METATRADER 4 — EXAMPLES

PRICE ACTION. AUTOMATING THE ENGULFING PATTERN TRADING STRATEGY

22 September 2015, 12:49

2 6 659

DMITRY IGLAKOV

Introduction

All Forex traders come across the Price Action at some point. This is not a mere chart analysis technique but the entire system for defining the possible future price movement direction. In this article, we will analyze the Engulfing pattern and create an Expert Advisor which will follow this pattern and make relevant trading decisions based on it.

We have previously examined the automated trading with Price Action patterns, namely the Inside Bar trading, in the article Price Action. Automating the Inside Bar Trading Strategy.

Rules of the Engulfing Pattern

The Engulfing pattern is when the body and shadows of a bar completely engulf the body and shadows of the previous bar. There are two types of patterns available:

- **BUOVB** — Bullish Outside Vertical Bar;
- **BEOVB** — Bearish Outside Vertical Bar.

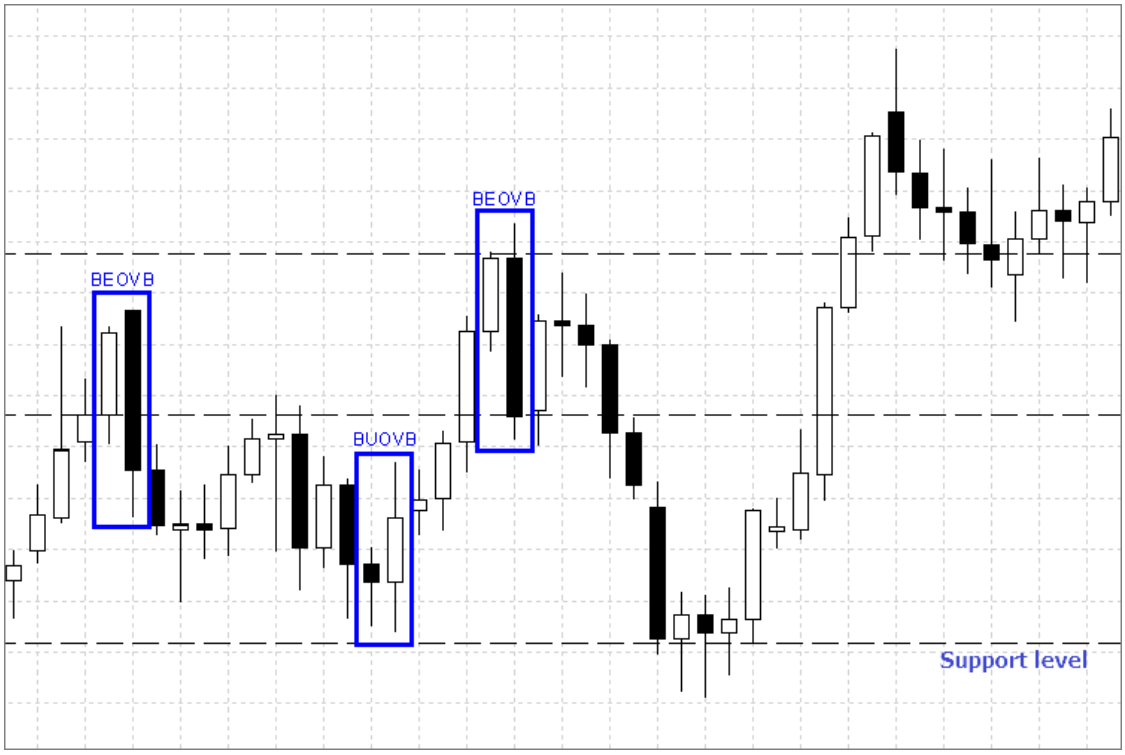

Fig. 1. Types of pattern shown on the chart

Let's have a closer look at this pattern.

BUOVB. The chart shows that the High of the outside bar is above the High of the previous bar, and the Low of the outside bar is below the Low of the previous one.

BEOVB. This pattern can also be easily identified on the chart. The High of the outside bar is above the High of the previous bar, and the Low of the outside bar is below the Low of the previous bar.

Their differences are that each pattern gives a clear understanding of the possible directions of the market.

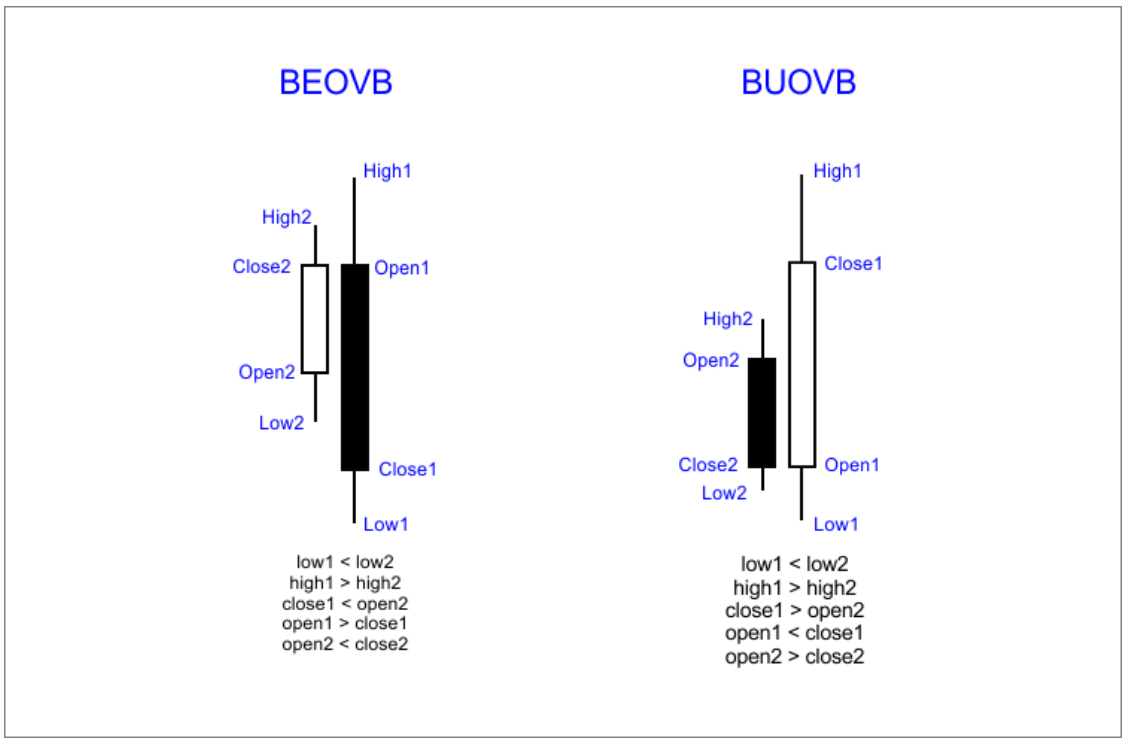

Fig. 2. Structure of the pattern

Rules of the Engulfing pattern:

- It is required to operate with this pattern on higher timeframes: H4, D1.
- For more refined entry, additional elements of graphical analysis should be applied, such as trend lines, support/resistance levels, Fibonacci levels, other Price Action patterns, etc.

Use pending orders to avoid premature or false market entries.
- Patterns repeated in flat trading should not be used as a signal for entering the market.

