Nyní, namísto monstrózního zápisu hodnoty **iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_MAIN,0)**, můžete ve zdrojovém kódu použít **MacdCurrent**.

4. **Kontrola obchodního terminálu – je prázdný? Pokud ano, pak:**
   V našem Expert Advisoru používáme pouze ty pozice, které jsou otevřeny příkazy trhu (market orders) a nepoužíváme entry příkazy „pending orders“. Pro jistotu ale nahlédněme, jak probíhá kontrola obchodního terminálu u dříve zadaných příkazů:

```
total=OrdersTotal();

if(total<1)
  {
```

   - **kontroly: dostupnost finančních prostředků na kontě atd....**
     Před analýzou situace na trhu se doporučuje zkontrolovat stav účtu, abyste se ujistili, zda máte dostatek volných prostředků pro otevření pozice.

```
if(AccountFreeMargin()<(1000*Lots))
  {
   Print("We have no money. Free Margin = ", AccountFreeMargin());
   return(0);
  }
```

   - **je možné zaujmout pozici long (NÁKUP)?**
     Podmínky pro vstup do pozice long: MACD je pod úrovní nula, stoupá a protíná klesající signální linii. Takto je to popsáno v MQL 4 (mějte na paměti, že pracujeme s hodnotami indikátoru, které byly dříve uloženy v proměnných):

```
// zkontrolujte možnosti pro pozici long (NÁKUP)
if(MacdCurrent<0 && MacdCurrent>SignalCurrent &&
   MacdPrevious<SignalPrevious &&
   MathAbs(MacdCurrent)>(MACDOpenLevel*Point) &&
   MaCurrent>MaPrevious)
  {
   ticket=OrderSend(Symbol(),OP_BUY,Lots,Ask,3,0,Ask+TakeProfit*Point,
                    "macd sample",16384,0,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);
  }
```

     Dodatečná kontrola velikosti 'kopečků' byla již popsaná výše. Proměnná MACDOpenLevel je uživatelem definovaná hodnota, která může být změněna bez zásahu do textu programu, čímž nabízí vyšší flexibilitu. Na začátku programu vkládáme popis této proměnné (stejně tak proměnné použité níže).

   - **Je možné zaujmout pozici short (PRODAT)?**
     Podmínky pro vstup do pozice short: MACD je nad úrovní nula, směřuje nahoru a protíná signální linii směřující nahoru. Zápis vypadá takto:

```
            // kontrola možnosti zaujetí pozice short (PRODEJ)
            if(MacdCurrent>0 && MacdCurrent<SignalCurrent &&
      MacdPrevious>SignalPrevious &&
               MacdCurrent>(MACDOpenLevel*Point) && MaCurrent<MaPrevious)
              {
               ticket=OrderSend(Symbol(),OP_SELL,Lots,Bid,3,0,Bid-TakeProfit*Point,
                                "macd sample",16384,0,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);
}
```

5. **Kontrola otevřené pozice v cyklu**

```
// je důležité správné vložení objednávky,
// ještě důležitější je správné opuštění...
for(cnt=0;cnt<total;cnt++)
  {
   OrderSelect(cnt, SELECT_BY_POS, MODE_TRADES);
   if(OrderType()<=OP_SELL &&   // kontrola otevřené pozice
      OrderSymbol()==Symbol())  // kontrola symbolu
     {
```

"cnt" – " je cyklus proměnné, který musí být definován na začátku prograu takto:

```
int cnt = 0;
```

- pokud se jedná o pozici long

```
      if(OrderType()==OP_BUY)   // pozice long je otevřená
        {
```

  - **měla by být uzavřena?**
    Podmínky pro opuštění pozice long: MACD je protnuto signální linií Signal Line, MACD je nad úrovní nula, směřuje dolů a je protnuto signální linií směřující nahoru.

```
      if(MacdCurrent>0 && MacdCurrent<SignalCurrent &&
      MacdPrevious>SignalPrevious &&
         MacdCurrent>(MACDCloseLevel*Point))
        {
         OrderClose(OrderTicket(),OrderLots(),Bid,3,Violet); // uzavření
      pozice
         return(0); // opuštění
        }
```

  - **měl by být trailing stop posunut?**
    Trailing stop se nastavuje pouze v případě, že pozice již přesáhla zisk nastavený bodem úrovně funkce trailing stop a v případě, že nová úroveň je lepší než ta předchozí.

```
      // kontrola funkce 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);
              }
           }
        }
```

Uzavřeme závorku operátoru.

```
   }
```

- pokud se jedná o pozici short

```
else //přechod do pozice short
  {
```

- **Měla by být uzavřena?**
  Podmínky pro opuštění pozice short: MACD je protnuto signální linií Signal Line, MACD je pod úrovní nula, směřuje nahoru a je protnuto signální linií směřující dolů.

```
      if(MacdCurrent<0 && MacdCurrent>SignalCurrent &&
      MacdPrevious<SignalPrevious &&
      MathAbs(MacdCurrent)>(MACDCloseLevel*Point))
        {
         OrderClose(OrderTicket(),OrderLots(),Ask,3,Violet); // uzavření
      pozice
         return(0); // opuštění
        }
```

měl by být trailing stop posunut?
Trailing stop se nastavuje pouze v případě, že pozice již přesáhla zisk nastavený bodem úrovně funkce trailing stop a v případě, že nová úroveň je lepší než ta předchozí.

```
      // kontrola funkce 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);
              }
           }
        }
```

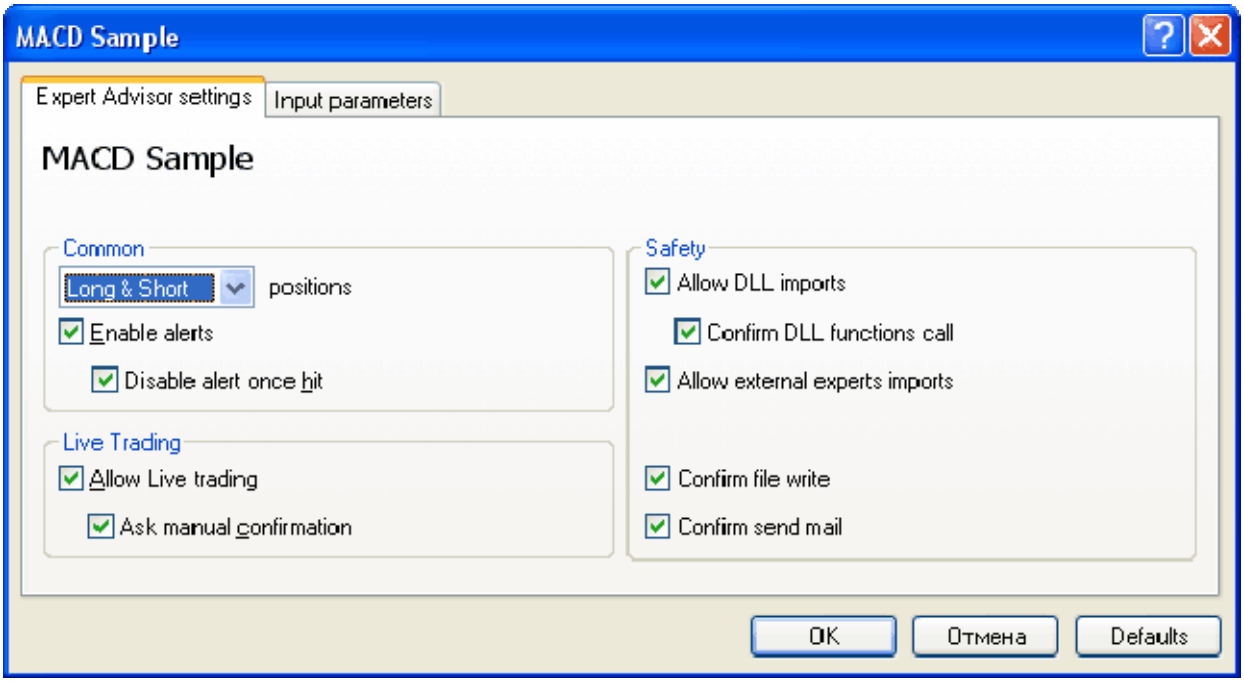
Uzavření všech svorek, které zůstaly otevřené.

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

Postupováním podle tohoto návodu krok za krokem zapíšete náš Expert Advisor...

