывшемся редакторе кода нажимаем New (Создать), затем Expert Advisor (template), потом Далее, в поле Name после Expertsдописываем MyFirstEA – это будет названием вашего первого советника. Получится Experts. Поля Autor, link нам в этом учебном советнике не важны. Нажимаем кнопку Далее. Появится окошко Event Handles of the Expert Advisor. Здесь ничего отмечать не нужно и просто нажмем Далее. В появившемся окошке Tester event handless of the Expert Advisor снова ничего не выбираем и жмем Готово. Получаем рабочую область, в которой скоро родится наш торговый робот.

На изображении в комментариях указано, какие блоки за что отвечают.

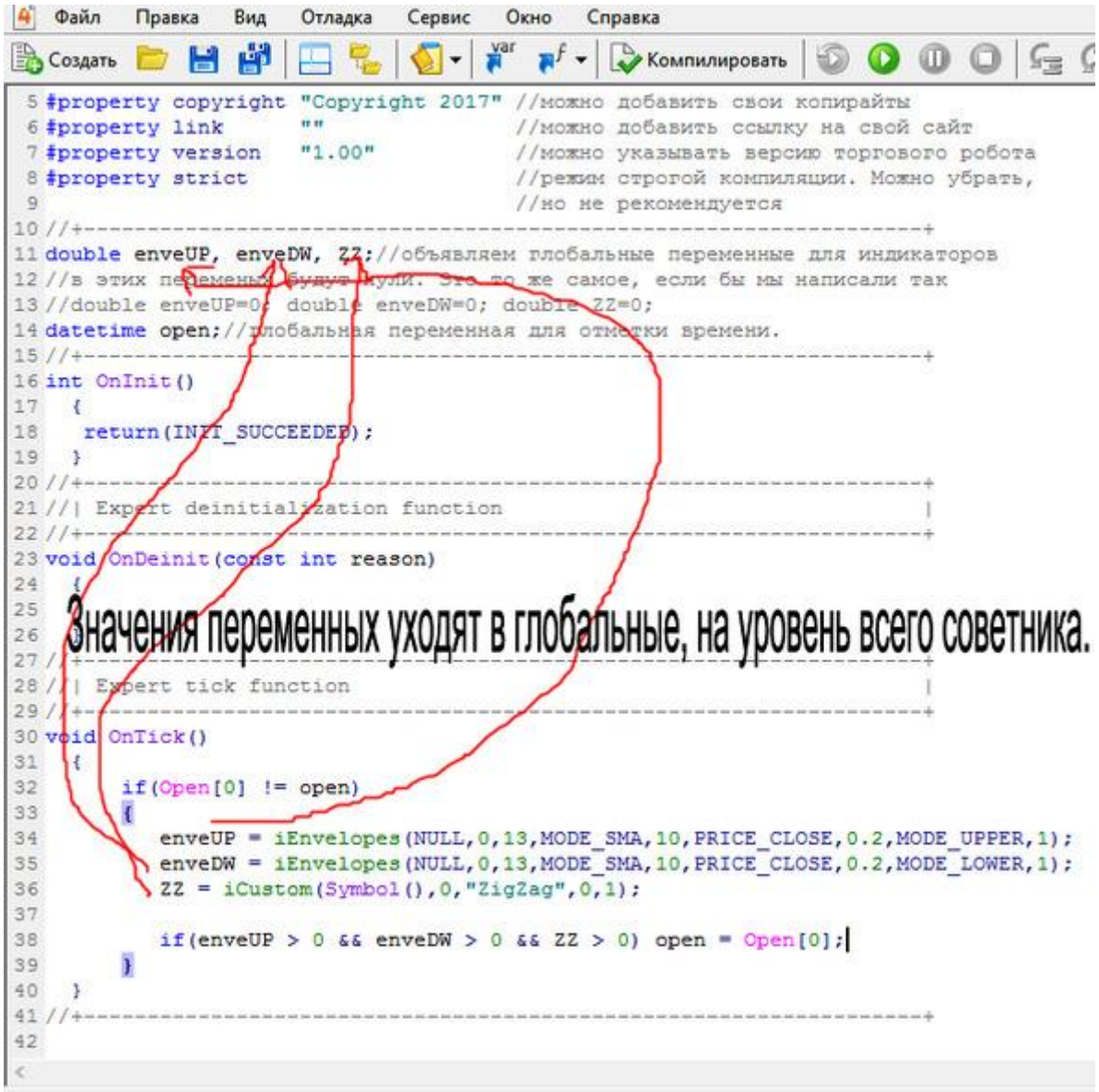


Чтобы узнать ценовые значения индикаторов нам нужно объявить глобальные переменные типа double для верхней и нижней линии индикатора Envelopes. Назовем их enveUP и enveDW. Эти названия можно придумывать самим. То же самое надо сделать и для получения ценового значения индикатора ZZ. Назовем эту переменную ZZ. Почему именно *глобальные* переменные? Для того, чтобы эти значения мы могли вызывать в любом месте программы (т. е. советника). Дело в том, что мы будем вызывать значения индикаторов не на каждом приходящем тике, а один раз на одной свече. Это существенно повысит производительность, т. к. терминалу не нужно будет выполнять одну и ту же операцию на каждом тике. Если мы обернем в фигурные скобки вызов наших индикаторов с записью их значений НЕ в глобальные переменные, то эти значения будут видны только в рамках этих же фигурных скобок. И за пределами их мы получим ошибку. Более подробно постараюсь описать на рисунке ниже.
Перепишите этот код в свой редактор:

```
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
//| MyFirstEA.mq4 |
```

```
//| Copyright 2017, |
//+ – —– —– —– —– —– —– —– —– —– —– —– —– —– —– —– —– –
—– +
#property copyright «Copyright 2017»
#property link»»
#property version «1.00»
#property strict
//+ – —– —– —– —– —– —– —– —– —– —– —– —– —– —– —– —– –
—– +
double enveUP, enveDW, ZZ;
datetime open;
//+ – —– —– —– —– —– —– —– —– —– —– —– —– —– —– —– —– –
—– +
int OnInit ()

return (INIT_SUCCEEDED);

void OnDeinit (const int reason)


void OnTick ()

if (Open [0]!= open)

enveUP = iEnvelopes
(NULL,0,13,MODE_SMA,10,PRICE_CLOSE,0.2,MODE_UPPER,1);
enveDW = iEnvelopes
(NULL,0,13,MODE_SMA,10,PRICE_CLOSE,0.2,MODE_LOWER,1);
ZZ = iCustom (Symbol (),0,«ZigZag», 0,1);
if (enveUP&gt; 0 &amp;&amp; enveDW&gt; 0 &amp;&amp; ZZ&gt; 0) open =
Open [0];
```

Разберем, что же означает каждая строчка.



В глобальных переменных, кроме переменных для значений индикаторов, мы объявили переменную типа datetime с названием open. Сейчас она содержит 0.

ВАЖНЫЙ МОМЕНТ: установите курсор на слово datetime и нажмите на клавиатуре F1 – появится СПРАВКА с описанием, что означает тип datetime. Так можно делать **НА всех** встроенных командах!

if (*Open [0]! = open*): **Если** *Время Открытия Нулевой Свечи* **НЕ РАВНО** *open* (т. е. нулю), то выполнится код в фигурных скобках. Команда Open [0] означает Время Открытия Нулевой (т. е. текущей, еще не закрытой свечи). Также, установите курсор на Open и нажмите F1 – почитайте, что это за команда.

EnveUP = iEnvelopes (NULL,0,13,MODE_SMA,10,PRICE_CLOSE,0.2,MODE_UPPER,1); – нажимаем на iEnvelopes и видим, в каком порядке и какие данные должны быть указаны:

double iEnvelopes (
– string symbol, // имя символа
– int timeframe, // таймфрейм
– int ma_period, // период
– int ma_method, // метод усреднения
– int ma_shift, // сдвиг средней
– int applied_price, // тип цены
– double deviation, // отклонение (в процентах)

– int mode, // индекс линии
– int shift // сдвиг
);

В нашем коде мы не предусмотрели возможность изменять данные индикатора Envelopes. Давайте это исправим. Нам нужно вывести во внешние параметры Период и Отклонение, выраженное в процентах.

В коде советника, над глобальными переменными напишем:

input int period = 14;

input double deviation = 0.10.

Команда input – делает переменную видимой в свойствах советника и пользователь может ее менять.