Establishing Entry Points for "BUOVB", Placing Stop Orders

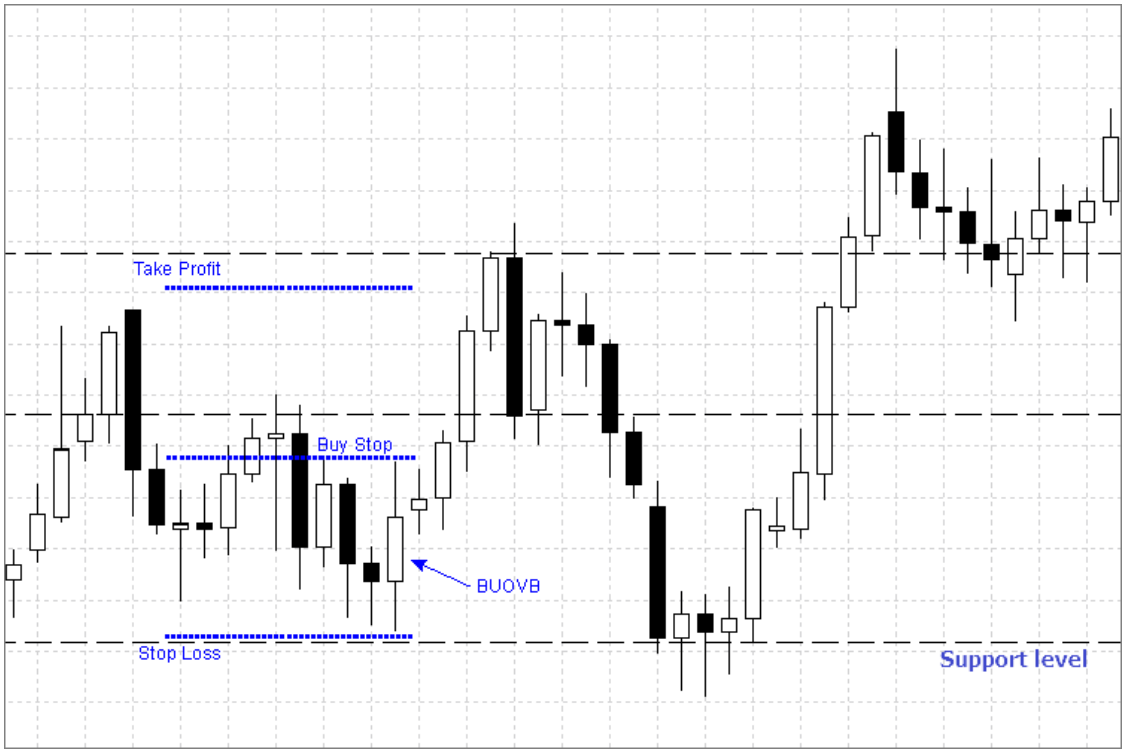

Fig. 3. Setting **Buy Stop** and stop orders

We will analyze the entry rules and stop orders placement for **BUOVB** (bullish outside vertical bar) using the example above:

1. We set Buy Stop pending order at a price slightly above the High price (by few points, for confirmation) of the outside bar.
2. Stop Loss level is set below the Low price of the outside bar.
3. And Take Profit level is set before it reaches the next resistance level.

Establishing Entry Points for "BEOVB", Placing Stop Orders

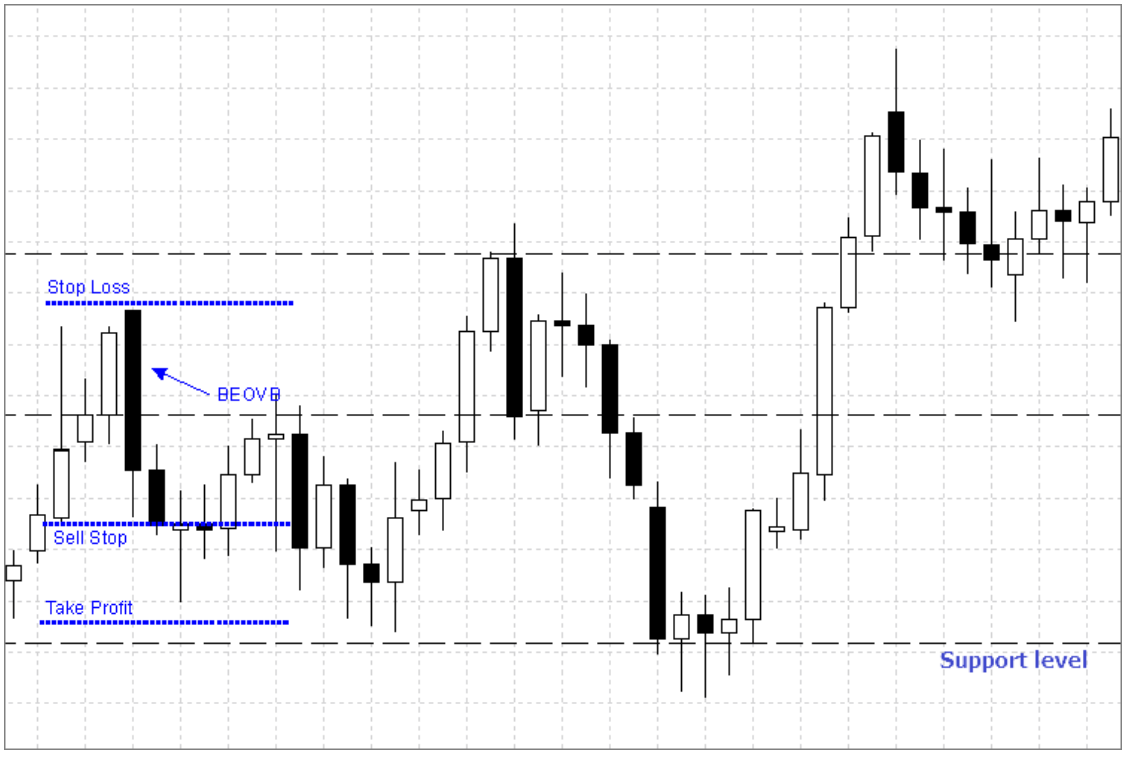

Fig. 4. Setting **Sell Stop** and stop orders

Let's examine the rules for entry and placement of stop orders for **BEOVB** (bearish outside vertical bar) from the example above:

1. We place the pending Sell Stop order at a price below the Low price (by few points, for confirmation) of an outside bar.
2. The Stop Loss level is set above the High price of the outside bar.
3. The Take Profit level is set before it reaches the next support level.

Creating an Expert Advisor for Trading the Engulfing Pattern

We reviewed the Engulfing pattern, learned how to enter the market safely, and also determined the levels of stop orders to limit losses or lock in profits.

Next we will try to implement the algorithms of an Expert Advisor and automate the Engulfing trading pattern.

