○「　擬似トレード提案（その１）　」

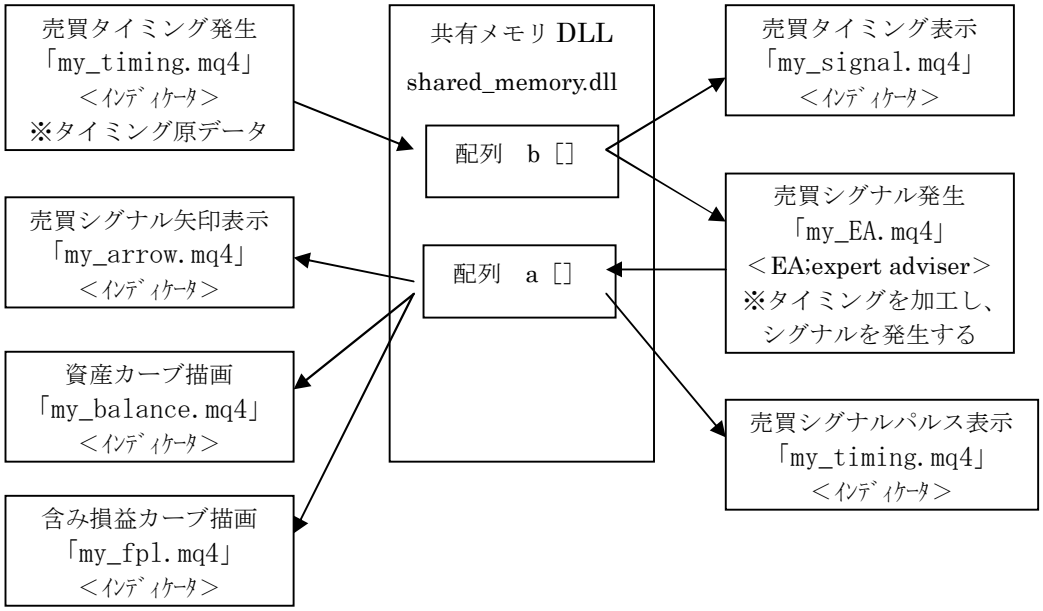
・MT4/5 のストレテジー・テスターのバックテスト機能は非常に強力ですが、幾つか不満があります。その一つは、今表示されているリアルチャート上で EA を試したらどうなるかが良く判らないことです。
アメンボが以前気に入っていた別のシステムでは、バックテスト機能は非力でしたが、リアルチャート上でのストラテジー・テスト機能があり、これは結構重宝していました。
要は、リアルチャート上で有効でないストラテジー（EA）を、いくらバックテストしたところで意味が無いわけですので、
①先ず、リアルチャート上で有効かを試し、
②有効だったら、バックテストを実施して詳しく調べる
と、言う手順をとっていました。
・そこでアメンボは、下記の手順で「MT4 用のツール」を作成・提案することにしました。
＜何れもリアルチャート上で実施＞
（１）全く擬似的なトレードで、資産カーブを描画する
①大雑把な手順の確立　←◆本稿「擬似トレード提案（その１）」
②いくつか EA を試してみる（および玉成）　・・次回以降
（２）デモトレードで、資産カーブを描画する　・・次次回以降
※（１）と（２）の違いは以下。
・・「(１)」は、リアルチャート上の過去のチャート部分でも資産カーブを描画します、
（過去のチャート部分に対してはバックテストと同じ）
・・「(２)」は、新規足に対して資産カーブを描画します。
・アメンボは、MQL 言語は勉強中のレベルですので、コード記述がエレガントではなく、
ゴチャついていることをご容赦ください、少しづつ MQL 言語に慣れていくつもりです。
また本稿では、前回投稿した DLL（shared_memory.dll）を利用しています、使用方法は
投稿済み原稿「shared_memory.dll の使い方」を参照ください。

目次：

１．擬似トレード実施（システム）概要

（１）システム

※MQL コードとデータの流れ図



①売買タイミング発生「my_timing.mq4」；＜インディケータ＞

・ema（長と短）のクロスを利用して売買タイミングを発生する。
　ema（短）が ema（長）をクロスアップすれば「買い」、
　ema（長）が ema（短）をクロスダウンすれば「売り」を基本とする。
・ただし、近すぎるタイミングは無視することとした。
　（詳細は MQL コードを参照）
・タイミングのデータは、共有メモリの配列 b[]に書き込む。
　（「買い」→「+1」、「売り」→「－1」とする）

※今回は、システム検討用にダミーのデータを発生用として設定した。（収益は無視）

②売買シグナル発生「my_signal.mq4」；＜EA＞

・共有メモリ配列 b[]からタイミングデータを読み出して、成行売買に使える
　「買い→売り」または「売り→買い」の売買シグナルに変換。
　（「買い→買い」や「売り→売り」のデータは削除する）
・変換後のデータは、売買シグナルとして、共有メモリ配列 a[]に格納する。