Поменяем код вызова данных Envelopes. Должно получиться так:

enveUP = iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_UPPER,1);

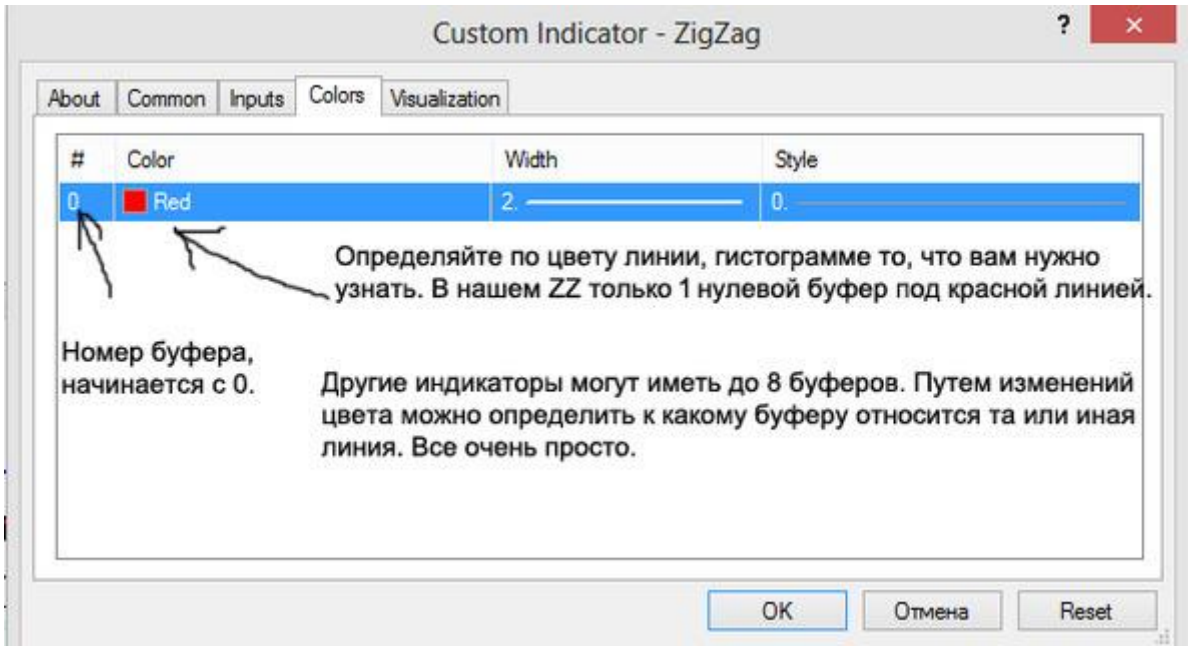
enveDW = iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_LOWER,1);

Цифра 1 в конце вызова означает, что вызываем значение индикатора на первой свече, т. е. предпоследней. Номера свечей считаются от нулевой – текущей и назад. Предыдущая свеча – первая, за ней – вторая и так далее.

Для получения ZZ используем функцию iCustom:

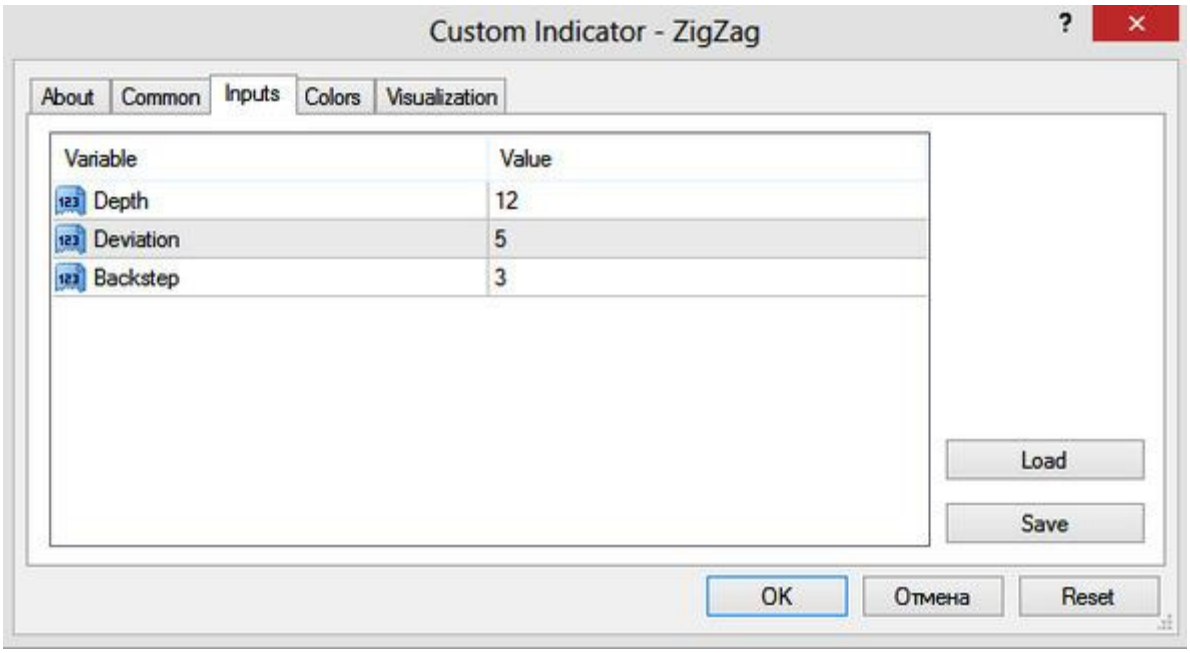
ZZ = iCustom (Symbol (),0,«ZigZag», 0,1);

Вызовите F1 и узнайте подробнее о ней. Предпоследняя цифра 0 здесь – номер буфера индикатора, который нам нужен. Чтобы узнать, какой буфер нам нужен, вызываем свойства нужного индикатора и переключаемся на вкладку Color:



Разберемся с ZZ – добавим его все три параметра во внешние переменные. Какие же параметры добавлять? Вызываем свойства индикатора:

Добавляем во внешние переменные Depth, Deviation, BackStep и присвоим им стандартные значения. Обратите внимание на синие обозначения перед названиями – 123, что означает, что данные переменные имеют тип int.



– **sinput** string zz = «Настройка ZZ»;
– input int Depth = 12;
– input int Deviation = 5;
– input int BackStep = 3.

Sinput – тоже делает видимой переменную, но она закрыта от оптимизации. Нам здесь это и не нужно. Строка sinput string zz = «Настройка ZZ» служит для визуального удобства управления настройками советника.

Добавляем в ранее написанное:

ZZ = iCustom (Symbol (),0,«ZigZag», 0,1);

наши внешние переменные через запятую в порядке очередности сразу после названия индикатора. Должно получиться так:

ZZ = iCustom (Symbol (),0,«ZigZag», Depth, Deviation, BackStep,0,1);

Теперь весь наш код выглядит так:

```
#property copyright "Copyright 2017"
#property link      ""
#property version   "1.00"
#property strict
//+------------------------------------------------------------------+
input int period = 14;
input double deviation = 0.10;
sinput string zz = "Настройка ZZ";
input int  Depth = 12;
input int  Deviation = 5;
input int  BackStep = 3;
double enveUP, enveDW, ZZ;
datetime open;
//+------------------------------------------------------------------+
int OnInit()
  {
   return(INIT_SUCCEEDED);
  }

void OnDeinit(const int reason)
  {

  }
void OnTick()
  {
      if(Open[0] != open)
      {
         enveUP = iEnvelopes(NULL,0,period,MODE_SMA,10,PRICE_CLOSE,deviation,MODE_UPPER,1);
         enveDW = iEnvelopes(NULL,0,period,MODE_SMA,10,PRICE_CLOSE,deviation,MODE_LOWER,1);
         ZZ = iCustom(Symbol(),0,"ZigZag",Depth, Deviation, BackStep,0,1);
         if(enveUP > 0 && enveDW > 0 && ZZ > 0) open = Open[0];
      }
  }
```

Comment («enveUP:», enveUP,»; enveDW:», enveDW,»; ZZ:», ZZ);

Теперь все, что в пределах фигурных скобок оператора if у нас должно выглядеть

enveUP=iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_UPPER,1);

enveDW=iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_LOWER,1);

ZZ = iCustom (Symbol (),0,«ZigZag», Depth, Deviation, BackStep,0,1);

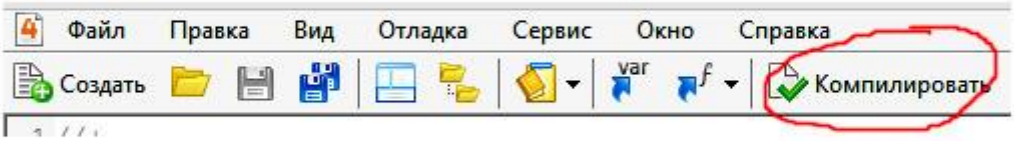
Comment («enveUP:», enveUP,»; enveDW:», enveDW,»; ZZ:», ZZ);//добавили этот код

if (enveUP> 0 && enveDW> 0 && ZZ> 0) open = Open [0];