We open **MetaEditor** from the **MetaTrader 4** terminal and create a new Expert Advisor (we will not go into details about creating Expert Advisors, as there is enough information available on the website). At the creation stage we leave all parameters blank. You can name them however you like. Eventually, you should get the following results:

```
//+------------------------------------------------------------------+
//|                                              BEOVB_BUOVB_Bar.mq4 |
//|                                Copyright 2015, Iglakov Dmitry. |
//|                                              cjdmitri@gmail.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, Iglakov Dmitry."
#property link      "cjdmitri@gmail.com"
#property version   "1.00"
#property strict
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---

  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

Converting the Pattern into MQL4 Algorithm

After creating an Expert Advisor we must define the Engulfing pattern after a candle is closed. For this purpose, we introduce new variables and assign values to them. See the code below:

```
//+------------------------------------------------------------------+
//|                                              BEOVB_BUOVB_Bar.mq4 |
//|                                Copyright 2015, Iglakov Dmitry. |
//|                                              cjdmitri@gmail.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, Iglakov Dmitry."
#property link      "cjdmitri@gmail.com"
#property version   "1.00"
#property strict

double open1,//first candle Open price
open2,      //second candle Open price
close1,     //first candle Close price
close2,     //second candle Close price
low1,       //first candle Low price
low2,       //second candle Low price
high1,      //first candle High price
high2;      //second candle High price
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

```
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- define prices of necessary bars
   open1        = NormalizeDouble(iOpen(Symbol(), Period(), 1), Digits);
   open2        = NormalizeDouble(iOpen(Symbol(), Period(), 2), Digits);
   close1       = NormalizeDouble(iClose(Symbol(), Period(), 1), Digits);
   close2       = NormalizeDouble(iClose(Symbol(), Period(), 2), Digits);
   low1         = NormalizeDouble(iLow(Symbol(), Period(), 1), Digits);
   low2         = NormalizeDouble(iLow(Symbol(), Period(), 2), Digits);
   high1        = NormalizeDouble(iHigh(Symbol(), Period(), 1), Digits);
   high2        = NormalizeDouble(iHigh(Symbol(), Period(), 2), Digits);
  }
//+------------------------------------------------------------------+
```

We find both types of the Engulfing pattern:

```
void OnTick()
  {
//--- define prices of necessary bars
   open1        = NormalizeDouble(iOpen(Symbol(), Period(), 1), Digits);
   open2        = NormalizeDouble(iOpen(Symbol(), Period(), 2), Digits);
   close1       = NormalizeDouble(iClose(Symbol(), Period(), 1), Digits);
   close2       = NormalizeDouble(iClose(Symbol(), Period(), 2), Digits);
   low1         = NormalizeDouble(iLow(Symbol(), Period(), 1), Digits);
   low2         = NormalizeDouble(iLow(Symbol(), Period(), 2), Digits);
   high1        = NormalizeDouble(iHigh(Symbol(), Period(), 1), Digits);
   high2        = NormalizeDouble(iHigh(Symbol(), Period(), 2), Digits);
//--- Finding bearish pattern BEOVB
   if(low1 < low2 &&// First bar's Low is below second bar's Low
      high1 > high2 &&// First bar's High is above second bar's High
      close1 < open2 && //First bar's Close price is below second bar's Open
      open1 > close1 && //First bar is a bearish bar
      open2 < close2)   //Second bar is a bullish bar
     {
      //--- we have described all conditions indicating that the first bar
completely engulfs the second bar and is a bearish bar
     }
```

The same way we find a bullish pattern:

```
//--- Finding bullish pattern BUOVB
   if(low1 < low2 &&// First bar's Low is below second bar's Low
      high1 > high2 &&// First bar's High is above second bar's High
      close1 > open2 && //First bar's Close price is higher than second
bar's Open
      open1 < close1 && //First bar is a bullish bar
      open2 > close2)   //Second bar is a bearish bar
     {
      //--- we have described all conditions indicating that the first bar
completely engulfs the second bar and is a bullish bar
     }
```

- We create customizable variables: stop orders, slippage, order expiration time, EA magic number, trading lot. *Stop loss can be omitted, as it will be set according to the pattern rules.*
- We introduce local variables to convert variables into a normal form.
- Furthermore, we bear in mind that stop orders are set at a certain distance from the bar's price values. In order to implement that, we add the **Interval** variable responsible for the interval between High/Low prices of bars and stop order levels, as well as pending order levels.
- We enter the variable **timeBUOVB_BEOVB** to prevent re-opening the order on this pattern.
- We enter the variable **bar1size** to check whether the outside bar is big enough. Thus, we can assume that the current market is not flat.

As a result, we obtain the following code:

```
//+------------------------------------------------------------------+
//|                                              BEOVB_BUOVB_bar.mq4 |
//|                                Copyright 2015, Iglakov Dmitry. |
//|                                              cjdmitri@gmail.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, Iglakov Dmitry."
#property link      "cjdmitri@gmail.com"
#property version   "1.00"
#property strict

extern int     interval         = 25;
//Interval
extern double  lot              = 0.1;                              //Lot
Size
extern int     TP               = 400;                              //Take
Profit
extern int     magic            = 962231;                           //Magic
number
extern int     slippage         =
2;             //Slippage
extern int     ExpDate          = 48;
//Expiration Hour Order
extern int     bar1size         = 900;                              //Bar 1
```

```
Size

double buyPrice,//define BuyStop setting price
buyTP,       //Take Profit BuyStop
buySL,       //Stop Loss BuyStop
sellPrice,   //define SellStop setting price
sellTP,      //Take Profit SellStop
sellSL;      //Stop Loss SellStop

double open1,//first candle Open price
open2,     //second candle Open price
close1,    //first candle Close price
close2,    //second candle Close price
low1,      //first candle Low price
low2,      //second candle Low price
high1,     //first candle High price
high2;     //second candle High price

datetime _ExpDate =0; // local variable for defining pending orders
expiration time
double _bar1size;// local variable required to avoid a flat market
datetime timeBUOVB_BEOVB;// time of a bar when pattern orders were opened, to
avoid re-opening
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   double   _bid = NormalizeDouble(MarketInfo (Symbol(), MODE_BID), Digits);
// define Low price
   double   _ask     = NormalizeDouble(MarketInfo(Symbol(), MODE_ASK),
Digits); //define High price
   double   _point   = MarketInfo(Symbol(), MODE_POINT);
//--- define prices of necessary bars
   open1        = NormalizeDouble(iOpen(Symbol(), Period(), 1), Digits);
   open2        = NormalizeDouble(iOpen(Symbol(), Period(), 2), Digits);
   close1       = NormalizeDouble(iClose(Symbol(), Period(), 1), Digits);
   close2       = NormalizeDouble(iClose(Symbol(), Period(), 2), Digits);
   low1         = NormalizeDouble(iLow(Symbol(), Period(), 1), Digits);
   low2         = NormalizeDouble(iLow(Symbol(), Period(), 2), Digits);
   high1        = NormalizeDouble(iHigh(Symbol(), Period(), 1), Digits);
   high2        = NormalizeDouble(iHigh(Symbol(), Period(), 2), Digits);
//---
   _bar1size=NormalizeDouble(((high1-low1)/_point),0);
//--- Finding bearish pattern BEOVB
   if(timeBUOVB_BEOVB!=iTime(Symbol(),Period(),1) && //orders are not yet
opened for this pattern
      _bar1size > bar1size && //first bar is big enough, so the market is not
flat
      low1 < low2 &&//First bar's Low is below second bar's Low
      high1 > high2 &&//First bar's High is above second bar's High
      close1 < open2 && //First bar's Close price is lower than second bar's
Open price
      open1 > close1 && //First bar is a bearish bar
      open2 < close2)   //Second bar is a bullish bar
     {
      //--- we have described all conditions indicating that the first bar
completely engulfs the second bar and is a bearish bar
      timeBUOVB_BEOVB=iTime(Symbol(),Period(),1); // indicate that orders are
already placed on this pattern
     }
//--- Finding bullish pattern BUOVB
   if(timeBUOVB_BEOVB!=iTime(Symbol(),Period(),1) && //orders are not yet
opened for this pattern
      _bar1size > bar1size && //first bar is big enough not to consider a
flat market
      low1 < low2 &&//First bar's Low is below second bar's Low
      high1 > high2 &&//First bar's High is above second bar's High
      close1 > open2 && //First bar's Close price is higher than second
bar's Open price
      open1 < close1 && //First bar is a bullish bar
      open2 > close2)   //Second bar is a bearish bar
     {
      //--- we have described all conditions indicating that the first bar
completely engulfs the second bar and is a bullish bar
      timeBUOVB_BEOVB=iTime(Symbol(),Period(),1); // indicate that orders are
already placed on this pattern
     }
  }
//+------------------------------------------------------------------+
```