Krok 3 – Sestavení výsledného kódu programu

Nyní otevřeme nastavení Expert Advisoru (použitím ikonky nebo řádku "Properties..." menu. Zobrazí se nám okno s nabídkou, ve kterém určíme externí nastavení pracovních parametrů:



Nyní sestavíme celý kód z předchozích sekcí:

```
//+------------------------------------------------------------------+
//|                                                  MACD Sample.mq4 |
//|                      Copyright й 2005, MetaQuotes Software Corp. |
//|                                       http://www.metaquotes.net/ |
//+------------------------------------------------------------------+

extern double TakeProfit = 50;
extern double Lots = 0.1;
extern double TrailingStop = 30;
extern double MACDOpenLevel=3;
extern double MACDCloseLevel=2;
extern double MATrendPeriod=26;

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int start()
  {
   double MacdCurrent, MacdPrevious, SignalCurrent;
   double SignalPrevious, MaCurrent, MaPrevious;
   int cnt, ticket, total;
// kontrola inicializačních dat
// je důležité, aby expert advisor pracoval s normálním
// grafem a uživatel nepochybil při nastavování externích
// proměnných (Lots, StopLoss, TakeProfit,
// TrailingStop) v našem případě kontrolujeme funkci TakeProfit
// v tabulce o méně než 100 svíčkách
   if(Bars<100)
     {
      Print("bars less than 100");
      return(0);
     }
   if(TakeProfit<10)
```

```
     {
      Print("TakeProfit less than 10");
      return(0);  // kontrola funkce TakeProfit
     }
// to simplify the coding and speed up access
// data are put into internal variables
   MacdCurrent=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_MAIN,0);
   MacdPrevious=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_MAIN,1);
   SignalCurrent=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_SIGNAL,0);
   SignalPrevious=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_SIGNAL,1);
   MaCurrent=iMA(NULL,0,MATrendPeriod,0,MODE_EMA,PRICE_CLOSE,0);
   MaPrevious=iMA(NULL,0,MATrendPeriod,0,MODE_EMA,PRICE_CLOSE,1);

   total=OrdersTotal();
   if(total<1)
     {
      // nebyly identifikovány žádné otevřené příkazy
      if(AccountFreeMargin()<(1000*Lots))
        {
         Print("We have no money. Free Margin = ", AccountFreeMargin());
         return(0);
        }
      // kontrola možnosti pozice long (NÁKUP)
      if(MacdCurrent<0 && MacdCurrent>SignalCurrent && MacdPrevious<SignalPrevious &&
         MathAbs(MacdCurrent)>(MACDOpenLevel*Point) && MaCurrent>MaPrevious)
        {
         ticket=OrderSend(Symbol(),OP_BUY,Lots,Ask,3,0,Ask+TakeProfit*Point,
                          "macd sample",16384,0,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);
        }
      // kontrola možnosti pozice short (PRODEJ)
      if(MacdCurrent>0 && MacdCurrent<SignalCurrent && MacdPrevious>SignalPrevious &&
         MacdCurrent>(MACDOpenLevel*Point) && MaCurrent<MaPrevious)
        {
         ticket=OrderSend(Symbol(),OP_SELL,Lots,Bid,3,0,Bid-TakeProfit*Point,
                          "macd sample",16384,0,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);
     }
 // je důležité správné vložení objednávky,
// ještě důležitější je správné opuštění...
   for(cnt=0;cnt<total;cnt++)
     {
      OrderSelect(cnt, SELECT_BY_POS, MODE_TRADES);
      if(OrderType()<=OP_SELL &&   // kontrola otevřené pozice
         OrderSymbol()==Symbol())  // kontrola symbolu
        {
         if(OrderType()==OP_BUY)   // pozice long je otevřena
           {
            // měla by být uzavřena?
            if(MacdCurrent>0 && MacdCurrent<SignalCurrent && MacdPrevious>SignalPrevious
&&
               MacdCurrent>(MACDCloseLevel*Point))
              {
               OrderClose(OrderTicket(),OrderLots(),Bid,3,Violet); // uzavření pozice
               return(0); // opuštění
```

```
                  }
               // kontrola funkce 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 // přechod do pozice short
              {
               // měla by být uzavřena?
               if(MacdCurrent<0 && MacdCurrent>SignalCurrent &&
                  MacdPrevious<SignalPrevious &&
MathAbs(MacdCurrent)>(MACDCloseLevel*Point))
                 {
                  OrderClose(OrderTicket(),OrderLots(),Ask,3,Violet); // uzavření pozice
                  return(0); // opuštění
                 }
               // kontrola funkce 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);
     }
// konec.
```

Pro konečnou konfiguraci našeho expert advisoru pouze specifikujte hodnoty externích proměnných "Lots = 1", "Stop Loss (S/L) = 0" (nepoužíván), "Take Profit (T/P) = 120" (odpovídá jednohodinovým intervalům), "Trailing Stop (T/S) = 30". Můžete nastavit své vlastní hodnoty. Použijte funkci "Compile" a pokud se neobjeví žádné chybové hlášení (můžete si zkopírovat výše uvedený text do MetaEditoru), použijte "Save" k uložení Expert Advisoru.

Features of Experts Advisors creation

Vytvoření Expert advisorů v obchodovacím systému MetaTrader má mnoho aspektů.

- Před otevřením pozice byste měli zkontrolovat, zda máte k dispozici na účtu dostatek financí. Pokud tomu tak není, operace otevírání účtu neproběhne úspěšně. Hodnota "FreeMargin" nesmí být menší než 1000 pouze během testu, protože v testu je cena jedné položky právě 1000.

```
if(AccountFreeMargin() < 1000) return(0);     // nedostatek peněz
```

- Můžete vstoupit do historie dat pomocí předdefinovaných seskupení Time, Open, Low, High, Close, Volume. Vzhledem k historickému původu se index těchto seskupení načítá vzestupně směrem od

konce na začátek. To znamená, že nejnovější data mají index 0. Index 1 označuje data o jednu periodu starší, index 2 data starší o 2 periody atd.

```
// Pokud je hodnota „Close" v předchozí svíčce nižší než
// hodnta Close ve svíčce o dvě svíce zpět
if(Close[1] < Close[2]) return(0);
```

- Rovněž je možné vstoupit do historie dat použitím jiných časových intervalů nebo také prostřednictvím jiných měn. K získání takovýchto dat je nutné definovat nejprve jednorozměrné pole a provést operaci kopírování pomocí funkce "ArrayCopySeries". Mějte na paměti, že během vyvolávání funkce je možné zadat menší počet parametrů a nespecifikovat výchozí (default) parametry.

```
double eur_close_m1[];
int number_copied = ArrayCopySeries(eur_close_m1, MODE_CLOSE, "EURUSD", PERIOD_M1);
```

- V procesu zapisování „expert advisoru", stejně tak jako u jiného softwaru, je občas potřebné zadat dodatečné ladící informace. Jazyk MQL 4 nabízí několik metod pro získání takovýchto informací.

    - Funkce "Alert" vyobrazuje okno dialogu s některými daty definovanými uživatelem.

      ```
      Alert("FreeMargin grows to ", AccountFreeMargin(), "!");
      ```

    - Funkce "Comment" function vyobrazí data definovaná v levém horním rohu tabulky. Znaková sekvence "\n" se používá ke spuštění nového řádku.

      ```
      Comment("FreeMargin is ", AccountFreeMargin(), ".");
      ```

    - Funkce "Print" ukládád data definovaná uživatelem do systémového registru.

      ```
      Print("FreeMargin is ", AccountFreeMargin(), ".");
      ```

- K získávání informaci o programových chybách je velmi užitečná funkce "GetLastError". Například, když se jedná o operaci s příkazem neustále vracejícím číslo tiketu. Pokud se číslo tiketu rovná nule (během procesu vykonávání operace se objevila nějaká chyba), je nutné vyvolat funkci "GetLastError", abychom získali doplňkové informace o chybě:

```
int iTickNum = 0;
int iLastError = 0;
...
iTickNum = OrderSet (OP_BUY, g_Lots, Ask, 3, 0, Ask + g_TakeProfit * g_Points,
Red);
if (iTickNum <= 0)
   {
     iLastError = GetLastError();
     if (iLastError != ERROR_SUCCESS) Alert("Some Message");
   }
```

Měli byste si zapamatovat, že vyvoláním funkce "GetLastError" se vyobrazí kód poslední chyby a resetuje její hodnotu. Proto bude opětovaně vyvolaná funkce v řadě vždy vracet hodnotu nula.