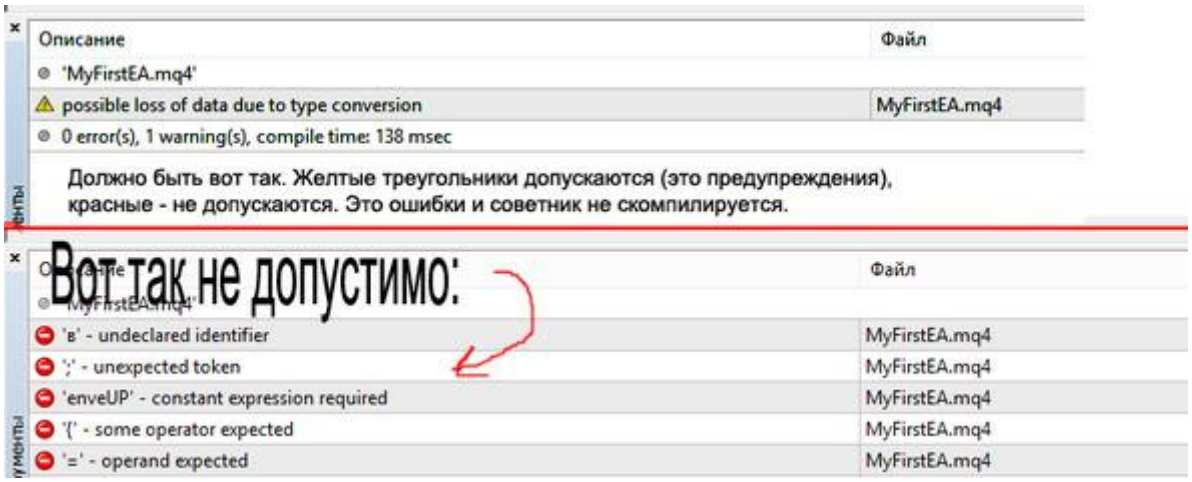
Последняя строчка этого когда говорит: если enveUP больше 0 (а значит, что значение получено) (&& – означает И) и (&&) enveDW тоже больше нуля и (&&) ZZ больше нуля, то в значение open записываем время открытия текущей свечи. Значит, код в фигурных скобках сработает в следующий раз только при формировании нового бара – когда open уже **будет не равно** времени открытия НОВОГО бара. И так постоянно. Есть один минус. Например, индикатор ZZ постоянно перерисовывается, и, советник отреагирует на его значение на первом баре, а позже индикатор перерисуется и пика уже не будет в том месте, где советник его «засёк».

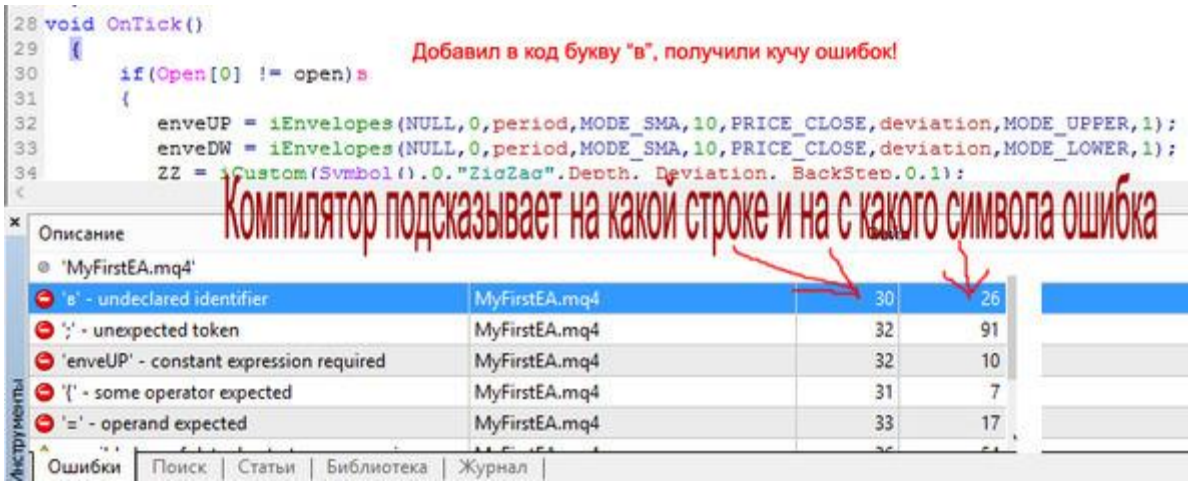
Когда все готово, нужно скомпилировать советник. Делается это нажатием кнопки Компиляция:

Результат должен быть вот таким:



Если ваш советник при компиляции выдает ошибку (красные «кирпичи»), надо найти ее найти и исправить. Обычно, это не сложно – компилятор подскажет в какой строке ошибка. НО! Если где-то не закрыта скобка»)» или фигурная скобка « {», компилятор сам не может определить где именно вы ее хотели закрыть и не покажет точно где ошибка появляется. Тогда придется искать ее самому, проверять каждую строчку кода. С опытом это делать все легче и легче.





Думаю, с компиляцией все понятно. Кода у нас не много и если уж у вас ошибка и появилась, вы ее легко найдете и исправите.

Теперь, проверим правильность получения данных из наших индикаторов:

В левом верхнем углу видим работу нашей функции Comment. Чтобы получить такую картинку в тестере стратегий, при тестировании советника наложите на график индикаторы ZigZag и Envelopes со стандартными параметрами, кроме сдвига (Shift) у Envelopes. В нашем советнике он равен 10. А затем сравните полученные значения, которые отображаются Comment-ом с ценовыми значениями индикаторов на графике. Вы увидите, что все сходится. Если на первом (предыдущем баре) пика ZZ нет, значит значение ZZ будет 0.

На данный момент, компилирущийся код нашего советника выглядит так:

```
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
//| MyFirstEA.mq4 |
//| Copyright 2017, |
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
#property copyright «Copyright 2017»
#property link»»
#property version «1.00»
#property strict
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
input int period = 14;
input double deviation = 0.10;
sinput string zz = «Настройка ZZ»;
input int Depth = 12;
input int Deviation = 5;
input int BackStep = 3;
double enveUP, enveDW, ZZ;
datetime open;
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
int OnInit ()
{
return (INIT_SUCCEEDED);
}
```

```
void OnDeinit (const int reason)
{

}
void OnTick ()
{
// if (Open [0]!= open)
{
enveUP=iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_UPPER,1);
enveDW=iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_LOWER,1);
ZZ = iCustom (Symbol (),0,«ZigZag», Depth, Deviation, BackStep,0,1);
Comment («enveUP:», enveUP,»; enveDW:», enveDW,»; ZZ:», ZZ);
if (enveUP> 0 && enveDW> 0 && ZZ> 0) open = Open [0];
}
}
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
```

Чтобы ваш код выглядел красиво и аккуратно, воспользуйтесь командами Сервис – Стилизатор. Компилятор приведет код вашего советника к «красивому» виду).

Данный этап создания нашего первого советника закончен. Следующая глава – функция подсчета количества открытых ордеров.

Функция подсчета количества открытых ордеров

Скоро мы уже будем писать код открытия ордеров. Чтобы не открывалось несколько сотен ордеров на одном сигнале, пока не закончатся свободные средства, нам понадобится функция подсчета количества уже открытых советником ордеров.

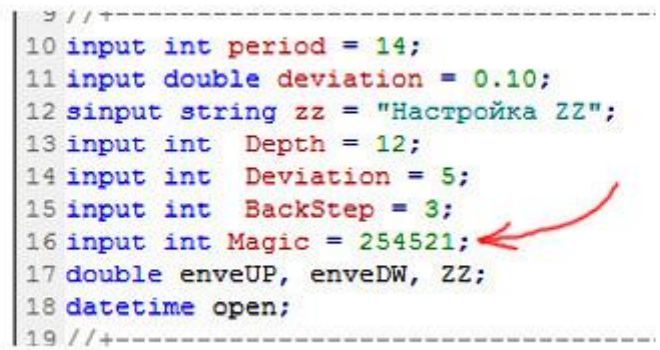
Для того, чтобы наш советник определял, где «его» ордер, а где «чужой», внедрим внешнюю переменную-идентификатор Magic – «магическое число». Дело в то, что при открытии любого ордера советником, этому ордеру терминалом присваивается идентификатор, который выражен каким-либо числом типа int. Для ручных ордеров этот идентификатор всегда равен нулю.

Добавим input int Magic во внешние переменные нашего советника. Допишите этот код:

input int Magic = 254521; (число берем абсолютно любое. Должно быть уникальным и не совпадать с Магиком какого-нибудь другого советника на счете. Иначе, советники будут путаться где чей ордер), под уже написанным ранее кодом:

input int BackStep = 3;

Должно получиться вот так:



Теперь вернемся к функции подсчета открытых ордеров. Функции mql4 располагаются ВНЕ блоков OnInit (), OnDeinit (const int reason) и OnTick (). Коды функций можно располагать как ДО этих блоков, так и после. Я предпочитаю последний вариант. По сути, функции – это кирпичики, из которых состоит ваша программа. С течением времени и накопления опыта, у вас будет свой набор функций, который вы будете постоянно использовать, копируя их из одного своего советника в другой. Изобретать колесо каждый раз не нужно.

