MQL4 COURSE

By Coders' guru

www.forex-tsd.com

-16-

Your First Expert Advisor

Part 4

We have reached the edge of the moon in our way to the truth, I'm not under the impact of alcoholic poisoning (I don't drink at all), but I'm so happy to reach the last part of explaining or first expert advisor. **Yes**! This is the last part of the expert advisor lesson.

I hope you enjoyed the journey discovering how to write our simple yet important expert advisor.
Let's take the final step.

The code we have:

```
//+------------------------------------------------------------------+
//|                                                  My_First_EA.mq4 |
//|                                                      Coders Guru |
//|                                          http://www.forex-tsd.com |
//+------------------------------------------------------------------+
#property copyright "Coders Guru"
#property link      "http://www.forex-tsd.com"

//---- input parameters
extern double    TakeProfit=250.0;
extern double    Lots=0.1;
extern double    TrailingStop=35.0;
//+------------------------------------------------------------------+
//| expert initialization function                                   |
//+------------------------------------------------------------------+
int init()
  {
//----

//----
   return(0);
  }
//+------------------------------------------------------------------+
//| expert deinitialization function                                 |
//+------------------------------------------------------------------+
int deinit()
  {
//----

//----
   return(0);
  }
```

```
int Crossed (double line1 , double line2)
   {
      static int last_direction = 0;
      static int current_direction = 0;

      if(line1>line2)current_direction = 1; //up
      if(line1<line2)current_direction = 2; //down



      if(current_direction != last_direction) //changed
      {
            last_direction = current_direction;
            return (last_direction);
      }
      else
      {
            return (0);
      }
   }
//+------------------------------------------------------------------+
//| expert start function                                            |
//+------------------------------------------------------------------+
int start()
  {
//----


   int cnt, ticket, total;
   double shortEma, longEma;


   if(Bars<100)
     {
      Print("bars less than 100");
      return(0);
     }
   if(TakeProfit<10)
     {
      Print("TakeProfit less than 10");
      return(0);  // check TakeProfit
     }


   shortEma = iMA(NULL,0,8,0,MODE_EMA,PRICE_CLOSE,0);
   longEma = iMA(NULL,0,13,0,MODE_EMA,PRICE_CLOSE,0);

   int isCrossed  = Crossed (shortEma,longEma);

   total  = OrdersTotal();
   if(total < 1)
     {
       if(isCrossed == 1)
         {

ticket=OrderSend(Symbol(),OP_BUY,Lots,Ask,3,0,Ask+TakeProfit*Point,
"My EA",12345,0,Green);
            if(ticket>0)
              {
               if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
Print("BUY order opened : ",OrderOpenPrice());
```

```
               }
            else Print("Error opening BUY order : ",GetLastError());
            return(0);
         }
         if(isCrossed == 2)
         {

            ticket=OrderSend(Symbol(),OP_SELL,Lots,Bid,3,0,
Bid-TakeProfit*Point,"My EA",12345,0,Red);
            if(ticket>0)
              {
               if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
Print("SELL order opened : ",OrderOpenPrice());
              }
            else Print("Error opening SELL order : ",GetLastError());
            return(0);
         }
         return(0);
      }
   for(cnt=0;cnt<total;cnt++)
     {
      OrderSelect(cnt, SELECT_BY_POS, MODE_TRADES);
      if(OrderType()<=OP_SELL && OrderSymbol()==Symbol())
        {
         if(OrderType()==OP_BUY)   // long position is opened
           {
            // should it be closed?
           if(isCrossed == 2)
                {
                 OrderClose(OrderTicket(),OrderLots(),Bid,3,Violet);
// close position
                 return(0); // exit
                }
            // check for trailing stop
            if(TrailingStop>0)
              {
               if(Bid-OrderOpenPrice()>Point*TrailingStop)
                 {
                  if(OrderStopLoss()<Bid-Point*TrailingStop)
                    {
                     OrderModify(OrderTicket(),OrderOpenPrice(),Bid-
Point*TrailingStop,OrderTakeProfit(),0,Green);
                     return(0);
                    }
                 }
              }
           }
         else // go to short position
           {
            // should it be closed?
            if(isCrossed == 1)
              {
               OrderClose(OrderTicket(),OrderLots(),Ask,3,Violet);
// close position
               return(0); // exit
              }
            // check for trailing stop
            if(TrailingStop>0)
              {
               if((OrderOpenPrice()-Ask)>(Point*TrailingStop))
                 {
```

```
                  if((OrderStopLoss()>(Ask+Point*TrailingStop)) ||
(OrderStopLoss()==0))
                    {

OrderModify(OrderTicket(),OrderOpenPrice(),Ask+Point*TrailingStop,
OrderTakeProfit(),0,Red);
                     return(0);
                    }
                 }
              }
           }
        }
     }
   return(0);
  }
//+------------------------------------------------------------------+
```