- Jak definovat začátek nové svíce? (Může být potřebné zjistit, zda byla předchozí svíčka dokončena). K tomu slouží několik metod.

První metoda je založena na kontrole počtu svíček:

```
static int prevbars = 0;
...
if(prevbars == Bars) return(0);
prevbars = Bars;
...
```

Tato metoda může selhat při nahrávání historie. To znamená, že počet svíček se změní, jelikož ta předchozí nebyla ještě dokončena. V tom případě můžete provést komplikovanější kontrolu zavedením kontroly rozdílu mezi hodnotami, které se rovnají jedné.
Další metoda je založena na faktu, že hodnota "Volume" je generována v závislosti na počtu ticků obsažených v každé svíčce a prvním políčku tak znamená, že hodnota "Volume" nové svíčky se rovná 1:

```
if( Volume > 1) return(0);
...
```

Tato metoda může selhat v případě příliš velkého množství příchozích ticků. Jde o to, že příchozí ticky jsou zpracovávány v odděleném procesu. Pokud je proces zaneprázdněn během akceptování dalšího ticku, nově příchozí tick není zpracován, aby nedošlo k přetížení procesoru! V tom případě můžete provést kontrolu komplikovanějším způsobem uložením předchozí hodnoty "Volume".

Třetí metoda je založena na času otevření svíčky:

```
static datetime prevtime=0;
...
if(prevtime == Time[0]) return(0);
prevtime = Time[0];
...
```

Jedná se o nejspolehlivější metodu. Funguje ve všech případech.

- Příklad práce se souborem typu "CSV":

```
  int h1;
  h1 = FileOpen("my_data.csv", MODE_CSV | MODE_WRITE, ";");
  if(h1<0)
    {
     Print("Unable to open file my_data.csv");
     return(false);
    }
  FileWrite(h1, High[1], Low[1], Close[1], Volume[1]);
  FileClose(h1);
```

Některá vysvětlení ke kódu. Soubor formátu "CSV" je otevřen jako první. V případě chyby otevíraného souboru je program opuštěn. V případě úspěšného otevření souboru se jeho obsah vyčistí, data jsou do souboru uložena a soubor uzavřen. Pokud chcete zachovat obsah otevíraného souboru, musíte jej otevřít v režimu MODE_READ:

```
int h1;
h1 = FileOpen("my_data.csv", MODE_CSV | MODE_WRITE | MODE_READ, ";");
if(h1<0)
  {
   Print("Unable to open file my_data.csv");
   return(false);
  }
FileSeek(h1, 0, SEEK_END);
FileWrite(h1, High[1], Low[1], Close[1], Volume[1]);
FileClose(h1);
```

V tomto příkladě jsou data přiřazena na konec souboru. K tomu jsme ihned po otevření použili funkci "FileSeek".

Features of Custom Indicators creation

Tvorba funkce „Custom Indicators“ v obchodním sytému „MetaTrader“ má několik rysů.

- Aby byl program plnohodnotným programem „Custom Indicator“, musí platit jedna ze dvou definic:

```
#property  indicator_chart_window      // indikátor je vykreslen v hlavním okně
tabulky
```

nebo

```
#property  indicator_separate_window   // indikátor je vykreslen v odděleném okně
```

- K nastavení stupnice odděleného okna identifikátoru jsou použity tyto definice:

```
#property  indicator_minimum Min_Value
#property  indicator_maximum Max_Value
```

přičemž "Min_Value" a "Max_Value" jsou odpovídající hodnoty. Například hodnoty musí být 0 a 100 na druhé straně pro „custom indicator“ RSI.

- Počet polí pro vykreslení indikátoru musí být definován takto:

```
  #property  indicator_buffers N
```

přičemž N může zaujmout hodnoty 1 až 8.

- Barvy řádků indiátoru jsou nastaveny prostřednictvím těchto definic:

```
#property  indicator_color1  Silver
#property  indicator_color2  Red
...
#property  indicator_colorN  <SomeColor>
```

přičemž N je počet polí indikátoru definovaný prostřednictvím "#property indicator_buffers".

- Jsou zde obsaženy funkce umožňující řízení procesu výpočtu a vizualizace indikátoru. Custom Indicator od programátora Ishimoku Kinko Hyo je zde použit pro ilustraci:

```
//+------------------------------------------------------------------+
//|                                                     Ichimoku.mq4 |
//|                   Copyright © 2004, MetaQuotes Software Corp. |
//|                                       http://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "Copyright © 2004, MetaQuotes Software Corp."
#property link      "http://www.metaquotes.net/"

#property indicator_chart_window
#property indicator_buffers 7
#property indicator_color1 Red
#property indicator_color2 Blue
#property indicator_color3 SandyBrown
#property indicator_color4 Thistle
#property indicator_color5 Lime
#property indicator_color6 SandyBrown
#property indicator_color7 Thistle
//---- vstupní parametry
extern int Tenkan=9;
extern int Kijun=26;
extern int Senkou=52;
```

```
//---- paměti indikátoru
double Tenkan_Buffer[];
double Kijun_Buffer[];
double SpanA_Buffer[];
double SpanB_Buffer[];
double Chinkou_Buffer[];
double SpanA2_Buffer[];
double SpanB2_Buffer[];
//---- span_a drawing begin
int a_begin;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int init()
  {
//----
   SetIndexStyle(0,DRAW_LINE);
   SetIndexBuffer(0,Tenkan_Buffer);
   SetIndexDrawBegin(0,Tenkan-1);
   SetIndexLabel(0,"Tenkan Sen");
//----
   SetIndexStyle(1,DRAW_LINE);
   SetIndexBuffer(1,Kijun_Buffer);
   SetIndexDrawBegin(1,Kijun-1);
   SetIndexLabel(1,"Kijun Sen");
//----
   a_begin=Kijun; if(a_begin<Tenkan) a_begin=Tenkan;
   SetIndexStyle(2,DRAW_HISTOGRAM,STYLE_DOT);
   SetIndexBuffer(2,SpanA_Buffer);
   SetIndexDrawBegin(2,Kijun+a_begin-1);
   SetIndexShift(2,Kijun);
   SetIndexLabel(2,NULL);
   SetIndexStyle(5,DRAW_LINE,STYLE_DOT);
   SetIndexBuffer(5,SpanA2_Buffer);
   SetIndexDrawBegin(5,Kijun+a_begin-1);
   SetIndexShift(5,Kijun);
   SetIndexLabel(5,"Senkou Span A");
//----
   SetIndexStyle(3,DRAW_HISTOGRAM,STYLE_DOT);
   SetIndexBuffer(3,SpanB_Buffer);
   SetIndexDrawBegin(3,Kijun+Senkou-1);
   SetIndexShift(3,Kijun);
   SetIndexLabel(3,NULL);
   SetIndexStyle(6,DRAW_LINE,STYLE_DOT);
   SetIndexBuffer(6,SpanB2_Buffer);
   SetIndexDrawBegin(6,Kijun+Senkou-1);
   SetIndexShift(6,Kijun);
   SetIndexLabel(6,"Senkou Span B");
//----
   SetIndexStyle(4,DRAW_LINE);
   SetIndexBuffer(4,Chinkou_Buffer);
   SetIndexShift(4,-Kijun);
   SetIndexLabel(4,"Chinkou Span");
//----
   return(0);
  }
//+------------------------------------------------------------------+
//| Ichimoku Kinko Hyo                                               |
//+------------------------------------------------------------------+
int start()
  {
   int    i,k;
   int    counted_bars=IndicatorCounted();
   double high,low,price;
//----
   if(Bars<=Tenkan || Bars<=Kijun || Bars<=Senkou) return(0);
//---- initial zero
   if(counted_bars<1)
     {
```

```
      for(i=1;i<=Tenkan;i++)    Tenkan_Buffer[Bars-i]=0;
      for(i=1;i<=Kijun;i++)     Kijun_Buffer[Bars-i]=0;
      for(i=1;i<=a_begin;i++) { SpanA_Buffer[Bars-i]=0; SpanA2_Buffer[Bars-i]=0; }
      for(i=1;i<=Senkou;i++)  { SpanB_Buffer[Bars-i]=0; SpanB2_Buffer[Bars-i]=0; }
     }
//---- Tenkan Sen
   i=Bars-Tenkan;
   if(counted_bars>Tenkan) i=Bars-counted_bars-1;
   while(i>=0)
     {
      high=High[i]; low=Low[i]; k=i-1+Tenkan;
      while(k>=i)
        {
         price=High[k];
         if(high<price) high=price;
         price=Low[k];
         if(low>price)  low=price;
         k--;
        }
      Tenkan_Buffer[i]=(high+low)/2;
      i--;
     }
//---- Kijun Sen
   i=Bars-Kijun;
   if(counted_bars>Kijun) i=Bars-counted_bars-1;
   while(i>=0)
     {
      high=High[i]; low=Low[i]; k=i-1+Kijun;
      while(k>=i)
        {
         price=High[k];
         if(highprice)  low=price;
         k--;
        }
      Kijun_Buffer[i]=(high+low)/2;
      i--;
     }
//---- Senkou Span A
   i=Bars-a_begin+1;
   if(counted_bars>a_begin-1) i=Bars-counted_bars-1;
   while(i>=0)
     {
      price=(Kijun_Buffer[i]+Tenkan_Buffer[i])/2;
      SpanA_Buffer[i]=price;
      SpanA2_Buffer[i]=price;
      i--;
     }
//---- Senkou Span B
   i=Bars-Senkou;
   if(counted_bars>Senkou) i=Bars-counted_bars-1;
   while(i>=0)
     {
      high=High[i]; low=Low[i]; k=i-1+Senkou;
      while(k>=i)
        {
         price=High[k];
         if(high<price) high=price;
         price=Low[k];
         if(low>price)  low=price;
         k--;
        }
      price=(high+low)/2;
      SpanB_Buffer[i]=price;
      SpanB2_Buffer[i]=price;
      i--;
     }
//---- Chinkou Span
   i=Bars-1;
   if(counted_bars>1) i=Bars-counted_bars-1;
```

```
   while(i>=0) { Chinkou_Buffer[i]=Close[i]; i--; }
//----
   return(0);
  }
//+------------------------------------------------------------------+
```