Перепишите этот код, вставив его после последней закрывающей фигурной скобки блока OnTick ():

```
int CountTrades (int type, int magic)
{
int count = 0;
for (int i = OrdersTotal () -1; i> =0; i – )
{
if (OrderSelect (i, SELECT_BY_POS, MODE_TRADES))
{
if (OrderSymbol () ==Symbol () && (OrderType () ==type || type==-1) && (OrderMagicNumber () ==magic || magic==-1))
count++;
}
}
return (count);
}
```

Теперь пройдемся по каждой строчке этого кода:

int CountTrades (int type, int magic) – название функции – CountTrades. Функция имеет тип int, что означает, что она вернет нам какое-то целое число, а именно – количество открытых рыночных ордеров. Функция имеет два входных параметра, которые мы должны будем ей передать. Эти параметры будут использоваться функцией при выполнении ее задачи. Функция получается универсальной: передавая ей разные параметры в разных участках программы, можно получать нужные программисту данные.

int type – параметр, принимающий ТИП ордеров (бай, селл, бай-стоп, селл-стоп, лимитные).

int magic – магик ордеров.

Например, нам нужно посчитать количество рыночных БАЙ-ордеров, открытых нашим советником. Мы вызываем функцию таким образом:

CountTrades (OP_BUY, Magic). Где OP_BUY – встроенная в редактор команда, означающая БАЙ-ордер, и Magic – внешняя переменная нашего советника. Подробнее на рисунке:

Эта функция уже сейчас работает и вернет 0, т. к. открытых советников ордеров нет.

```
void OnTick()
  {
     // if(Open[0] != open)
      {
         enveUP = iEnvelopes(NULL,0,period,MODE_SMA,10,PRICE_CLOSE,deviation,MODE_UPPER
         enveDW = iEnvelopes(NULL,0,period,MODE_SMA,10,PRICE_CLOSE,deviation,MODE_LOWER
         ZZ = iCustom(Symbol(),0,"ZigZag",Depth, Deviation, BackStep,0,1);
         if(enveUP > 0 && enveDW > 0 && ZZ > 0) open = Open[0];
      }
     CountTrades(OP_BUY, Magic);// Вызов функции
  }


//+------------------------------------------------------------------+
int CountTrades(int type,int magic)
  {
   int count = 0;
   for(int i = OrdersTotal()-1; i>=0; i--)
     {
      if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES))
        {
         if(OrderSymbol()==Symbol() && (OrderType()==type || type==-1) && (OrderMagicNu
            count++;
        }
     }
   return(count);
  }
```

Эта функция отработает с переданными ей параметрами OP_BUY и Magic

Идем дальше.

Int count = 0; Объявляем целочисленную переменную, которая будет работать только в пределах нашей функции. В начале она равна нулю, а к концу выполнения функции в ней будет записано число, означающее, сколько открытых ордеров имеется на счете по заданным параметрам.

Вот мы и подошли вплотную к нашему первому циклу – for. Цикл содержит три оператора (может быть и больше, F1 в помощь). **int i = OrdersTotal ()** – переменная **i** равняется **OrdersTotal ()** – общему количеству открытых ордеров на счете – не важно кем и какого типа. Это **ВСЕГО** ордеров – рыночных, отложенных, ручных, установленных советниками. Затем, оператор **i> =0;** означает, что цикл for будет выполняться до тех пор, пока i будет больше или равно нулю. Т.е., пока есть ордера ордера на счете. i – означает, что с каждым проходом цикла (итерацией), переменная i будет уменьшаться на единицу. Получается так: сначала i равна общему количеству ордеров на счете, после каждого прохода i уменьшается на единицу и цикл будет работать до тех пор, пока i не станет равным нулю.

Следующий оператор **if (OrderSelect (i, SELECT_BY_POS, MODE_TRADES))** —

выбор ордеров по порядку (перебор). В OrderSelect передаем нашу i, т. е. перебираем все открытые номера на счете. Подробнее про команду OrderSelect почитайте в справке – F1. Думаю, нет смысла сюда переписывать то, что там указано.

В круглых скобках нашего if происходит «отсеивание» не нужных для подсчета ордеров. Снова работает незаменимый оператор if – если:

OrderSymbol () ==Symbol () — символ (т. е. торговый инструмент) равен текущему, т. е. тому, на график которого установлен советник && (и)

(OrderType () ==type || type== -1) — тип ордера равен тому, который мы передали в аргументы функции **ИЛИ** (|| – этот знак означает ИЛИ) **type** равен -1 (для универсальности функции мы можем передать в функцию -1, тогда функция посчитает ордера НЕ ЗАВИСИМО какого они типа – buy, sell, limit или stop) && **(OrderMagicNumber () ==magic || magic==-1)** – магик перебираемого ордера равен тому, который мы передали в аргументы функции ИЛИ (||) переданный параметр равен -1 – тогда функция посчитает ордера с любым магиком.

Если какой-то ордер прошел все «проверки» оператором if, строка count++ прибавит к нашей переменной count единицу (инкремент «++» прибавляет к переменной единицу, а инкремент « – » вычитает единицу).

Таким образом, после проверки циклом for всех ордеров и отсеивания «чужих», на выходе, оператором return (count) наша функция вернет количество открытых нашим советником ордеров. Надеюсь, объяснил понятно.

Уберем из нашего кода строчку Comment, где мы проверяли получение данных из индикаторов. На данный момент, наш код должен иметь такой вид и компилироваться без ошибок:

```
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
//| MyFirstEA.mq4 |
//| Copyright 2017, |
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
#property copyright «Copyright 2017»
#property link»»
#property version «1.00»
#property strict
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
input int period = 14;
input double deviation = 0.10;
sinput string zz = «Настройка ZZ»;
input int Depth = 12;
input int Deviation = 5;
input int BackStep = 3;
input int Magic = 254521;
double enveUP, enveDW, ZZ;
datetime open;
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
int OnInit ()
{
return (INIT_SUCCEEDED);
}

void OnDeinit (const int reason)
{

}
void OnTick ()
{
if (Open [0]!= open)
{
enveUP = iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_UPPER,1);
enveDW = iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_LOWER,1);
ZZ = iCustom (Symbol (),0,«ZigZag», Depth, Deviation, BackStep,0,1);
if (enveUP> 0 && enveDW> 0 && ZZ> 0) open = Open [0];
}
}
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
int CountTrades (int type, int magic)
{
int count = 0;
for (int i = OrdersTotal () -1; i> =0; i – )
{
if (OrderSelect (i, SELECT_BY_POS, MODE_TRADES))
{
```

```
if (OrderSymbol () ==Symbol () && (OrderType () ==type || type==-1) && (OrderMagicNumber ()
==magic || magic==-1))
count++;
}
}
return (count);
}
```