In the previous lesson, we checked the ***OrdersTotal*** is less than **1** in order to open a **Buy** or a **Sell** orders in the case that there were no already opened orders.
We have used this code:

```
if(total < 1)
     {
       if(isCrossed == 1)
         {
          .....
         }
         if(isCrossed == 2)
         {

            .....
         }
           return(0);
     }
```

This was the ***Open New Order routine***. Today we will study ***Modify-Close Opened Orders routine***.

```
for(cnt=0;cnt<total;cnt++)
   {
    ....
   }
```

In the above block of code we used a ***for*** loop to go through all the already opened orders. We start the loop from the ***cnt*** = **0** and the end of the loop is the ***total*** number of already orders. Every loop cycle we increase the number of ***cnt*** by **1** (***cnt++***).So, ***cnt*** will hold in every cycle the poison of the order (0,1,2,3 etc) which we will use with ***OrderSelect*** function to select each order by its position.

Our today's mission is studying what's going inside the heart of the above loop.

```
OrderSelect(cnt, SELECT_BY_POS, MODE_TRADES);
      if(OrderType()<=OP_SELL && OrderSymbol()==Symbol())
        {
         ....
        }
```

The ***OrderSelect*** function used to select an opened order or a pending order by the ticket number or by index.

We used the ***OrderSelect*** here before using the ***OrderType*** and ***OrderSymbol*** functions because if we didn't use ***OrderSelect,*** the ***OrderType*** and ***OrderSymbol*** functions will not work.

Note: You have to use ***OrderSelect*** function before the trading functions which takes no parameters:
OrderMagicNumber, OrderClosePrice, OrderCloseTime, OrderOpenPrice, OrderOpenTime, OrderComment, OrderCommission, OrderExpiration, OrderLots, OrderPrint, OrderProfit, OrderStopLoss, OrderSwap, OrderSymbol, OrderTakeProfit, OrderTicket and OrderType

We used ***SELECT_BY_POS*** selecting type which means we want to select the order by its index (position) not by its ticket number.

Note: The index of the first order is **0** and the index of the second one is **1** index etc.

And we used ***MODE_TRADES*** mode which means we will select from the currently trading orders (opened and pending orders) not from the history.

The ***OrderType*** function returns the type of selected order that will be one of:
OP_BUY, OP_SELL, OP_BUYLIMIT, OP_BUYSTOP, OP_SELLLIMIT or ***OP_SELLSTOP***

We checked the type of the order to find is it ***equal*** or ***lesser*** than ***OP_SELL***.
Which means it maybe one of two cases: ***OP_SELL*** or ***OP_BUY*** (because ***OP_SELL***=1 and ***OP_BUY*** = 0). We did that because we will not work with pending orders.

We want too to work only with the order opened in the chart we loaded our expert advisor on, so we check the ***OrderSymbol*** of the order with the return value of ***Symbol*** function which returns the current chart symbol. If they are equal it means we are working with the currently loaded symbol.

So, all the coming code will work only if the ***OrderType*** is ***OP_SELL*** or ***OP_BUY*** and the ***Symbol*** = ***OrderSymbol***.

```
if(OrderType()==OP_BUY)   // long position is opened
   {
    ....
   }
```

We are working only with two types of orders, the first type is ***OP_BUY***.
The code above means:

Is there a long (Buy) position opened? If yes! Execute this block of code….
Let's see what will do in the case of a long position has been opened

---

```
if(isCrossed == 2)
  {
      OrderClose(OrderTicket(),OrderLots(),Bid,3,Violet);
      // close position
      return(0); // exit
  }
```

We have opened a **Buy** order when the ***shortEma*** crossed the ***longEma*** **upward**.
It's a logical to close this position when the ***shortEma*** and ***longEma*** crosses each others in reversal direction (**downward**).

So, we checked the ***isCrossed*** to find is it = **2** which means the reversal has been occurred and in this case we close the **Buy** order.

We used the ***OrderClose*** function to close the order. ***OrderClose*** function closes a specific opened order by its ticket. (Review appendix 2).

We've got the ticket number of the selected order using the ***OrderTicket*** function and passed it as the first parameter for ***OrderClose*** function.
The second parameter in the ***OrderClose*** is ***Lots*** (the number of lots); we used the ***OrderLots*** function to get the lots value of the selected order.
The third parameter in ***OrderClose*** is the preferred close price and we used the ***Bid*** function to get the bid price of the selected order.
The fourth parameter is the ***slippage*** value and we used **3**.
The fifth parameter is the color of the closing arrow and we used ***Violet*** color.