- Funkce "SetIndexStyle" řídí vykreslení parametrů seskupení indikátoru. Vykreslovací režim DRAW_LINE předpokládá, že budou vykresleny linie určené mezi hodnotami odpovídajícího pole indikátoru. Vykreslovací režim DRAW_HISTOGRAM má po aplikaci do hlavního okna indikátoru rovněž zvláštní vlastnosti. Histogram je vykreslen mezi odpovídajícími hodnotami dvou indexů polí: jednou sudou (SpanA_Buffer) a jednou lichou (SpanB_Buffer). Přitom je použita barva indexu pole, jehož hodnota je vyšší.
- Funkce "SetIndexDrawBegin" specifikuje, který element důležitých dat pole indikátoru spustit.
- Funkce "SetIndexBuffer" umožňuje vyjádření jakéhokoliv jednorozměrného pole typu "double" jako indexu pole. Přitom bude systém řídit indexová pole. To je důvod, proč tato pole nemusí být specifikována.
- `//---- paměti indikátorů`
- `double Tenkan_Buffer[];`
- `double Kijun_Buffer[];`
- `double SpanA_Buffer[];`
- `double SpanB_Buffer[];`
- `double Chinkou_Buffer[];`
- `double SpanA2_Buffer[];`
- `double SpanB2_Buffer[];`

Funkce ArrayResize nemůže být aplikována do indikátoru pole, kromě toho je to zbytečné. Zbytečné je rovněž aplikování funkce ArrayInitialize do pole indikátoru, zejména funkce 'init', pokud ještě nedošlo k rozdělení indikátorových polí. Indikátory pole jsou inicializovány automaticky během alokace a relokace paměti. EMPTY_VALUE, nebo hodnota specifikována hodnotou funkce SetIndexEmptyValue jsou používány jako inicializační hodnoty. Hodnoty "Empty" nejsou vyobrazeny.

- Funkce "SetIndexLabel" nastaví jméno k vyobrazení v liště „tool tips“ a „data window“ spolu s odpovídající hodnotou (hodnota"ValueN" je nastavena jako výchozí (default), kde N je číslo indexového pole . Pokud je hodnota NULL vložena na místo jména, odpovídající hodnota nebude vyobrazena ani v liště „tool tips“ ani v „data window“. V daných případech je použito zastínění použitím histogramu a omezení řádkem. Přitom hodnoty odpovídajícího řádku a histogramu seskupení jsou shodné a je možné vyobrazit pouze jednu z nich.
- Funkce "IndicatorCounted" umožňuje organizovat výpočet indikátoru. Tato funkce vrací počet svíček v momentě před spuštěním indikátoru, tj. počet již propočítaných svíček (pokud nedošlo k chybám nebo předčasnému ukončení předchozí aktivace), které žádné další přepočítávání nepotřebují. Při reinicializaci funkce „custom indicator“ nebo důležité aktualizaci historie dat je tento počet automaticky vynulován.
- Nyní se podíváme ještě na jeden příklad. „Custom indicator“ se jménem „Accelerator/Decelerator Oscillator“:

```
//+------------------------------------------------------------------+
//|                                                 Accelerator.mq4 |
//|                      Copyright © 2005, MetaQuotes Software Corp. |
//|                                       http://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property  copyright "Copyright © 2005, MetaQuotes Software Corp."
#property  link      "http://www.metaquotes.net/"
//---- nastavení indikátoru
#property  indicator_separate_window
#property  indicator_buffers 3
#property  indicator_color1  Black
#property  indicator_color2  Green
#property  indicator_color3  Red
//---- paměti indikátoru
double     ExtBuffer0[];
double     ExtBuffer1[];
```

```
double     ExtBuffer2[];
double     ExtBuffer3[];
double     ExtBuffer4[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int init()
  {
//---- 2 dodatečné paměti pro výpočet.
   IndicatorBuffers(5);
//---- vykreslení nastavení
   SetIndexStyle(0,DRAW_NONE);
   SetIndexStyle(1,DRAW_HISTOGRAM);
   SetIndexStyle(2,DRAW_HISTOGRAM);
   IndicatorDigits(Digits+2);
   SetIndexDrawBegin(0,38);
   SetIndexDrawBegin(1,38);
   SetIndexDrawBegin(2,38);
//---- 4 mapování 4 pamětí indikátoru
   SetIndexBuffer(0,ExtBuffer0);
   SetIndexBuffer(1,ExtBuffer1);
   SetIndexBuffer(2,ExtBuffer2);
   SetIndexBuffer(3,ExtBuffer3);
   SetIndexBuffer(4,ExtBuffer4);
//---- pojmenování pro DataWindow a označení indikátoru subwindow
   IndicatorShortName("AC");
   SetIndexLabel(1,NULL);
   SetIndexLabel(2,NULL);
//---- inicializace hotová
   return(0);
  }
//+------------------------------------------------------------------+
//| Accelerator/Decelerator Oscillator                               |
//+------------------------------------------------------------------+
int start()
  {
   int    limit;
   int    counted_bars=IndicatorCounted();
   double prev,current;
//---- poslední započtená svíčka bude přepočítána
   if(counted_bars>0) counted_bars--;
   limit=Bars-counted_bars;
//---- macd započten v první dodatečné paměti
   for(int i=0; i<limit; i++)
      ExtBuffer3[i]=iMA(NULL,0,5,0,MODE_SMA,PRICE_MEDIAN,i)-
                    iMA(NULL,0,34,0,MODE_SMA,PRICE_MEDIAN,i);
//---- signální linie započtena ve druhém řádku dodatečné paměti
   for(i=0; i=0; i--)
     {
      current=ExtBuffer3[i]-ExtBuffer4[i];
      prev=ExtBuffer3[i+1]-ExtBuffer4[i+1];
      if(current>prev) up=true;
      if(current<prev) up=false;
      if(!up)
        {
         ExtBuffer2[i]=current;
         ExtBuffer1[i]=0.0;
        }
      else
        {
         ExtBuffer1[i]=current;
         ExtBuffer2[i]=0.0;
        }
       ExtBuffer0[i]=current;
     }
//---- hotovo
   return(0);
  }
//+------------------------------------------------------------------+
```