Функция открытия ордеров

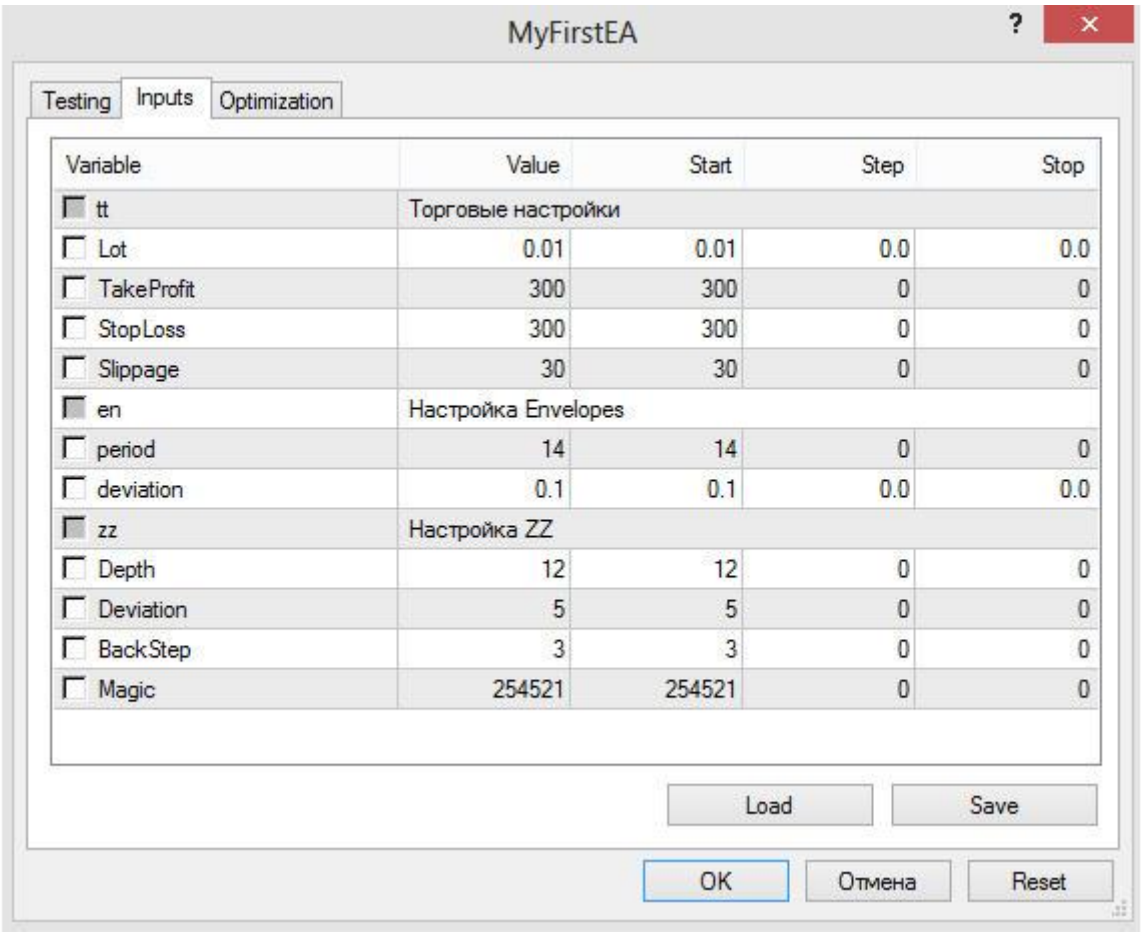
Ну вот мы уже подошли к самому сокровенному – открытию ордеров. Реализуем это отдельной функцией. Для начала, добавим во внешние переменные необходимые нам параметры: размер лота, стоп-лосса, тейк-профита, уровень проскальзывания.

Чтобы поля настроек были логически отделены друг от друга обозначим «Торговые настройки» и выделим настройки индикатора Envelopes в отдельные, так сказать, блоки. Теперь, код наших внешних параметров должен выглядеть так:

Количество пунктов указываем для 5-значного потока котировок. Если кто-то работаем с 4-значными, размеры стоп-лосса, тейк-профита и проскальзывания нужно разделить на 10. Стоп-лосс получится 30, тейк-профит 30, проскальзывание – 3. Или можно в блоке OnInit () реализовать автоподстройку под нужное количество знаков после занятой. Но, я полагаю, это не слишком важная деталь для разработки нашего первого учебного советника.

```
//+------------------------------------------
sinput string tt = "Торговые настройки";
input double Lot = 0.01;
input int TakeProfit = 300;
input int StopLoss   = 300;
input int Slippage   = 30;
sinput string en = "Настройка Envelopes";
input int period = 14;
input double deviation = 0.10;
sinput string zz = "Настройка ZZ";
input int  Depth = 12;
input int  Deviation = 5;
input int  BackStep = 3;
input int Magic = 254521;


double enveUP, enveDW, ZZ;
datetime open;
//+------------------------------------------
int OnInit()
```

Открыв свойства нашего наполовину готового советника, у нас получается такая картина:
Перепишите эту функцию сразу после функции CountTrades:




```
void openOrders (int type, double lot)
{
int ticket;
double price;
if (type == OP_BUY) price = Ask;
if (type == OP_SELL) price = Bid;
if (price <= 0) return;
ticket=OrderSend (Symbol (),type, lot, price,30,0,0,»», Magic,0,clrLimeGreen);

if (ticket> 0)
{
//Устанавливаем Стоп-лосс и Тейк-профит для Бай-ордера
if (OrderSelect (ticket, SELECT_BY_TICKET, MODE_TRADES))
{
double sl, tp;
if (type==OP_BUY)
{
sl=OrderOpenPrice () – (StopLoss*_Point);
sl=NormalizeDouble (sl,_Digits);
tp=OrderOpenPrice () + (TakeProfit*_Point);
tp=NormalizeDouble (tp,_Digits);
}
if (type==OP_SELL)
{
sl=OrderOpenPrice () + (StopLoss*_Point);
sl=NormalizeDouble (sl,_Digits);
```

```
tp=OrderOpenPrice () – (TakeProfit*_Point);
tp=NormalizeDouble (tp,_Digits);
}
bool mod=false;
int count=0;
while (!mod)
{
mod=OrderModify (ticket, OrderOpenPrice (),sl, tp,0,clrYellow);
count++;
if (count> =100)
{
mod=true;
break;
}
}
}
}
}
```

Теперь будем разбираться, что же выполняет данная функция.

Первая строка – **void openOrders (int type, double lot):** void – означает, что наша функция не возвращает никаких данных. Она просто отработала, и на этом все.

OpenOrders – название функции. **(int type, double lot)** – аргументы функции. Наша функция будет принимать параметр ТИПА открываемого ордера (бай или селл) и размер лота, который мы задаем во внешних параметрах эксперта.

Объявляем переменные, необходимые для работы внутри функции:

int ticket; – в эту переменную будет записан **тикет** (уникальный номер ордера, который будет ему присвоен торговым сервером) после открытия.

double price; – переменная для цены. Ордера Бай открываются по цене АСК (Ask), ордера Селл открываются по цене БИД (Bid) – таково правило торговли. Разница между ценами Аск и Бид называется Spread – «**спред**».

if (type == OP_BUY) price = Ask; – если мы будем открывать Бай-ордер, запишем в переменную price текущую цену Аск.

if (type == OP_SELL) price = Bid; – если собираемся открывать Селл-ордер, в переменную запишем цену Бид.

if (price <= 0) return; – если по какой-то причине в переменную price ничего не записалось, внутренняя команда return прекратит работу всю работу функции и приказ на открытие ордера не отправится на сервер. Даже если приказ и отправится на сервер, ордер не будет исполнен, т. к. по нулевой цене это невозможно. К тому же, многочисленные бессмысленные обращения к серверу, заканчивающиеся ошибками, могут быть причиной блокировки торгового счета.

ticket=OrderSend (Symbol (),type, lot, price, Slippage,0,0,»», Magic,0,clrLimeGreen); – эта функция отправляет команду на сервер для открытия ордера.

Ticket – после открытия ордера сюда запишется его уникальный номер. Пока ордер не открыт, ticket равняется нулю.

OrderSend – открывает ордер (не забывайте про F1 – отметьте курсор на данной функции, нажмите F1 и ознакомьтесь с ее официальным описанием).

Рассмотрим аргументы функции OrderSend по порядку:

Symbol () – текущий торговый инструмент, по которому будет открыт ордер. Т.е. советник откроет ордер по тому торговому инструменту, на график которого этот советник установлен. Это параметр типа string. Если в функции OrderSend первым аргументом указать, например, «EURUSD», то, даже если советник прикрепить (установить) на график валютной пары EURGBP, ордер будет открыт по EURUSD. Внимание: в тестере стратегий терминала MetaTrader4 такой «фокус» не проходит. А вот в MetaTrader5 – проходит.

Type – тип открываемого ордера: бай или селл.

Lot – торговый лот.

Price – цена, по которой будет открыт ордер.

Slippage – максимальный уровень проскальзывания. Если выше – ордер открыт не будет.

Затем идут два нуля. Первый – на этом месте должен стоять Стоп-Лосс, на втором месте – Тейк-Профит. Для ECN-счетов, по крайней мере в настоящее время, при установке ордера «по-рынку» невозможно сразу же установить Стоп-Лосс и Тейк-Профит. Поэтому, мы здесь ставим нули. А стоп и тейк установим сразу после установки ордера путем его **модификации.**

«» – между этими кавычками можно написать комментарий к открытым ордера, он будет отображаться здесь:

Magic – каждому ордеру будет присвоен идентификатор Magic, который будет отличать ордер советника от других ордеров.

Следующий аргумент – срок истечения отложенного ордера. Для «рыночных» ордеров не используется, поэтому, ставим нуль.

Последний аргумент – цвет стрелки на уровне открытия ордера. Наберите clr и MetaEditor предложит вам на выбор множество вариантов цветов. Если написать clrNONE, стрелка не установится.

Как только сервер установит ордер, в переменную ticket запишется его уникальный номер (не путать с Magic).

Далее по коду:

если ticket больше нуля, значит, наш ордер установился.

if (OrderSelect (ticket, SELECT_BY_TICKET, MODE_TRADES)) – здесь мы выбираем наш ордер, используя тикет (SELECT_BY_TICKET – означает, что ордер выбираем по тикету).