Defining Stop Order Levels

We have fulfilled all the conditions and found high-quality patterns. Now it is necessary to set the stop order levels, pending order prices, as well as orders expiration date for each pattern.

Let's add the following code to the **OnTick()** function body:

```
//--- Define prices for placing orders and stop orders
```

```
   buyPrice =NormalizeDouble(high1 + interval * _point,Digits); //define a
price of order placing with intervals
   buySL =NormalizeDouble(low1-interval * _point,Digits); //define stop-loss
with an interval
   buyTP =NormalizeDouble(buyPrice + TP * _point,Digits); //define take
profit
   _ExpDate =TimeCurrent() + ExpDate*60*60; //pending order expiration time
calculation
//--- We also calculate sell orders
   sellPrice=NormalizeDouble(low1-interval*_point,Digits);
   sellSL=NormalizeDouble(high1+interval*_point,Digits);
   sellTP=NormalizeDouble(sellPrice-TP*_point,Digits);
```

Correction of Execution Errors

If you have ever engaged in the development of Expert Advisors, you probably know that errors often occur when closing and setting orders, including waiting time, incorrect stops, etc. To eliminate such errors, we should write a separate function with a small built-in handler of basic errors.

```
//+------------------------------------------------------------------
-----------------------------------------------+
//| The function opens or sets an
order
|
//| symbol      - symbol, at which a deal is
performed.
//| cmd         - a deal (may be equal to any of the deal
values).
//| volume      - amount of
lots.
|
//| price       - Open
price.
|
//| slippage    - maximum price deviation for market buy or sell
orders.
//| stoploss    - position close price when an unprofitability level is
reached (0 if there is no unprofitability level).|
//| takeprofit  - position close price when a profitability level is reached
(0 if there is no profitability level).     |
//| comment     - order comment. The last part of comment can be changed by
the trade server.
//| magic       - order magic number. It can be used as a user-defined
ID.
//| expiration  - pending order expiration
time.
//| arrow_color - open arrow color on a chart. If the parameter is absent or
equal to CLR_NONE,
//|               the open arrow is not displayed on a
chart.
//+------------------------------------------------------------------
-----------------------------------------------+
int OrderOpenF(string      OO_symbol,
               int         OO_cmd,
               double      OO_volume,
               double      OO_price,
               int         OO_slippage,
               double      OO_stoploss,
               double      OO_takeprofit,
               string      OO_comment,
               int         OO_magic,
               datetime    OO_expiration,
               color       OO_arrow_color)
  {
   int result = -1;// result of opening an order
   int Error = 0; // error when opening an order
   int attempt = 0; // amount of performed attempts
   int attemptMax = 3; // maximum amount of attempts
   bool exit_loop = false; // exit the loop
   string lang =TerminalInfoString(TERMINAL_LANGUAGE);// trading terminal
language for defining the language of the messages
   double stopllvl =NormalizeDouble(MarketInfo (OO_symbol, MODE_STOPLEVEL) *
MarketInfo (OO_symbol, MODE_POINT),Digits);// minimum stop loss/take profit
level, in points

//the module provides safe order opening
//--- checking stop orders for buying
   if(OO_cmd==OP_BUY || OO_cmd==OP_BUYLIMIT || OO_cmd==OP_BUYSTOP)
     {
      double tp = (OO_takeprofit - OO_price)/MarketInfo(OO_symbol,
MODE_POINT);
      double sl = (OO_price - OO_stoploss)/MarketInfo(OO_symbol, MODE_POINT);
      if(tp>0 && tp<=stopllvl)
        {
         OO_takeprofit=OO_price+stopllvl+2*MarketInfo(OO_symbol,MODE_POINT);
        }
      if(sl>0 && sl<=stopllvl)
        {
         OO_stoploss=OO_price -(stopllvl+2*MarketInfo(OO_symbol,MODE_POINT));
        }
     }
//--- checking stop orders for selling
   if(OO_cmd==OP_SELL || OO_cmd==OP_SELLLIMIT || OO_cmd==OP_SELLSTOP)
     {
      double tp = (OO_price - OO_takeprofit)/MarketInfo(OO_symbol,
MODE_POINT);
      double sl = (OO_stoploss - OO_price)/MarketInfo(OO_symbol, MODE_POINT);
      if(tp>0 && tp<=stopllvl)
        {
         OO_takeprofit=OO_price -
(stopllvl+2*MarketInfo(OO_symbol,MODE_POINT));
        }
      if(sl>0 && sl<=stopllvl)
        {
```

```
         OO_stoploss=OO_price+stoplvl+2*MarketInfo(OO_symbol,MODE_POINT);
        }
      }
//--- while loop
   while(!exit_loop)
     {
      result=OrderSend
(OO_symbol,OO_cmd,OO_volume,OO_price,OO_slippage,OO_stoploss,OO_takeprofit,OO_c
//attempt to open an order using the specified parameters
      //--- if there is an error when opening an order
      if(result<0)
        {
         Error = GetLastError();                                  //assign
a code to an error
         switch(Error)                                            //error
enumeration
           {                                                      //order
closing error enumeration and an attempt to fix them
            case  2:
               if(attempt<attemptMax)
                 {
                  attempt=attempt+1;                              //define
one more attempt
                  Sleep(3000);                                    //3
seconds of delay
                  RefreshRates();
                  break;                                          //exit
switch
                 }
               if(attempt==attemptMax)
                 {
                  attempt=0;                                      //reset
the amount of attempts to zero
                  exit_loop = true;                               //exit
while
                  break;                                          //exit
switch
                 }
            case  3:
               RefreshRates();
               exit_loop = true;                                  //exit
while
               break;                                             //exit
switch
            case  4:
               if(attempt<attemptMax)
                 {
                  attempt=attempt+1;                              //define
one more attempt
                  Sleep(3000);                                    //3
seconds of delay
                  RefreshRates();
                  break;                                          //exit
switch
                 }
               if(attempt==attemptMax)
                 {
                  attempt = 0;                                    //reset
the amount of attempts to zero
                  exit_loop = true;                               //exit
while
                  break;                                          //exit
switch
                 }
            case  5:
               exit_loop = true;                                  //exit
while
               break;                                             //exit
switch
            case  6:
               if(attempt<attemptMax)
                 {
                  attempt=attempt+1;                              //define
one more attempt
                  Sleep(5000);                                    //3
seconds of delay
                  break;                                          //exit
switch
                 }
               if(attempt==attemptMax)
                 {
                  attempt = 0;                                    //reset
the amount of attempts to zero
                  exit_loop = true;                               //exit
while
                  break;                                          //exit
switch
                 }
            case  8:
               if(attempt<attemptMax)
                 {
                  attempt=attempt+1;                              //define
one more attempt
                  Sleep(7000);                                    //3
seconds of delay
                  break;                                          //exit
switch
                 }
               if(attempt==attemptMax)
                 {
                  attempt = 0;                                    //reset
the amount of attempts to zero
                  exit_loop = true;                               //exit
while
                  break;                                          //exit
switch
                 }
            case 64:
               exit_loop = true;                                  //exit
```

```
while
                  break;                                                  //exit
switch
               case 65:
                  exit_loop = true;                                       //exit
while
                  break;                                                  //exit
switch
               case 128:
                  Sleep(3000);
                  RefreshRates();
                  continue;                                               //exit
switch
               case 129:
                  if(attempt<attemptMax)
                    {
                     attempt=attempt+1;                                   //define
one more attempt
                     Sleep(3000);                                         //3
seconds of delay
                     RefreshRates();
                     break;                                               //exit
switch
                    }
                  if(attempt==attemptMax)
                    {
                     attempt = 0;                                         //reset
the amount of attempts to zero
                     exit_loop = true;                                    //exit
while
                     break;                                               //exit
switch
                    }
               case 130:
                  exit_loop=true;                                         //exit
while
                  break;
               case 131:
                  exit_loop = true;                                       //exit
while
                  break;                                                  //exit
switch
               case 132:
                  Sleep(10000);                                           //sleep
for 10 seconds
                  RefreshRates();                                         //update
data
                  //exit_loop = true;                                     //exit
while
                  break;                                                  //exit
switch
               case 133:
                  exit_loop=true;                                         //exit
while
                  break;                                                  //exit
switch
               case 134:
                  exit_loop=true;                                         //exit
while
                  break;                                                  //exit
switch
               case 135:
                  if(attempt<attemptMax)
                    {
                     attempt=attempt+1;                                   //define
one more attempt
                     RefreshRates();
                     break;                                               //exit
switch
                    }
                  if(attempt==attemptMax)
                    {
                     attempt = 0;                                         //set the
number of attempts to zero
                     exit_loop = true;                                    //exit
while
                     break;                                               //exit
switch
                    }
               case 136:
                  if(attempt<attemptMax)
                    {
                     attempt=attempt+1;                                   //define
one more attempt
                     RefreshRates();
                     break;                                               //exit
switch
                    }
                  if(attempt==attemptMax)
                    {
                     attempt = 0;                                         //set the
amount of attempts to zero
                     exit_loop = true;                                    //exit
while
                     break;                                               //exit
switch
                    }
               case 137:
                  if(attempt<attemptMax)
                    {
                     attempt=attempt+1;
                     Sleep(2000);
                     RefreshRates();
                     break;
                    }
                  if(attempt==attemptMax)
                    {
                     attempt=0;
                     exit_loop=true;
```

```
                  break;
                 }
            case 138:
               if(attempt<attemptMax)
                 {
                  attempt=attempt+1;
                  Sleep(1000);
                  RefreshRates();
                  break;
                 }
               if(attempt==attemptMax)
                 {
                  attempt=0;
                  exit_loop=true;
                  break;
                 }
            case 139:
               exit_loop=true;
               break;
            case 141:
               Sleep(5000);
               exit_loop=true;
               break;
            case 145:
               exit_loop=true;
               break;
            case 146:
               if(attempt<attemptMax)
                 {
                  attempt=attempt+1;
                  Sleep(2000);
                  RefreshRates();
                  break;
                 }
               if(attempt==attemptMax)
                 {
                  attempt=0;
                  exit_loop=true;
                  break;
                 }
            case 147:
               if(attempt<attemptMax)
                 {
                  attempt=attempt+1;
                  OO_expiration=0;
                  break;
                 }
               if(attempt==attemptMax)
                 {
                  attempt=0;
                  exit_loop=true;
                  break;
                 }
            case 148:
               exit_loop=true;
               break;
            default:
               Print("Error: ",Error);
               exit_loop=true; //exit while
               break;          //other options
            }
         }
      //--- if no errors detected
      else
        {
         if(lang == "Russian") {Print("Ордер успешно открыт. ", result);}
         if(lang == "English") {Print("The order is successfully opened.",
result);}
         Error = 0;                                    //reset the error code to
zero
         break;                                        //exit while
         //errorCount =0;                              //reset the amount of
attempts to zero
        }
     }
   return(result);
  }
//+------------------------------------------------------------------+
```