- Funkce "IndicatorBuffers" specifikuje počet vyrovnávacích pamětí určených k výpočtu indikátoru. Obesně se tato funkce aktivuje při použití více indexových polí než je nutné pro vykreslení indikátoru. V tom případě systém řídí dodatečná pole.
- Funkce "SetIndexDigits" řídí přesnost informačního výstupu. V případě, že je rozdíl mezi pohyblivými průměry a rozdíl mezi výsledky a signálními liniemi propočítáván, standardní přesnost o čtyřech znacích za desetinnou tečkou bude zjevně nedostatečná.
- Funkce "SetIndexDrawBegin" specifikuje element, u kterého důležitá data indikátoru pole začínají. V našem případě je signální linie vypočtena jako jednoduchý pohyblivý průměr jiného jednoduchého pohyblivého průměru. Z tohoto důvodu je prvních 38 hodnot indikátoru považováno za prázdné hodnoty a nejsou vykresleny.
- Funkce "IndicatorShortName" nastavuje tzv. krátká jména indikátoru pro vyobrazení v levé horní části okna indikátoru a v okně "DataWindow". Pokud nebylo krátké jméno nastaveno, jméno custom indikátoru bude použito jako v minulosti. V daném příkladě není třeba používat SetIndexLabel, protože jen jedna hodnota je výstupní. Proto stačí jedno pojmenování indikátoru pro výstup jednoduché hodnoty.
- Funkce "SetIndexStyle" řídí vykreslování parametrů indikátoru seskupení. Vykreslovací režim DRAW_NONE znamená, že řádek linii není třeba vykreslovat. Jedná se o to, že histogram vyobrazeného indikátoru musí být vykreslen ve dvou odlišných barvách. Data z paměti ExtBuffer0 jsou alokována ve dvou dalších polích, ExtBuffer1 a ExtBuffer2. Abychom v nástroji tool tips nebo data window na výstupu nezdvojovali data, použije se funkce SetIndexLabel s parametrem NULL. Vykreslovací režim DRAW_HISTOGRAM je aplikován do indikátoru v odděleném okně a umožňuje vykreslení histogramu v rozpětí mezi hodnotou nula a hodnotou odpovídajícího pole. (porovnejte s vykreslováním histogramu hlavního okna popsaného výše).
- Vstupní parametry používané pro kalkulaci pomocí funkce „custom indicators“ a jiných funkcí musí být definovány jako externí "extern" a mohou být odlišeného typu.
- Pokud nebyly vstupní parametry nastaveny, funkce „custom indicator“ bude aktivována v nejjednodušším formátu.

  ```
  double current_AC = iCustom( NULL, 0, "Accelerator", 0, 0 );
  ```

  Přenesení prvních dvou hodnot "NULL" nebo "0" znamená, že bude použita aktuální graf. Jméno odpovídajícího souboru (bez přípony mq4) je použito jako jméno funkce custom indicator. Pokud je hodnota předposledního parametru 0, znamená to, že nás zajímají data z úplně prvního indikátoru pole. Hodnota 0 u posledního parametru znamená, že nás zajímá hodnota posledního elementu (tj. nejnovější, aktuální hodnota) požadovaného pole indikátoru.

- Parametry jsou ve funkci custom indicator propočítávány ve stejném pořadí, v jakém byly zadávány. Na příklad custom indicator pojmenovaný "Ichimoku" s parametry (9,26,52) bude aktivován takto:

  ```
  iCustom( NULL, 0, "Ichimoku", 9, 26, 52, 0, shift );
  ```

  Stručně řečeno, parametry funkce custom indicator nemusí být nutně přenášeny do vlastní funkce. Pokud není definována žádná externí proměnná v programu, je přenos parametrů zbytečný. Nebo mohou být v případě potřeby použity vstupní hodnoty použité pro popis parametrů. Jako příklad můžeme uvést funkci custom indicator bez zadaných parametrů, která bude vyvolána takto:

  ```
  iCustom(NULL, 0, "Ichimoku", 0, shift);
  ```

  To znamená, že hodnoty budou použity k inicializaci proměnných "Tenkan", "Kijun", "Senkou", tj. 9, 26, a 52. Pokud je však vyvolán jeden custom indicator s rozdílnou sestavou parametrů v jednom Expert Advisoru, důrazně se nedoporučuje použití default nastavení.

Je třeba si uvědomit, že nadbytek funkcí custom indicator, stejně tak jejich chybný zápis, může vést ke značnému zpomalení práce na klientském terminálu!

Tester strategie: Režimy modelování v průběhu testování

Úvod

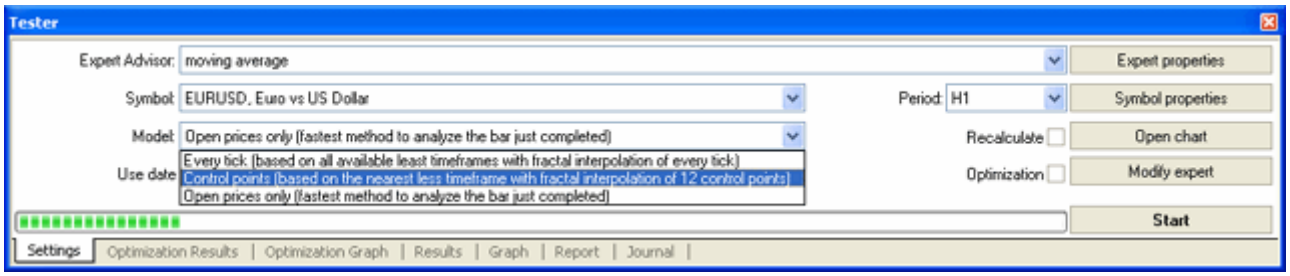
Mnoho programů technických analýz umožňuje testování obchodních strategií v historických datech. Ve většině případů testování vychází z již provedených akcí bez jakéhokoliv pokusu modelování trendů v rámci cenové svíce. Je to sice rychlý způsob, nikoliv však dostatečně přesný.

Je důležité zvolit odpovídající způsob vývoje modelování cenových svíček za účelem kvalitního provedení testování obchodní strategie. Ve skutečnosti nemůže nikdy nastat ideální situace, kdy by byla historie plně ošetřena tak, aby mohlo být provedeno maximálně přesné testování. Pro běžného obchodníka je velmi obtížné nalézt odpovídající historii k analýze. Ta by v ideálním případě musela zasahovat do časových úseků starých několik let.

K vyřešení tohoto problému mohou být jako referenční body použita data jiných (preciznějších) historií a modelování změn cen nastalých mezi nimi.

Způsoby modelování cenových svíček

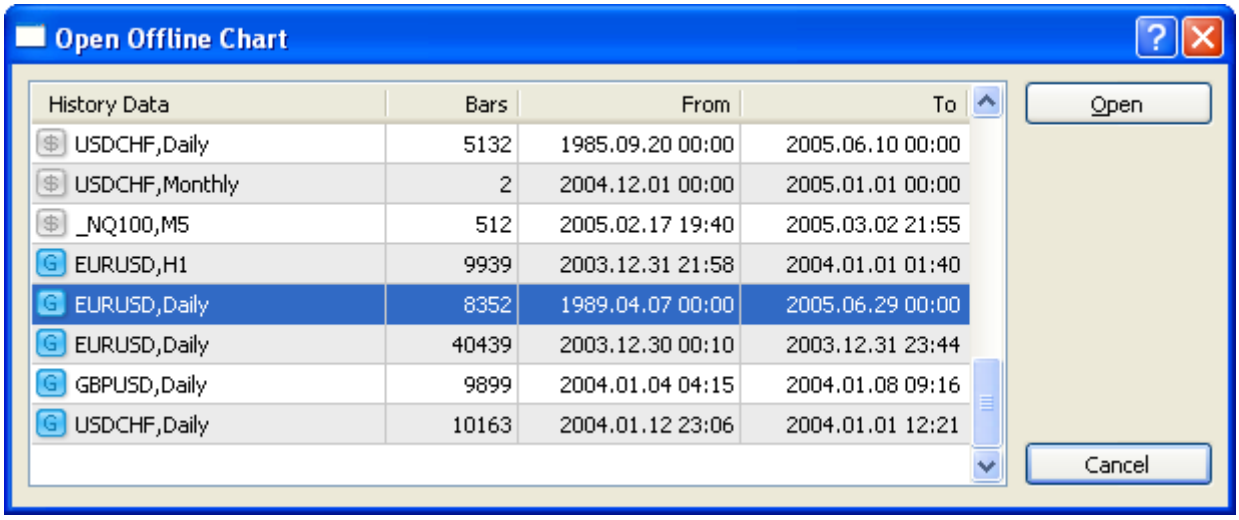
V klientském terminálu programu MetaTrader 4 jsou použity 3 způdoby modelování:



- Každý tick (založeno na dostupných nejmenších časových rámcích s fraktální interpolací každého ticku).
- Kontrolní body (je použit nejbližší časový rámec s fraktální interpolací)
- Otevřené ceny (rychlá metoda pracující s celými svíčkami)

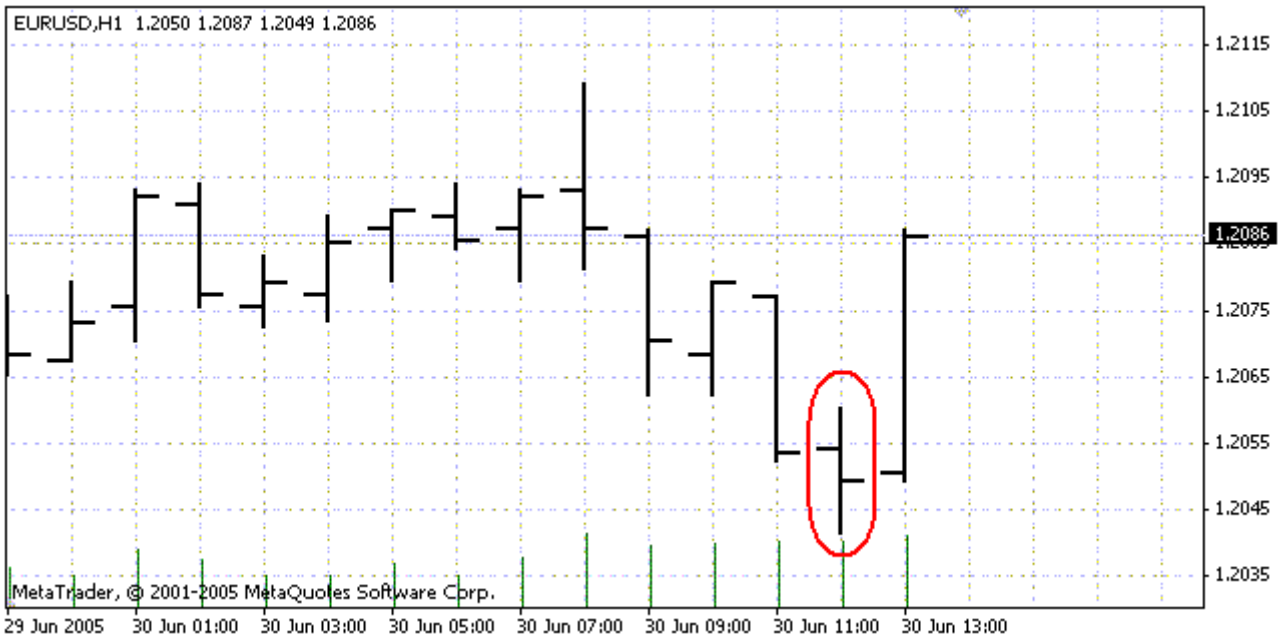
Před spuštěním testu jsou generovány střední cenové svíce, výsledek je uložen v souboru (Např.: **/tester/history/eurusd_1440_1.fxt**). Data uložená tímto způsobem umožňují pozdější značné urychlení testeru. Po aktivaci funkce pro přepočítání "Recalculate" může být provedeno přepočítávání středních dat.

Použití dat uložených dříve umožňuje provedení testu na základě vlastních dat. K tomu stačí uložit soubor v odpovídajícím formátu (*.FXT format, celkvě otevřený) do **/tester/history/** adresáře. Tyto soubory se jednoduše otevřou v terminálu jako offline tabulky prostřednictvím příkazu **File -> Open offline**.



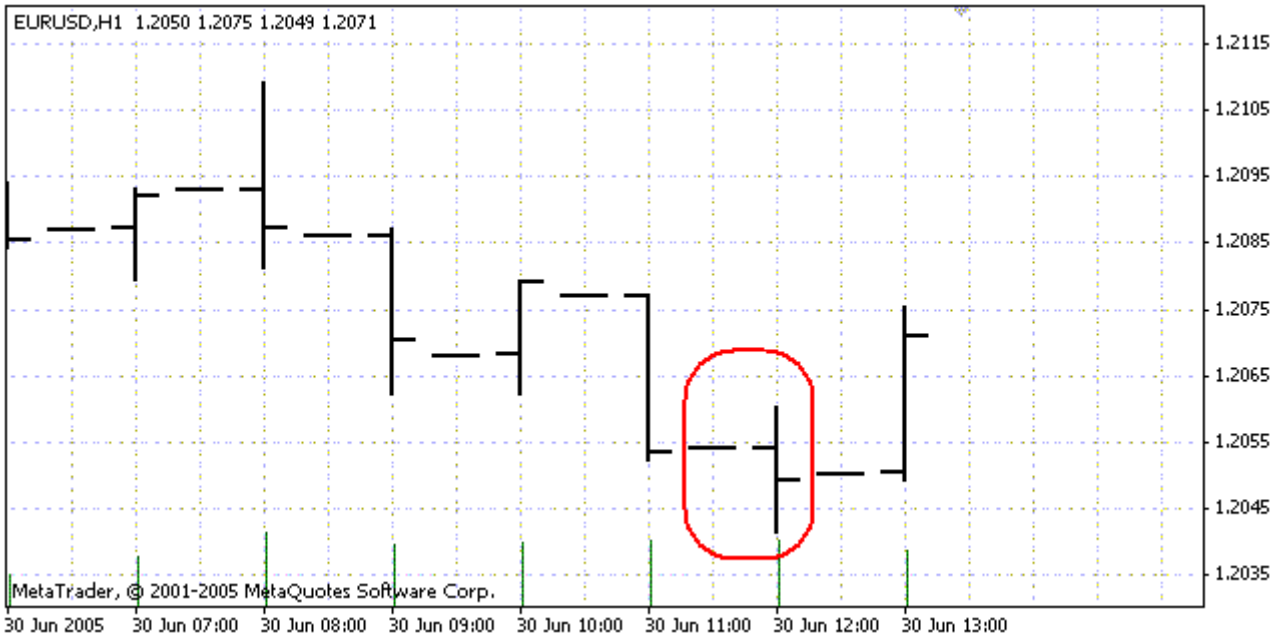
Příklady modelování

Začněme s nejjednodušší modelovací metodou založenou na jednohodinovém grafu. Nyní si prostudujeme tabulku sestavenou po jednotlivých hodinách. Datum (June, 30 2005, 12:00) je zvýrazněno červeně:



Otevřené ceny

Někteří obchodníci si nepřejí být závislí na specifičnostech modelování intratabulek a vytvářejí experty obchodující na dokončených svíčkách. Fakt, že je aktuální cenová svíčka skutečně kompletní, můžeme poznat jedině až po vyobrazení další svíce. To jsou programy, pro které je určeno modelování v režimu "**Open Price**".