Выбор ордера для дальнейшей с ним работы происходит в операторе if. А значит, если ордер выбран не будет, дальнейший код в фигурных скобках не выполнится.

double sl, tp; – объявляем переменные, в которые будем заносить ценовые уровни стоп-лосса и тейк-профита.

if (type==OP_BUY) – если собираемся открыть Бай-ордер, рассчитываем, на какой цене будет Стоп-Лосс и на какой цене Тейк-Профит.

sl=OrderOpenPrice () – (StopLoss*_Point); – означает, что от уровня установки ордера вычитаем (StopLoss*_Point), где штатная функция _Point содержит размер пункта текущего инструмента в валюте котировки. У 5-значного брокера этот параметр равен 0,00001 (у йеновых валютных пар – 0,001). По умолчанию, во внешних переменных указан Стоп-Лосс 300 пунктов. Выразим эти 300 пунктов в ценовом виде = 300 * 0,00001 = 0,003. Допустим, бай-ордер был открыт по цене 1,23548. Для получения цены, на которой нужно установить стоп-лосс, вычитаем из цены открытия ордера ценовое выражение стоп-лосса. 1,23548 – 0,003 = 1,23248 – это и есть тот ценовой уровень, на который нужно поставить стоп-лосс. С тейк-профитом ситуация наоборот.

sl=NormalizeDouble (sl,_Digits); – Приведем цену установки стоп-лосса в нормализованный вид. Иногда бывает, что при вычислении какого-либо ценового уровня, получается число типа

1,35458455555. Т.е. более 5 знаков после запятой. И при отправлении этой «цены» на сервер, будет ошибка и ничего не произойдет. Для приведения числа в «нужный» уровень, надо это число нормализовать с помощью функции **NormalizeDouble – о**кругление числа с плавающей точкой до указанной точности. Первый параметр – число, которое нужно нормализовать, второе число – количество знаков после запятой. В нашем случае, на месте второго аргумента установим **_Digits.** Эта штатная переменная хранит количество десятичных знаков после запятой, определяющее точность измерения цены символа текущего графика.

tp=OrderOpenPrice () + (TakeProfit*_Point); – тоже самое, только с тейк-профитом.

tp=NormalizeDouble (tp,_Digits); – нормализация цены.

if (type==OP_SELL): следующий блок – если нам надо открыть Селл-ордер. Этот блок вычисляет уровни цен для стопа и тейка для селл-ордера. Принцип расчета такой же, как и при бай-ордере, только адаптивно в селл-ордеру.

После того, как советник рассчитает и запишет в переменные уровни для стоп-лосса и тейк-профита для нужного типа ордера, переходим к разбору следующего кода:

bool mod=false; – переменная, будем использовать ее цикле while.

int count=0; – еще одна переменная для использования в цикле while.

while (!mod) – объявляем цикл while. Он будет «крутиться» до тех пор, пока mod == false. Знак! перед bool-переменной означает НЕ. Т. е. Пока Не-mod (аналог mod == false), то цикл работает.

mod=OrderModify (ticket, OrderOpenPrice (),sl, tp,0,clrYellow); – функция модификации ордера. Разберем что она возвращает и ее аргументы.

OrderModify – возвращает true в случае успешной модификации ордера, и false – в случае неудачи. Получается, если функция модифицирует наш ордер, то в переменную mod запишется true и цикл прекратит работу.

Так как этот код у нас находится внутри условия if (OrderSelect (ticket, SELECT_BY_TICKET, MODE_TRADES)), то мы уже работаем с выбранным по тикету ордером.

Первым аргументом функции OrderModify является параметр ticket – уникальный номер ордера, который советник собирается модифицировать.

OrderOpenPrice () – цена открытия ордера. Параметр можно менять (модифицировать) для отложенных ордеров. Здесь мы ставим OrderOpenPrice (), что означает уже имеющуюся цену на этом ордере, она не изменится.

Далее ставим наши переменные – сначала стоп-лосс (sl), потом тейк-профит (tp).

После тейк-профита параметр времени истечения ордера. Мы его не меняем и ставим нуль.

Последний аргумент – цвет стрелки.

Count++; – Если с первого раза модификация ордера не удалась, прибавляем к переменной count единицу. Далее проверяем, какое значение имеет эта count:

if (count> =100). Если count меньше 100, то цикл выполнятся еще раз. Если снова неудача, опять count++ и проверка if (count> =100). По сути, мы даем функции выполнится 100 раз. И, если за 100 попыток модификации ордера ничего не получилось, выполняется следующий код:

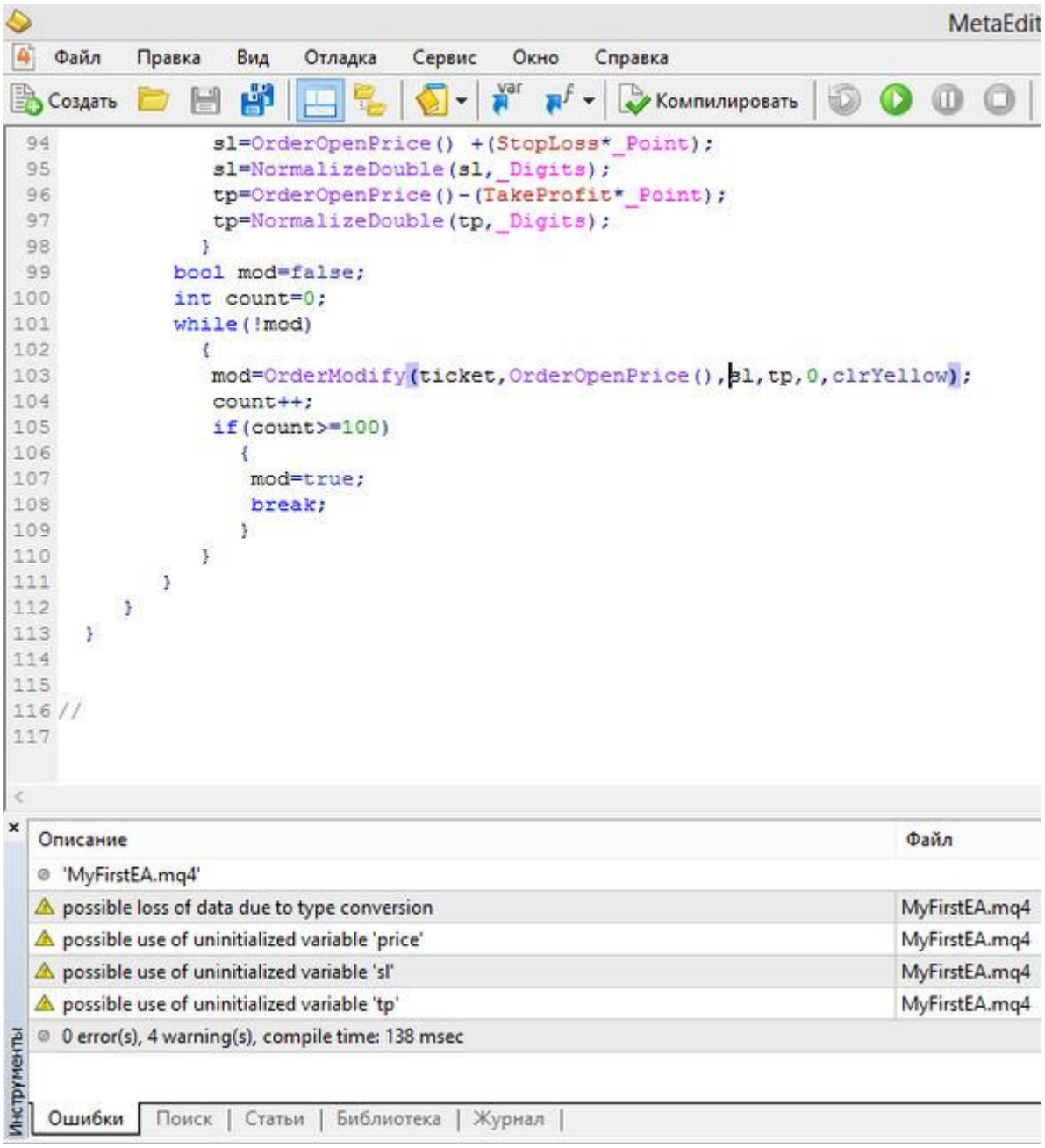
mod=true; – делаем mod – true. Цикл while должен прекратиться.

Break; – и на всякий случай пишем break – прекращение цикла.

Этим мы обезопасим себя от бесконечного цикла, когда в расчете стопа или тейка что-то сделано не правильно и советник бесконечно пытается с этими не правильными параметрами изменить ордер. В нашем случае у советника будет только 100 попыток.

На этом разбор функции открытия и модификации ордера закончен! Теперь смело нажмите кнопку Компиляция и, если вы все правильно переписали, советник скомпилируется:

Теперь давайте откроем какой-нибудь ордер!