As a result, we obtain the following code:

```
//+------------------------------------------------------------------+
//|                                                BEOVB_BUOVB_bar.mq4 |
//|                                Copyright 2015, Iglakov Dmitry. |
//|                                                 cjdmitri@gmail.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, Iglakov Dmitry."
#property link      "cjdmitri@gmail.com"
#property version   "1.00"
#property strict

extern int     interval         = 25;
//Interval
extern double  lot              = 0.1;                          //Lot
Size
extern int     TP               = 400;                          //Take
Profit
extern int     magic            = 962231;                       //Magic
number
extern int     slippage         =
2;                               //Slippage
extern int     ExpDate          = 48;
//Expiration Hour Order
extern int     bar1size         = 900;                          //Bar 1
Size

double buyPrice,//define BuyStop price
```

```
buyTP,      //Take Profit BuyStop
buySL,      //Stop Loss BuyStop
sellPrice,  //define SellStop price
sellTP,     //Take Profit SellStop
sellSL;     //Stop Loss SellStop

double open1,//first candle Open price
open2,    //second candle Open price
close1,   //first candle Close price
close2,   //second candle Close price
low1,     //first candle Low price
low2,     //second candle Low price
high1,    //first candle High price
high2;    //second candle High price

datetime _ExpDate =0; // local variable for defining pending orders
expiration time
double _bar1size;// local variable required to avoid a flat market
datetime timeBUOVB_BEOVB;// time of a bar when pattern orders were opened, to
avoid re-opening
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   double   _bid = NormalizeDouble(MarketInfo (Symbol(), MODE_BID), Digits);
// define Low price
   double   _ask     = NormalizeDouble(MarketInfo(Symbol(), MODE_ASK),
Digits); //define High price
   double   _point   = MarketInfo(Symbol(), MODE_POINT);
//--- define prices of necessary bars
   open1        = NormalizeDouble(iOpen(Symbol(), Period(), 1), Digits);
   open2        = NormalizeDouble(iOpen(Symbol(), Period(), 2), Digits);
   close1       = NormalizeDouble(iClose(Symbol(), Period(), 1), Digits);
   close2       = NormalizeDouble(iClose(Symbol(), Period(), 2), Digits);
   low1         = NormalizeDouble(iLow(Symbol(), Period(), 1), Digits);
   low2         = NormalizeDouble(iLow(Symbol(), Period(), 2), Digits);
   high1        = NormalizeDouble(iHigh(Symbol(), Period(), 1), Digits);
   high2        = NormalizeDouble(iHigh(Symbol(), Period(), 2), Digits);

//--- Define prices for placing orders and stop orders
   buyPrice =NormalizeDouble(high1 + interval * _point,Digits); //define a
price of order placing with intervals
   buySL =NormalizeDouble(low1-interval * _point,Digits); //define stop loss
with an interval
   buyTP =NormalizeDouble(buyPrice + TP * _point,Digits); //define take
profit
   _ExpDate =TimeCurrent() + ExpDate*60*60; //pending order expiration time
calculation
//--- We also calculate sell orders
   sellPrice=NormalizeDouble(low1-interval*_point,Digits);
   sellSL=NormalizeDouble(high1+interval*_point,Digits);
   sellTP=NormalizeDouble(sellPrice-TP*_point,Digits);
//---
   _bar1size=NormalizeDouble(((high1-low1)/_point),0);
//--- Finding bearish pattern BEOVB
   if(timeBUOVB_BEOVB!=iTime(Symbol(),Period(),1) && //orders are not yet
opened for this pattern
      _bar1size > bar1size && //first bar is big enough, so the market is not
flat
      low1 < low2 &&//First bar's Low is below second bar's Low
      high1 > high2 &&//First bar's High is above second bar's High
      close1 < open2 && //First bar's Close price is lower than second bar's
Open price
      open1 > close1 && //First bar is a bearish bar
      open2 < close2)   //Second bar is a bullish bar
     {
      //--- we have described all conditions indicating that the first bar
completely engulfs the second bar and is a bearish bar
      OrderOpenF(Symbol(),OP_SELLSTOP,lot,sellPrice,slippage,sellSL,sellTP,NULL
,magic,_ExpDate,Blue);
      timeBUOVB_BEOVB=iTime(Symbol(),Period(),1); //indicate that orders are
already placed on this pattern
     }
//--- Finding bullish pattern BUOVB
   if(timeBUOVB_BEOVB!=iTime(Symbol(),Period(),1) && //orders are not yet
opened for this pattern
      _bar1size > bar1size && //first bar is big enough, so the market is not
flat
      low1 < low2 &&//First bar's Low is below second bar's Low
      high1 > high2 &&//First bar's High is above second bar's High
      close1 > open2 && //First bar's Close price is higher than second
bar's Open price
      open1 < close1 && //First bar is a bullish bar
      open2 > close2)   //Second bar is a bearish bar
     {
      //--- we have described all conditions indicating that the first bar
completely engulfs the second bar and is a bullish bar
      OrderOpenF(Symbol(),OP_BUYSTOP,lot,buyPrice,slippage,buySL,buyTP,NULL
,magic,_ExpDate,Blue);
      timeBUOVB_BEOVB = iTime(Symbol(),Period(),1); //indicate that orders
are already placed on this pattern
     }
  }
//+------------------------------------------------------------------+
```