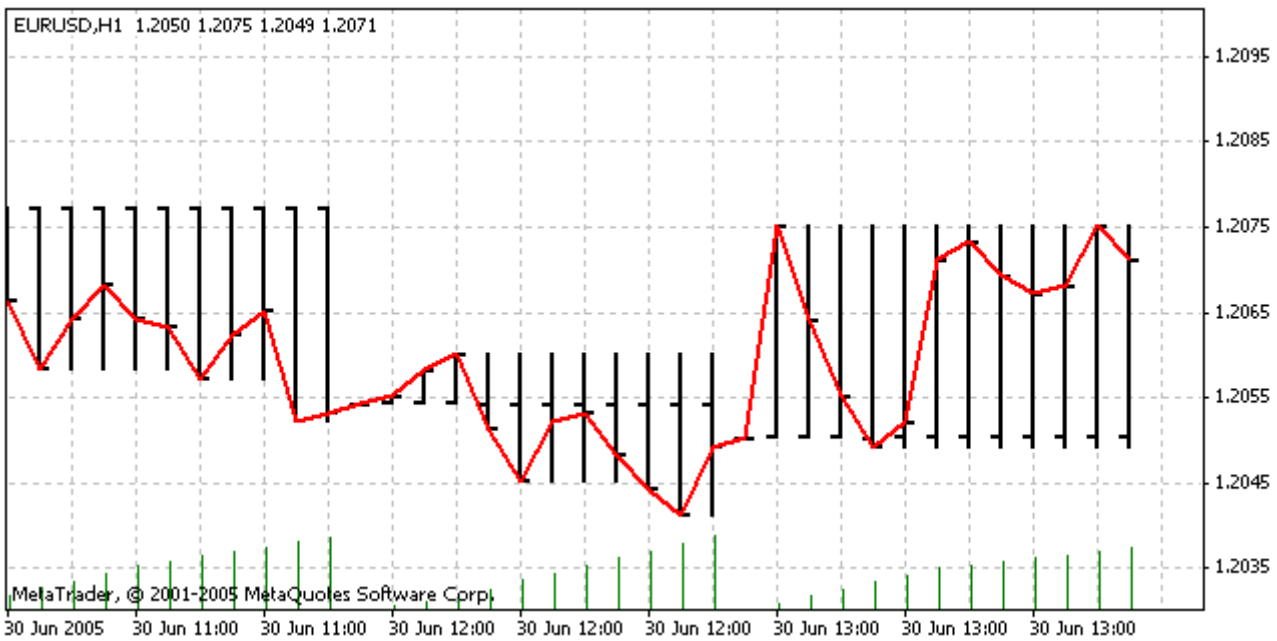


V tomto režimu je nejprve otevřena svíčka (Open = High = Low = Close, Volume=1), čímž je umožněno expertu rozpoznat zakončení předchozí lišty. Jedná se o počáteční svíčku, na jejímž základě je spuštěno testování Expertu. V další části je upřednostněna lišta aktuální, na níž však není provedeno žádné testování!

Kontrolní body (nejbližší nejmenší časové rámce)

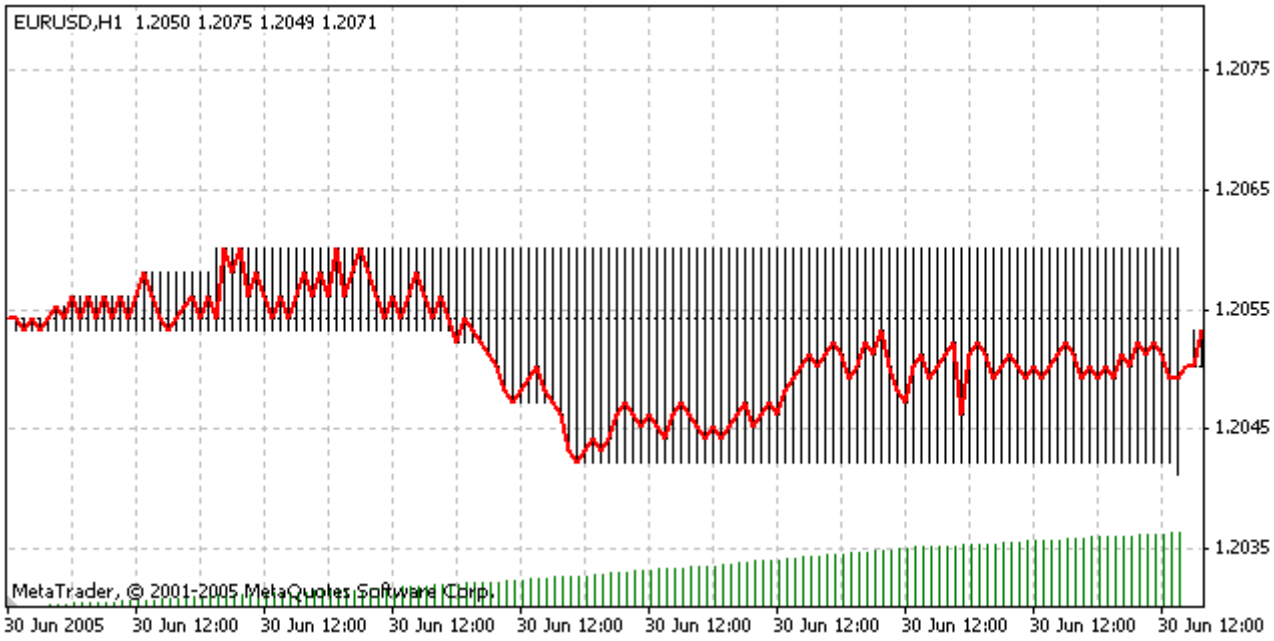
Metoda modelování kontrolních bodů je určena pro hrubý odhad expertů obchodujících v rámci jedné svíčky. K použití této metody je nutná dostupnost nejbližšího nejmenšího časového rámce. Ve většině případů data menších časových rámců nepokrývají v celkovém rozsahu časové rozpětí celého testu. Pokud nejsou k dispozici žádná data menších časových rámců vývoj svíce je generován na základě uzavření cen u předchozích 12 svíček. Tj. pohyb v rámci svíček reflektuje pohyb cen ve 12 časových rámcích, což se nazývá fraktální interpolace.

Tato generační metoda je zcela odlišná od metody „flooding“, používané v předchozích verzích klientského terminálu, jelikož předchozí metoda umožňovala striktní určování vývoje svíce. Jakmile se objeví historie dat menších časových rámců, aplikuje se do těchto nových dat fraktální interpolace. Použito však není 12, ale pouze 6 předchozích svíček. Tj., reálné ceny Open, High, Low, Close plus 2 další vygenerované ceny. Hodnota a lokace těchto dvou vygenerovaných cen závisí na pohybu cen v 6 předchozích časových rámcích.



Každý tick (založeno na dostupnosti nejmenších časových rámců s fraktální interpolací každého ticku)

Tento režim umožňuje modelování pohybu cen v rámci tabulky s největší přesností. Narozdíl od metody "control points" metoda „every-tick“ nepoužívá pro generování pouze data nejbližších menších časových rámců, nýbrž všech ostatních dostupných menších časových rámců. Přitom, pokud se současně objeví data z více než jednoho časového rámce pro dané časové rozpětí, jsou ke generování použita data menšího časového rámce. Jako u předchozích metod, i tato používá fraktální generování kontrolních bodů. Fraktální interpolace se používá opět ke generování pohybu cen mezi kontrolními body. Je možné, že dojde k vyobrazení několika identických ticků najednou. V tomto případě jsou zdvojené kotace odfiltrovány a objem takto po sobě jdoucích kotací zafixován.



Možnost použití velkého množství datových ticků musí být rozváženo. Může to mít vliv na zatížení operačního systému a rychlost testování.

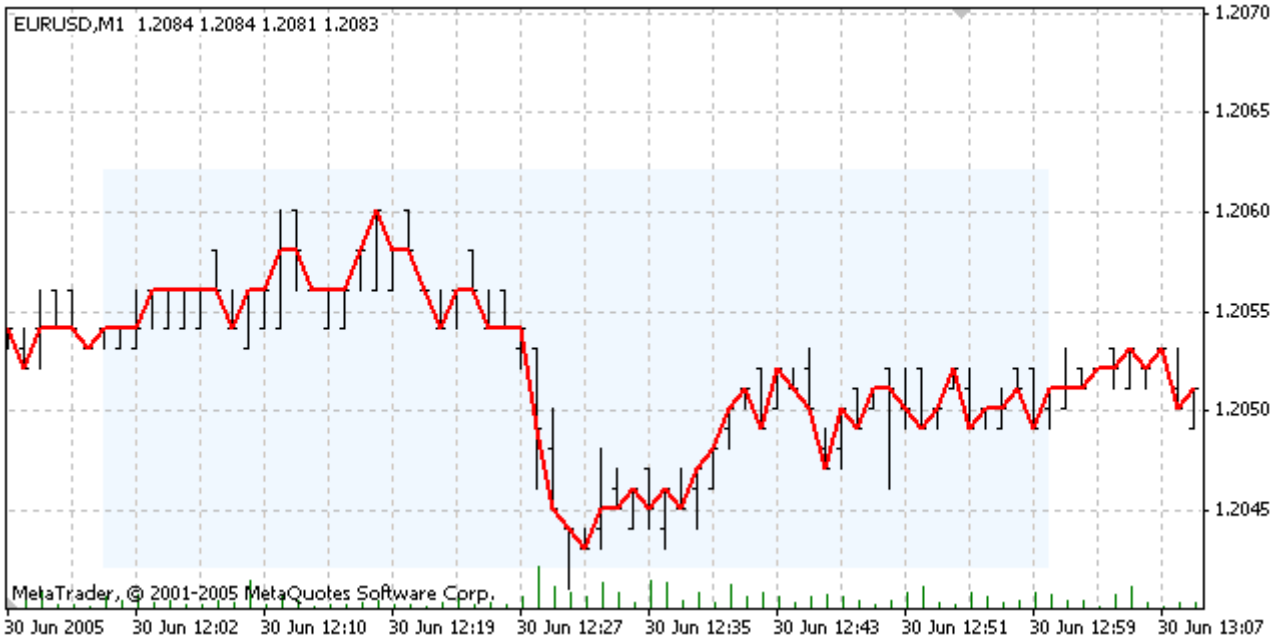
Pozor: Pokud nejsou k dispozici menší časové rámce plně pokrývající časový rámec pro testování, je zbytečné spouštět test v režimu „all tick“. Toto testování je určeno pro používání na základě dat menších časových rámců!