В блоке OnTick () перед закрывающей блок скобкой сделаем вызов нашей свежеиспеченной функции с целью открытия бай-ордера:

openOrders (OP_BUY, Lot) – передаем тип ордера и размер лота из внешней переменной Lot.

Должно получиться вот так вот:

```
void OnTick ()
{
if (Open [0]!=open)
{
enveUP=iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_UPPER,1);
enveDW=iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_LOWER,1);
ZZ=iCustom (Symbol (),0,«ZigZag», Depth, Deviation, BackStep,0,1);
openOrders (OP_SELL, Lot);
if (enveUP> 0 && enveDW> 0 && ZZ> 0) open=Open [0];
}
```

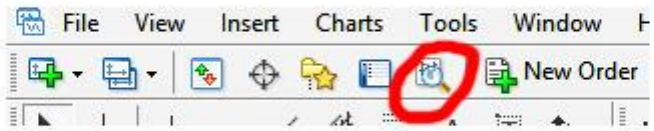
openOrders (OP_BUY, Lot);//вызов функции открытия и модификации ордера

}

Пора попробовать работу нашего советника. В редакторе кода нажимаем F4 или эту кнопку:

В открывшемся торговом терминале вызовем тестер стратегий сочетанием клавиш ctrl+R или кнопкой:

В появившемся окне тестера стратегий нужно выбрать наш советник, отметить галкой Visual mode (визуализация), выбрать любой период работы и любой торговый инструмент:

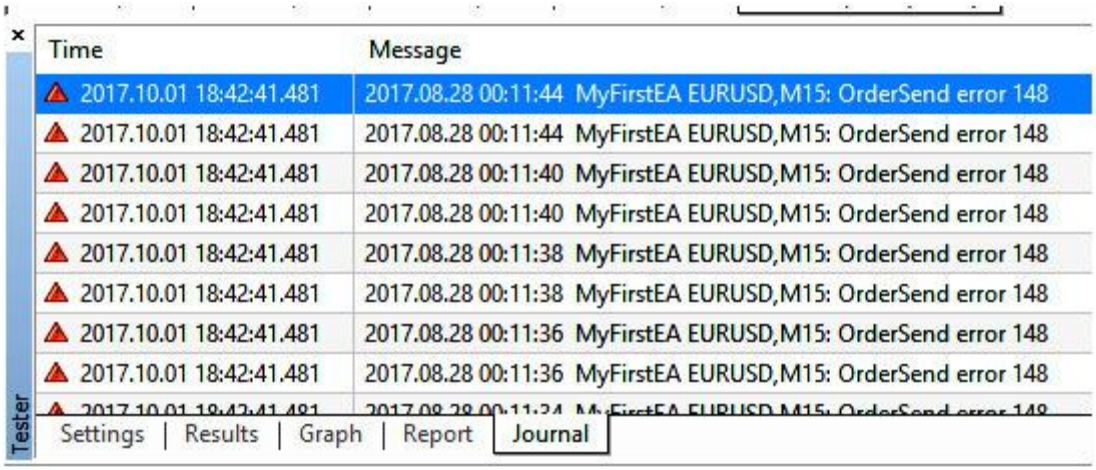


Далее нажимаем кнопку Start и видим такую картину:

Советник открывает множество ордеров! Что мы видим еще? У каждого ордера установлены стоп и тейк! На вкладке Result мы видим последовательные открытия ордеров sell и за каждым открытием ордера – его модификацию, т. е. установку стоп-лосса и тейк-профита:

На вкладке Journal (Журнал) видим массу ошибок:

Переходим в наш редактор кода, нажимаем F1 и в строке поиска открывшегося справочника пишем 148 (код ошибки):



И читаем: количество открытых ордеров достигло предела и бла-бла-бла. Ну и отлично! Главное, мы сделали функцию открытия и модификации ордера!

Открываем ордера по техническому заданию

Посмотри еще раз, что требует от нас техническое задание:

– Если верхний пик индикатора ZigZag (далее – ZZ) сформировался выше верхней линии индикатора Envelopes (с параметром Shift = 10, остальные – стандартные), выставляем ордер на продажу фиксированным лотом, определенным в настройках советника.

– Если нижний пик ZZ сформировался ниже нижней Envelopes – сигнал на покупку (т. е. наоборот от buy-сигнала).

Переходим в блок OnInit () и после получения данных индикаторов пишем:

if (CountTrades (OP_BUY, Magic) == 0 && Ask <enveDW && ZZ <enveDW) openOrders (OP_BUY, Lot); //Бай-ордер

«Расшифровка кода»: Если ордеров типа БАЙ нет (равно нулю) и цена Аск меньше Нижней линии Envelopes и ZZ меньше Envelopes открываем ордера типа БАЙ лотом, указанным в настройках.

Затем, зеркальный код для ордера типа СЕЛЛ:

if (CountTrades (OP_SELL, Magic) == 0 && Bid> enveUP && ZZ> enveUP) openOrders (OP_SELL, Lot); // Селл ордер

В итоге, ваш блок OnInit () должен выглядеть таким образом:

void OnTick ()

{

if (Open [0]!=open)

{

enveUP=iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_UPPER,1);

enveDW=iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_LOWER,1);

ZZ=iCustom (Symbol (),0,«ZigZag», Depth, Deviation, BackStep,0,1);

//openOrders (OP_SELL, Lot); – это нужно убрать

if (enveUP> 0 && enveDW> 0 && ZZ> 0) open=Open [0];

}

if (CountTrades (OP_BUY, Magic) == 0 && Ask <enveDW && ZZ <enveDW) openOrders (OP_BUY, Lot);

if (CountTrades (OP_SELL, Magic) == 0 && Bid> enveUP && ZZ> enveUP) openOrders (OP_SELL, Lot);

}

Ваш блок OnInit () должен выглядеть вот так:

Теперь запускаем тестер стратегий в режиме визуализации, ставим его на паузу, накладываем индикаторы Envelopes (не забудьте про shift =10) и ZZ и снимаем тестер с паузы. Видим вот такую картину:

```
//+------------------------------------------------------------------+
void OnTick()
  {
   if(Open[0]!=open)
     {
      enveUP = iEnvelopes(NULL,0,period,MODE_SMA,10,PRICE_CLOSE,deviation,MODE_UPPER,1);
      enveDW = iEnvelopes(NULL,0,period,MODE_SMA,10,PRICE_CLOSE,deviation,MODE_LOWER,1);
      ZZ=iCustom(Symbol(),0,"ZigZag",Depth,Deviation,BackStep,0,1);
      //openOrders(OP_SELL,Lot);
      if(enveUP>0 && enveDW>0 && ZZ>0) open=Open[0];
     }
     if(CountTrades(OP_BUY,Magic) == 0 && Ask < enveDW && ZZ < enveDW ) openOrders(OP_BUY,Lot);
     if(CountTrades(OP_SELL,Magic) == 0 && Bid > enveUP && ZZ > enveUP ) openOrders(OP_SELL,Lot);
  }
//+------------------------------------------------------------------+
```

Сигнал на продажу отработан, ордер установлен, стоп-лосс и тейк-профит на месте. С бай-ордером ситуация зеркальная:

На данный момент рабочий код выглядит вот так:

```
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
```

```
//| MyFirstEA.mq4 |
//| Copyright 2017, |
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
#property copyright «Copyright 2017»
#property link»»
#property version «1.00»
#property strict
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
sinput string tt = «Торговые настройки»;
input double Lot = 0.01;
input int TakeProfit = 300;
input int StopLoss = 300;
input int Slippage = 30;
sinput string en = «Настройка Envelopes»;
input int period = 14;
input double deviation=0.10;
sinput string zz = «Настройка ZZ»;
input int Depth = 12;
input int Deviation= 5;
input int BackStep = 3;
input int Magic=254521;

double enveUP, enveDW, ZZ;
datetime open;
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
int OnInit ()
{
return (INIT_SUCCEEDED);
}
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
//| |
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
void OnDeinit (const int reason)
{

}
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
//| |
//+ – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – —— – +
void OnTick ()
{
if (Open [0]!=open)
{
enveUP = iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_UPPER,1);
enveDW = iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_LOWER,1);
ZZ=iCustom (Symbol (),0,«ZigZag», Depth, Deviation, BackStep,0,1);
if (enveUP> 0 && enveDW> 0 && ZZ> 0) open=Open [0];
}
if (CountTrades (OP_BUY, Magic) == 0 && Ask <enveDW && ZZ <enveDW) openOrders (OP_BUY, Lot);
if (CountTrades (OP_SELL, Magic) == 0 && Bid> enveUP && ZZ> enveUP) openOrders (OP_SELL, Lot);
}
```

```
//+ – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – +
int CountTrades (int type, int magic)
{
int count = 0;
for (int i = OrdersTotal () -1; i> =0; i – )
{
if (OrderSelect (i, SELECT_BY_POS, MODE_TRADES))
{
if (OrderSymbol () ==Symbol () && (OrderType () ==type || type==-1) && (OrderMagicNumber ()
==magic || magic==-1))
count++;
}
}
return (count);
}
//+ – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – +
//| |
//+ – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – –– – +
void openOrders (int type, double lot)
{
int ticket;
double price;
if (type == OP_BUY) price = Ask;
if (type == OP_SELL) price = Bid;
if (price <= 0) return;
ticket=OrderSend (Symbol (),type, lot, price, Slippage,0,0,»», Magic,0,clrLimeGreen);

if (ticket> 0)
{
//Устанавливаем Стоп-лосс и Тейк-профит для Бай-ордера
if (OrderSelect (ticket, SELECT_BY_TICKET, MODE_TRADES))
{
double sl, tp;
if (type==OP_BUY)
{
sl=OrderOpenPrice () – (StopLoss*_Point);
sl=NormalizeDouble (sl,_Digits);
tp=OrderOpenPrice () + (TakeProfit*_Point);
tp=NormalizeDouble (tp,_Digits);
}
if (type==OP_SELL)
{
sl=OrderOpenPrice () + (StopLoss*_Point);
sl=NormalizeDouble (sl,_Digits);
tp=OrderOpenPrice () – (TakeProfit*_Point);
tp=NormalizeDouble (tp,_Digits);
}
bool mod=false;
int count=0;
while (!mod)
{
mod=OrderModify (ticket, OrderOpenPrice (),sl, tp,0,clrYellow);
count++;
if (count> =100)
{
```

```
mod=true;
break;
}
}
}
}
}
```