Now, let's perform the compilation and check for error messages in the log.

Testing the Expert Advisor

It is time to test our Expert Advisor. Let's launch the Strategy Tester and set the input parameters.

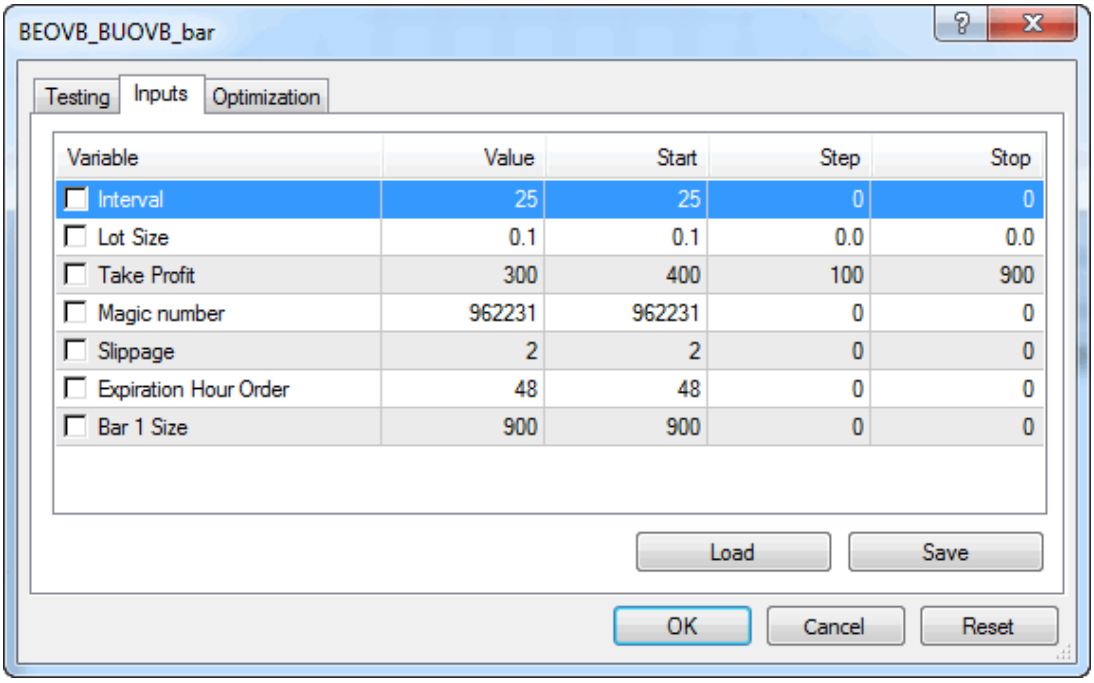

Fig. 5. Input Parameters for Testing

1. Choose a currency pair for testing. I chose **EURAUD.**
2. Make sure to set "Every tick" mode and define that testing is to be performed on history data. I have selected the entire year of 2014.
3. Set **D1** timeframe.
4. Launch the test.
5. After the test is complete, check the log. As we can see, no execution errors have occurred in the process.

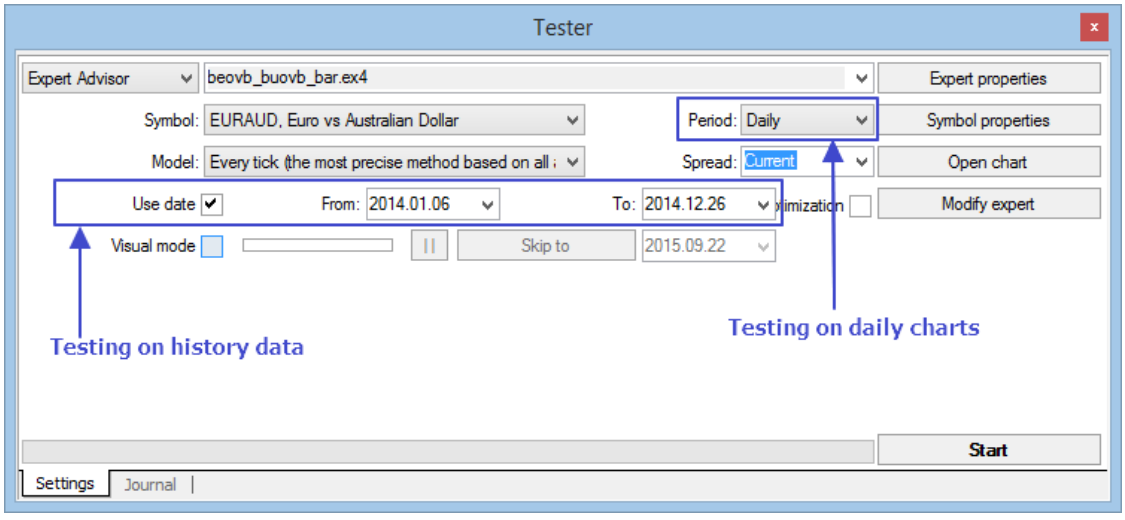

Fig. 6. Setting up testing conditions

Below is the EA testing journal:

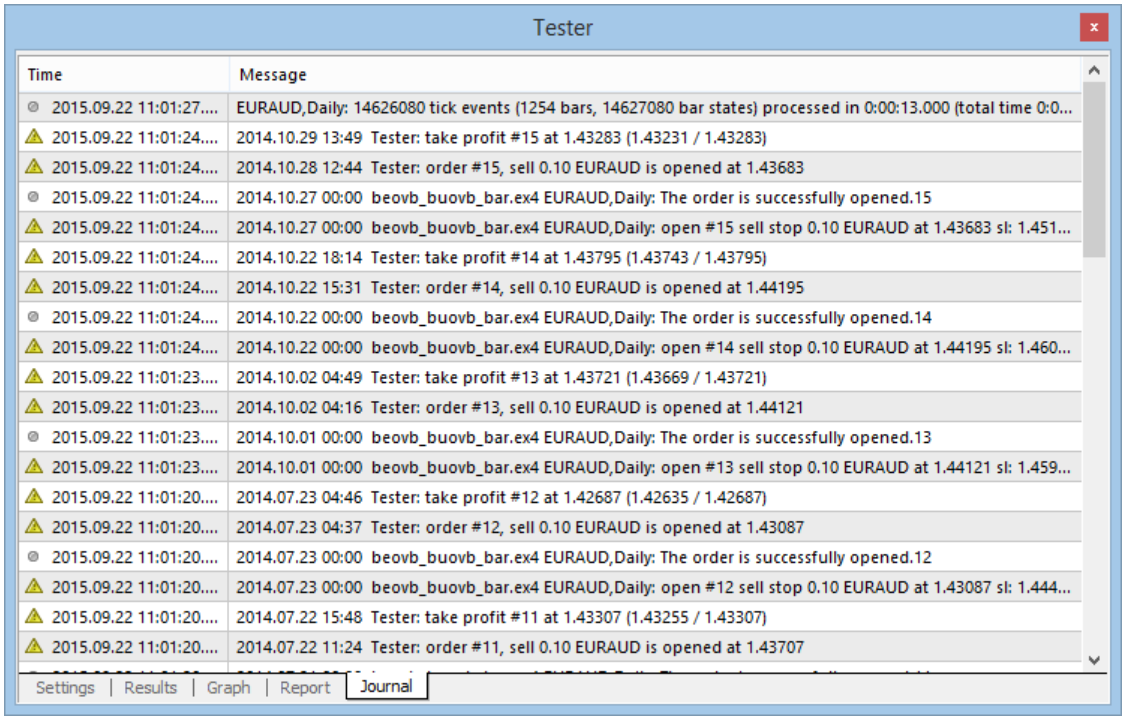

Fig. 7. Expert Advisor testing journal

Make sure there are no mistakes and optimize the EA.

Optimization

I have selected the following parameters for optimization:

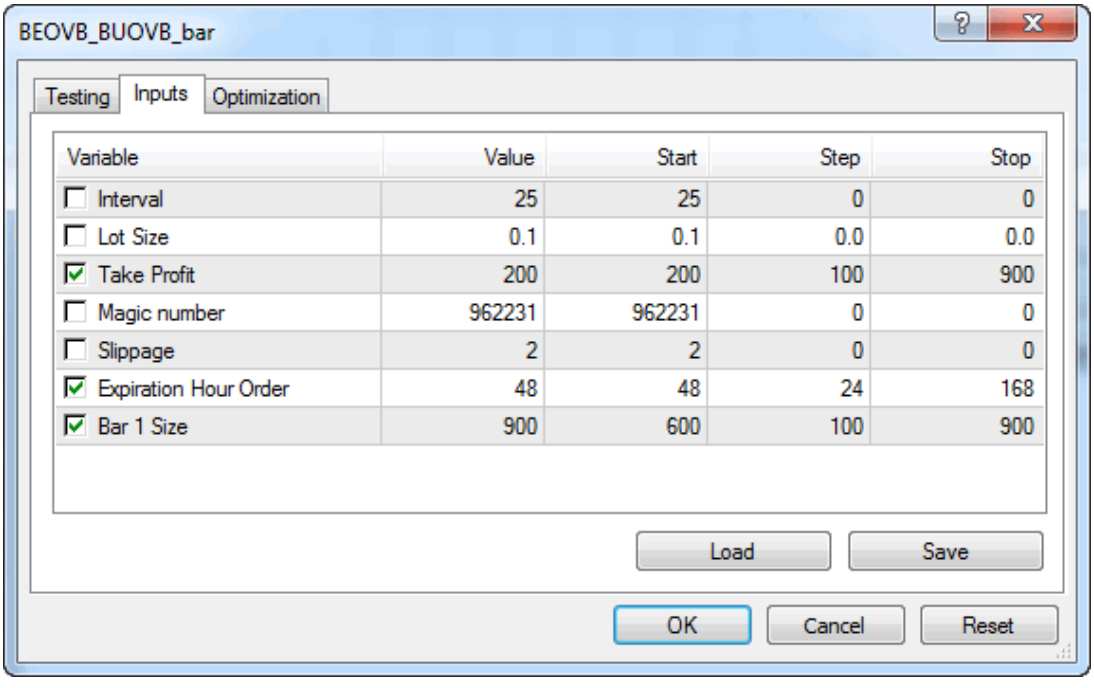

Fig. 8. Optimization parameters

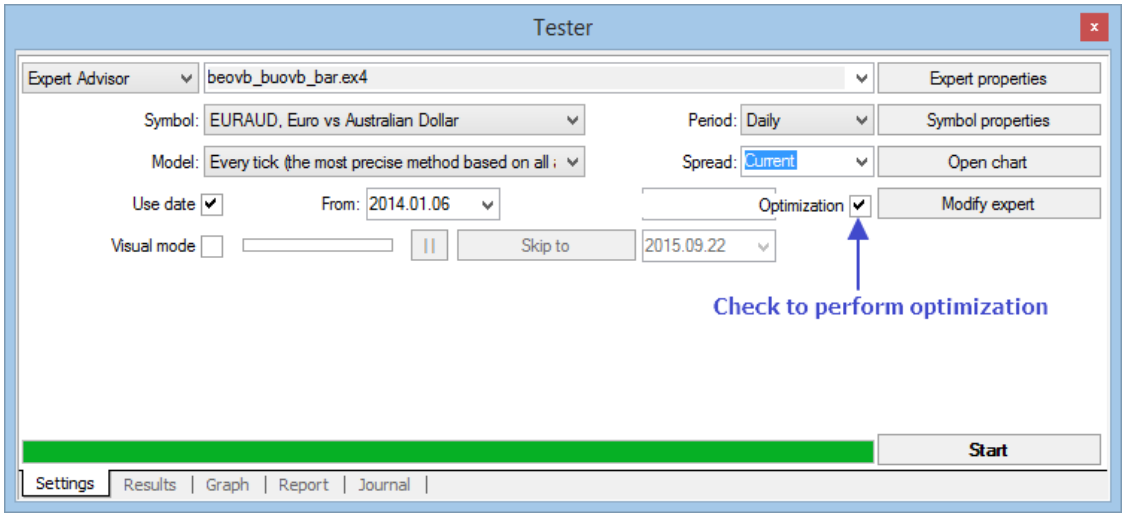

Fig. 9. Optimization settings

Thus, as a result of optimization and testing, we now have the ready-to-use robot.