Používání rozsahu dat při modelování

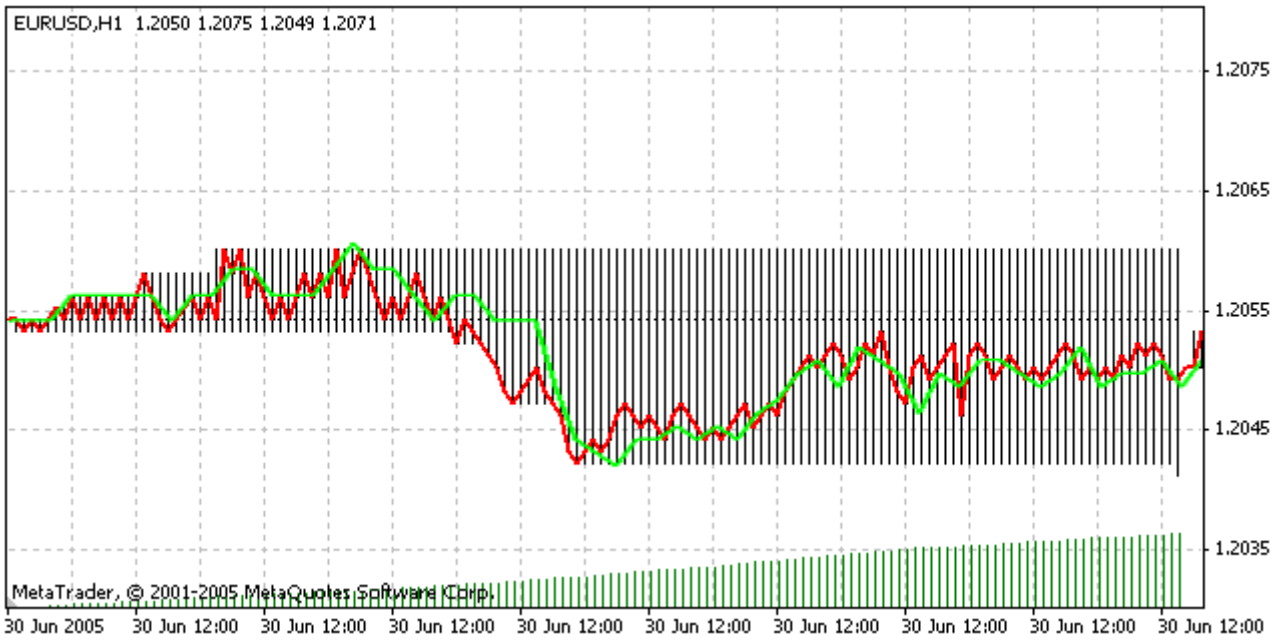
Rozsah dat může být nastaven v tabulce „Settings tab“: zakřížkujte "Use date" box a specifikujte data v polích "From:" a "To:". Datový rozsah může být použit nejen pro testování funkce expert advisor, ale i pro generování testovacích sekvencí svíček. Často není třeba generovat data celé historie, zejména při modelování v režimu Every Tick, kde může být množství nepoužitých dat velmi vysoké. Takže, pokud při vstupním generování testovací sekvence (nebo při každém zakřížkování pole "Recalculate") existovala možnost použití určitého rozsahu dat, pak svíčky přesahující daný rozsah nebudou generovány. Budou jen přepsány na výstupní sekvenci. Data, která nejsou vyloučena ze sekvence, se použijí ke správnému propočtu celé historie obdržených dat. Je třeba vědět, že není generováno ani prvních 100 svíček. Tento limit není závislý na nastaveném rozsahu dat.

Reference časových rámců M1

Ke kontrole přesnosti mezisvíčkového modelování se použije graf s datem 30.červen 2005, v rozmezí 12:00 a.m. a 1:00 p.m.



Jednoduchá změna měřítka originální „minutového“ grafu (zelenou barvou je vyznačena uzavírací cena) a její přeměna do grafu modelování v režimu „all-tick“. Data se velmi přesně shodují:



Závěry

Maximální přesnost testování a věrohodnost simulace mohou být dosaženy za pomocí menších časových rámců, pokrývajících testovací časové pásmo na 100%. To znamená, že problém kvalitního testování v režimu every-tick spočívá ve vyhledávání podrobné historie dat.

Vlastnosti testování a limity programu MetaTrader 4

Úvod

Tento odstavec umožňuje zjistit více o vlastnostech a limitech Testeru strategií v programu MetaTrader 4.

Zvláštní vlastnosti testování strategií na historických datech

- Některé funkce jsou zpracovávány/předávány bez výstupu

  Jsou to funkce: Sleep(), Alert(), SendMail(), SpeechText(), PlaySound(), MessageBox(), WindowFind(), WindowHandle(), WindowIsVisible()

- Obchodování je povoleno pouze u testovaných symbolů, neprobíhá testování portfolia

  Pokusy o obchodování s použitím jiného symbolu bude vráceno s chybovým hlášením

- Objem obchodu – počet lotů včetně vstupního objemu a kroku navyšování, marží a úroků, by měly být prováděny z aktivního účtu.

  Před testováním je nutné se ujistit, zda je aktivován alespoň jeden účet v terminálu v seznamu okna "Navigator".

- Veškeré úroky, požadavky na marže, vypršení lhůt, GTC-příkazy jsou vymodelovány

  Testování se provádí za podmínek pokud možno co nejbližším podmínkám serveru. Může se však stát, že dojde k nepřesnostem v odhadu požadavků na marže při převodu jednotek měn, vzhledem k nedostatečným informacím o aktuálním kurzu.

- Nulová svíčka jiného časového rámce pro stejný symbol v testování je vymodelována přibližně.

  Open = správné otevření, Close = správné uzavření, Low = min (Open,Close), High = max (Open,Close), Volume = konečný objem (false)

- Použitím režimu Instant Execution se předpokládá v obchodování bez slippage.
- Procesní příkazy, Open/Close jsou aktivovány bez slippage
- Testování se zastaví po „StopOut“
- Týdenní, měsíční a neregulerní časové rámce nejsou testovány
- Měna vkladu může být změněna, převodní ceny jsou však již nastaveny a ty dostupné jsou použity.
- Při provádění obchodních operací se nadále nevyskytují žádné prodlevy.

  Prodleva pro nastavení je plánována na úvod zpracování transakce.

- Historie účtu je plně k dispozici a není závislá na nastavení
- Pokud jsou aktivně používány jiné symboly a periody, je vhodné je nahrát všude, kde jen to je možné
- Při modelování typu every-tick tester načerpá všechny potřebné časové rámce pro testovaný symbol nezávisle
- Použití funkce MarketInfo vygeneruje chybu ERR_FUNCTION_NOT_ALLOWED_IN_TESTING_MODE(4059), správné informace o aktuálních cenách testovaného symbolu, rozměry úrovně stop level, velikosti bodu rozměru každého rozšířeného symbolu obsaženého v okně „quotes“, však zůstávají k dispozici.

Zvláštní vlastnosti optimalizačního procesu

- Výstup funkce journal je prázdný (včetně funkce Print())

  To je provedeno za účelem urychlení testování a vyšetření místa na disku. Pokud by byly na výstupu všechny položky, soubory journal by spotřebovaly stovky megabajtů.

- Vykreslování objektů není za běžných okolností nastaveno

  Objekty jsou deaktivovány za účelem urychlení testování .

- Je použita funkce "Skip useless results" (přeskakování nepoužitelných výsledků)

  Aby nedošlo k zahlcení tabulek a grafů výsledků testů, je zde možnost vynechávání nejhorších výsledků. Tato funkce může být aktivována z „místní nabídky“ tabulky "Optimization Results" -> Skip useless results".