Закрытие ордеров на противоположной линии Envelopes

Осталось сделать четвертую часть технического задания: добавить возможность закрывать ордера при касании ценой противоположной линии Envelopes. Эту функцию можно выключать в настройках.

Для этого напишем отдельную функцию закрытия ордеров:

```
//Функция закрытия ордеров
void closeOrders (int type, int magic)
{
for (int i=OrdersTotal () -1; i> =0; i – )
{
if (OrderSelect (i, SELECT_BY_POS, MODE_TRADES))
{
if (OrderMagicNumber () ==magic || magic==-1)
{
if (OrderSymbol () ==Symbol () && (OrderType () ==type || type==-1))
{
if (OrderType () ==OP_BUY)
{
OrderClose (OrderTicket (),OrderLots (),Bid, Slippage, clrAqua);
}
if (OrderType () ==OP_SELL)
{
OrderClose (OrderTicket (),OrderLots (),Ask, Slippage, clrAqua);
}
}
}
}
}
}
//+ – — – — – — – — — – — – — – — – — – — – — – — – — – — – — – — +
```

Все как и раньше – в цикле перебираем ордера, отсеиваем не нужные, если они есть на счете и командой OrderClose закрываем. Ордера Бай закрываются по цене Бид (легко запомнить – Бай по Бид, две буквы Б) и ордера Селл закрываются по цене Аск. Поэтому, в функции мы это учли.

Теперь, перед закрывающей скобкой блока OnInit () реализуем вызов функции закрытия ордеров:

```
if (CountTrades (OP_BUY, Magic)> 0 && Bid> enveUP) closeOrders (OP_BUY, Magic);
if (CountTrades (OP_SELL, Magic)> 0 && Ask <enveDW) closeOrders (OP_SELL, Magic);
```

Сначала мы проверяем, если ли вообще ордера данного типа. Если бай-ордер существует и цена Бид поднялась ниже канала Envelopes – закрываем этот ордер. И наоборот: если существует селл-ордер и цена Аск опустилась ниже нижнего канала Envelopes – этот селл ордер закрываем.

На этом разработка нашего учебного советника закончена.

Заключение

Информации, изложенной в этой тоненькой книге для старта вам хватит за глаза. Я намеренно старался не использовать заумных и специфических терминов, не вставлял в книгу блоки текста из официальной документации языка программирования mql4. Кстати, программисты-монстры mql4 даже не считают языком программирования ввиду его крайней простоты и узкости применения. Отчасти, это так и есть. Однако, со своей задачей mql4 вполне справляется и спасибо разработчикам компании MetaQuotes за него!

Мой путь как программиста начался более 6 лет назад именно с разработки торговых роботов для терминала MetaTrader4. После mql4 я изучил mql5, Pyton, Java, PHP и кардинально сменил род своей деятельности с юриста-гуманитария на совершенно противоположную, которая мне ужасно нравится и приносит гораздо больше дохода.

Может быть, для кого-то из вас эта книга станет отправной точкой в новую профессию и новую жизнь!

Полный код советника

```
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
//| MyFirstEA.mq4 |
//| Copyright 2017, |
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
#property copyright «Copyright 2017»
#property link»»
#property version «1.00»
#property strict
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
sinput string tt = «Торговые настройки»;
input double Lot = 0.01;
input int TakeProfit = 300;
input int StopLoss = 300;
input int Slippage = 30;
sinput string en = «Настройка Envelopes»;
input int period = 14;
input double deviation=0.10;
sinput string zz = «Настройка ZZ»;
input int Depth = 12;
input int Deviation= 5;
input int BackStep = 3;
input int Magic=254521;

double enveUP, enveDW, ZZ;
datetime open;
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
int OnInit ()
{
return (INIT_SUCCEEDED);
}
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
//| |
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
void OnDeinit (const int reason)
{

}
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
//| |
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
```

```
void OnTick ()
{
if (Open [0]!=open)
{
enveUP = iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_UPPER,1);
enveDW = iEnvelopes (NULL,0,period, MODE_SMA,10,PRICE_CLOSE, deviation, MODE_LOWER,1);
ZZ=iCustom (Symbol (),0,«ZigZag», Depth, Deviation, BackStep,0,1);
if (enveUP> 0 && enveDW> 0 && ZZ> 0) open=Open [0];
}
if (CountTrades (OP_BUY, Magic) ==0 && Ask <enveDW && ZZ <enveDW) openOrders (OP_BUY, Lot);
if (CountTrades (OP_SELL, Magic) ==0 && Bid> enveUP && ZZ> enveUP) openOrders (OP_SELL, Lot);

if (CountTrades (OP_BUY, Magic)> 0 && Bid> enveUP) closeOrders (OP_BUY, Magic);
if (CountTrades (OP_SELL, Magic)> 0 && Ask <enveDW) closeOrders (OP_SELL, Magic);
}
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
int CountTrades (int type, int magic)
{
int count = 0;
for (int i = OrdersTotal () -1; i> =0; i – )
{
if (OrderSelect (i, SELECT_BY_POS, MODE_TRADES))
{
if (OrderSymbol () ==Symbol () && (OrderType () ==type || type==-1) && (OrderMagicNumber () ==magic ‖ magic==-1))
count++;
}
}
return (count);
}
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
//| |
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
void openOrders (int type, double lot)
{
int ticket;
double price;
if (type == OP_BUY) price = Ask;
if (type == OP_SELL) price = Bid;
if (price <= 0) return;
ticket=OrderSend (Symbol (),type, lot, price, Slippage,0,0,»», Magic,0,clrLimeGreen);

if (ticket> 0)
{
//Устанавливаем Стоп-лосс и Тейк-профит для Бай-ордера
if (OrderSelect (ticket, SELECT_BY_TICKET, MODE_TRADES))
{
double sl, tp;
if (type==OP_BUY)
{
sl=OrderOpenPrice () – (StopLoss*_Point);
```

```
sl=NormalizeDouble (sl,_Digits);
tp=OrderOpenPrice () + (TakeProfit*_Point);
tp=NormalizeDouble (tp,_Digits);
}
if (type==OP_SELL)
{
sl=OrderOpenPrice () + (StopLoss*_Point);
sl=NormalizeDouble (sl,_Digits);
tp=OrderOpenPrice () – (TakeProfit*_Point);
tp=NormalizeDouble (tp,_Digits);
}
bool mod=false;
int count=0;
while (!mod)
{
mod=OrderModify (ticket, OrderOpenPrice (),sl, tp,0,clrYellow);
count++;
if (count> =100)
{
mod=true;
break;
}
}
}
}
}
//Функция закрытия ордеров
void closeOrders (int type, int magic)
{
for (int i=OrdersTotal () -1; i> =0; i – )
{
if (OrderSelect (i, SELECT_BY_POS, MODE_TRADES))
{
if (OrderMagicNumber () ==magic || magic==-1)
{
if (OrderSymbol () ==Symbol () && (OrderType () ==type || type==-1))
{
if (OrderType () ==OP_BUY)
{
OrderClose (OrderTicket (),OrderLots (),Bid, Slippage, clrAqua);
}
if (OrderType () ==OP_SELL)
{
OrderClose (OrderTicket (),OrderLots (),Ask, Slippage, clrAqua);
}
}
}
}
}
}
//+ – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – — – +
```