Optimization and Testing Results

After the optimization of the most popular currency pairs, we obtain the following results:

Currency pair	Net profit	Profit factor	Drawdown (%)	Gross Profit	Gross loss
EURAUD	523.90$	3.70	2.13	727,98$	196.86$
USDCHF	454.19$	-	2.25	454.19$	0.00$
GBPUSD	638.71$	-	1.50	638.71$	0.00$
EURUSD	638.86$	-	1.85	638.86$	0.00$
USDJPY	423.85$	5.15	2.36	525.51$	102.08$
USDCAD	198.82$	2.41	2.74	379.08$	180.26$
AUDUSD	136.14$	1.67	2.39	339.26$	203.12$

Table 1. Optimization results

More detailed testing results were achieved on the currency pair **EURAUD:**

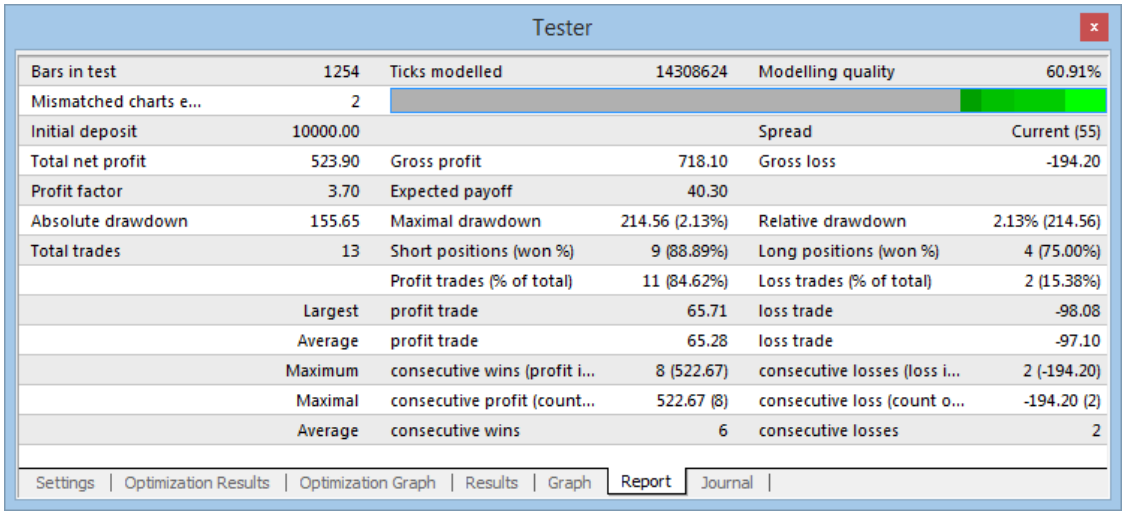

Fig. 10. Testing results

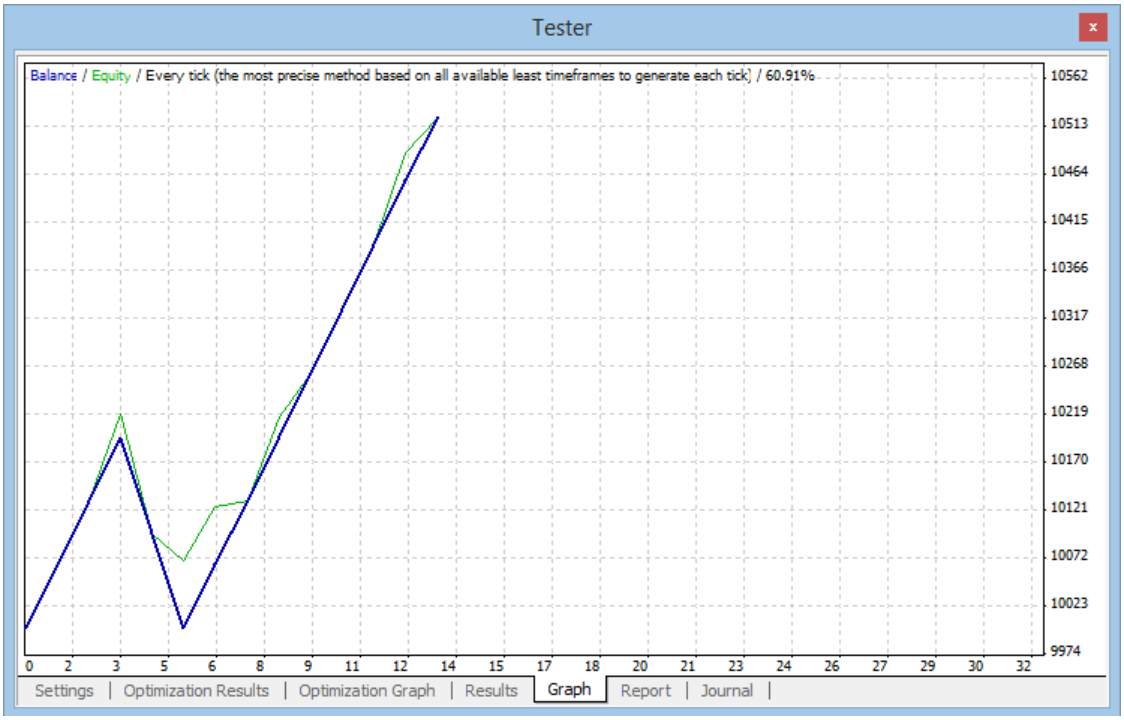

Fig. 11. Testing results chart

Conclusion

1. In this article, we have created an Expert Advisor trading the Engulfing pattern.
2. We made sure that **Price Action patterns can work** even with no additional market entry filters.
3. No tricks (like Martingale or averaging) have been used.
4. The drawdown has been minimized through the correct setting of the stop orders.
5. No technical indicators have been used. The EA was based solely on reading a "bare" chart.

Thank you for reading, and I hope you find this article helpful.

Translated from Russian by MetaQuotes Software Corp.
Original article: https://www.mql5.com/ru/articles/1946

Attached files | Download ZIP
beovb_buovb_bar.mq4 (40.66 KB)

Last comments | Go to discussion (2)

3rjfx | 23 Sep 2015 at 12:53
Very useful.. Thanks a lot.. :D

ander_assoc | 4 Oct 2015 at 22:01
spent the part of Saturday optimization and back testing on some favorite pairs and found some great results, but there is one thing there a lot of negs in a row I had one that went 70+ negs in a row and it still double the acct size in my 9 week back testing periods , can you think of a setting for I can optimize to bring the number of negs. in a row down. nice ea I would like test this a little bit more

How to Develop a Profitable Trading Strategy
This article provides an answer to the question: "Is it possible to formulate an automated trading strategy based on

history data with neural networks?".

Price Action. Automating the Inside Bar Trading Strategy

The article describes the development of a MetaTrader 4 Expert Advisor based on the Inside Bar strategy, including Inside Bar detection principles, as well as pending and stop order setting rules. Test and optimization results are provided as well.