We didn't forget to terminate the ***start*** function with ***return(0)*** statement.

```
// check for trailing stop
            if(TrailingStop>0)
              {
               if(Bid-OrderOpenPrice()>Point*TrailingStop)
                 {
                  if(OrderStopLoss()<Bid-Point*TrailingStop)
                    {
                     OrderModify(OrderTicket(),OrderOpenPrice(),Bid-
Point*TrailingStop,OrderTakeProfit(),0,Green);
                     return(0);
                    }
                 }
              }
```

Note: We are still inside the block of: ***if(OrderType()==OP_BUY)***.

We are going to apply our trailing stop technique for the opened **Buy** position in this block of code.

Firstly, we have checked the ***TrailingStop*** variable the user supplied to check was it a valid value or not (greater than **0**).

Then we applied our trailing stop technique for the opened **Buy** orders which is:

We **modify** the ***stoploos*** of the order when the **subtraction** of the current **bid** price and the **opened price** of order is **greater** than the ***TrailingStop***
and
the current ***stoploss*** is **lesser** than the **subtraction** of the current **bid** price and the ***TrailingStop***.

We used the ***OrderModify*** function to make the desired modification.
These are parameters we used with ***OrderModify***:

ticket: We've got the current order ticket with ***OrderTicket*** function.

price: We've got the open price of the order with ***OrderOpenPrice*** function.

stoploss: Here's the real work! Because we are in a **Buy** position we set our new ***stoploss*** to the value of the **subtraction** of the current **bid** price and the ***TrailingStop***.
That's our way trailing the ***stoploss*** point every time we make profits.

Note: Stop losses are always set **BELOW** the current bid price on a **buy** and **ABOVE** the current asking price on a **sell**.

takeprofit: No changes, we've got the current profit value of the order with ***OrderTakeProfit*** function.

expiration: We didn't set an ***expiration*** date to our order, so we used **0**.

arrow_color: Still ***Green*** color.

Finally we terminate the start function.

```
else // go to short position
   {
    ....
   }
```

Note: ***else*** here belongs to the code:

```
if(OrderType()==OP_BUY)   // long position is opened
   {
    ....
   }
```

We have studied the case of the type of the order is a **Buy** order.
So, we are working now a **Sell** order type.
Let's see what are we going to do in the case of a short (Sell) position has been already opened?

```
if(isCrossed == 1)
  {
```

```
      OrderClose(OrderTicket(),OrderLots(),Ask,3,Violet);
      // close position
      return(0); // exit
  }
```

We've opened a **Sell** order when the ***shortEma*** crossed the ***longEma*** **downward**.
It's the time to close this position when the ***shortEma*** and ***longEma*** crosses each others in reversal direction (**upward**). Which happens in the case of ***isCrossed*** = **1**.

We used the ***OrderClose*** function to close the order, we used the same parameters we use in the case of closing a **Buy** order except the third parameters the preferred close price, in this case is the **Ask** price.
Then we terminated the ***start*** function.

```
// check for trailing stop
            if(TrailingStop>0)
              {
               if((OrderOpenPrice()-Ask)>Point*TrailingStop)
                 {
                  if((OrderStopLoss()>(Ask+Point*TrailingStop)) ||
(OrderStopLoss()==0))
                    {

OrderModify(OrderTicket(),OrderOpenPrice(),Ask+Point*TrailingStop,
OrderTakeProfit(),0,Red);
                     return(0);
                    }
                 }
              }
```

We are going to apply our trailing stop technique for the opened **Sell** position in this block of code.

Firstly, we've checked the ***TrailingStop*** variable the user supplied to check was it a valid value or not (greater than **0**).
Then we applied our trailing stop technique for the opened **Sell** orders which is:

We **modify** the ***stoploss*** of the order when the **subtraction** of the order's **opened price** and the current **ask** price is **greater** than the ***TrailingStop*** **and** the current ***stoploss*** is **greater** than the **addition** of the current **ask** price and the ***TrailingStop***.

We used the ***OrderModify*** function to make the desired modification. And used the same parameters we use in the case of modifying already opened Buy order except the third parameter which indicates our ***stoploss*** value:
We set our new ***stoploss*** to the value of the **addition** of the current **ask** price and the ***TrailingStop***.

And the fifth parameter which indicates the color of the arrow in this case is ***Red***.

Then we terminated the ***start*** function using ***return(0)***;

```
return(0);
```

This line terminates the ***start*** function in all other case; there are no conditions to open new positions and there are no needs to close or modify the already opened orders. Just don't forget it.

I hope you enjoyed the lesson.
I welcome very much your questions and suggestions.

Coders' Guru
28-12-2005