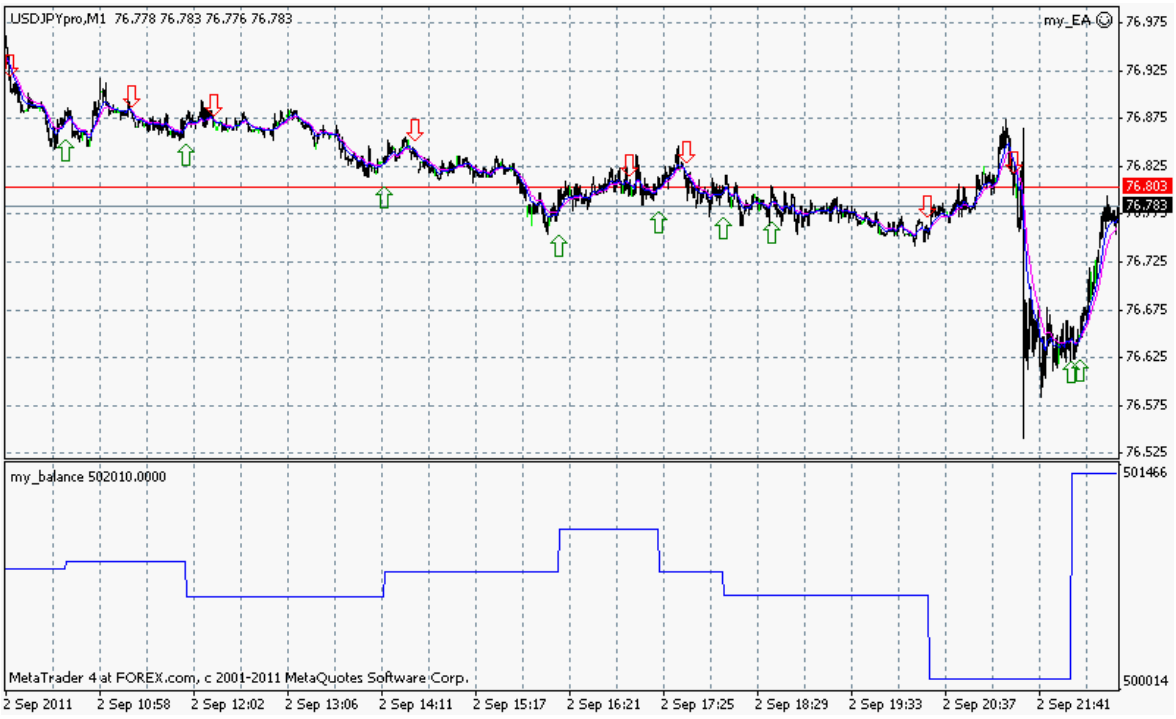
※上記と同様に、システム検討用のダミーとして設定

◎他のインディケータの内容は、それぞれの MQL コードを参照ください。

・複雑な操作は行っていません、また開発中のため、厳密な検討が抜けていること、
　コードがエレガントでないことはご容赦ください。

◎全ての「コード」はダウンロード用に設定しています。

（２）表示例１（売買シグナルと資産カーブ）

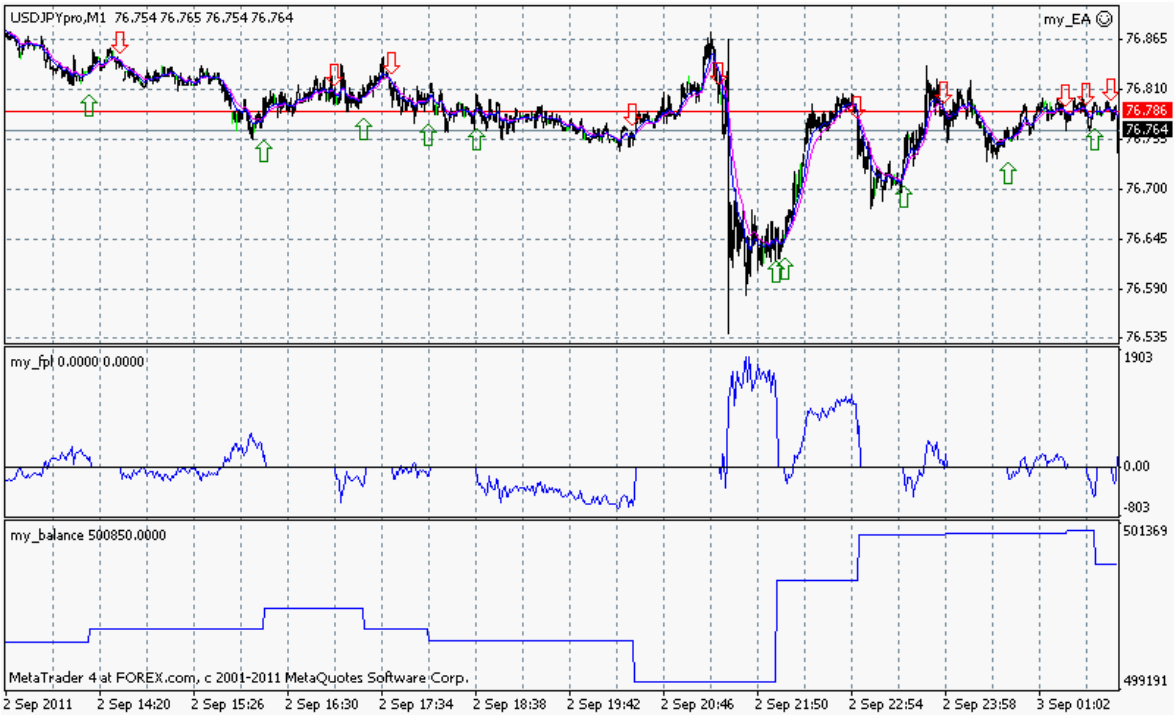


①上側のチャート；　・緑色の矢印；買いシグナル　・赤色の矢印；売りシグナル

②下側のチャート；　上記のシグナルで成行売買した場合の「資産カーブ」

③使用コード；　「my_timing.mq4」、「my_EA.mq4」、「my_arrow.mq4」、「my_balance.mq4」

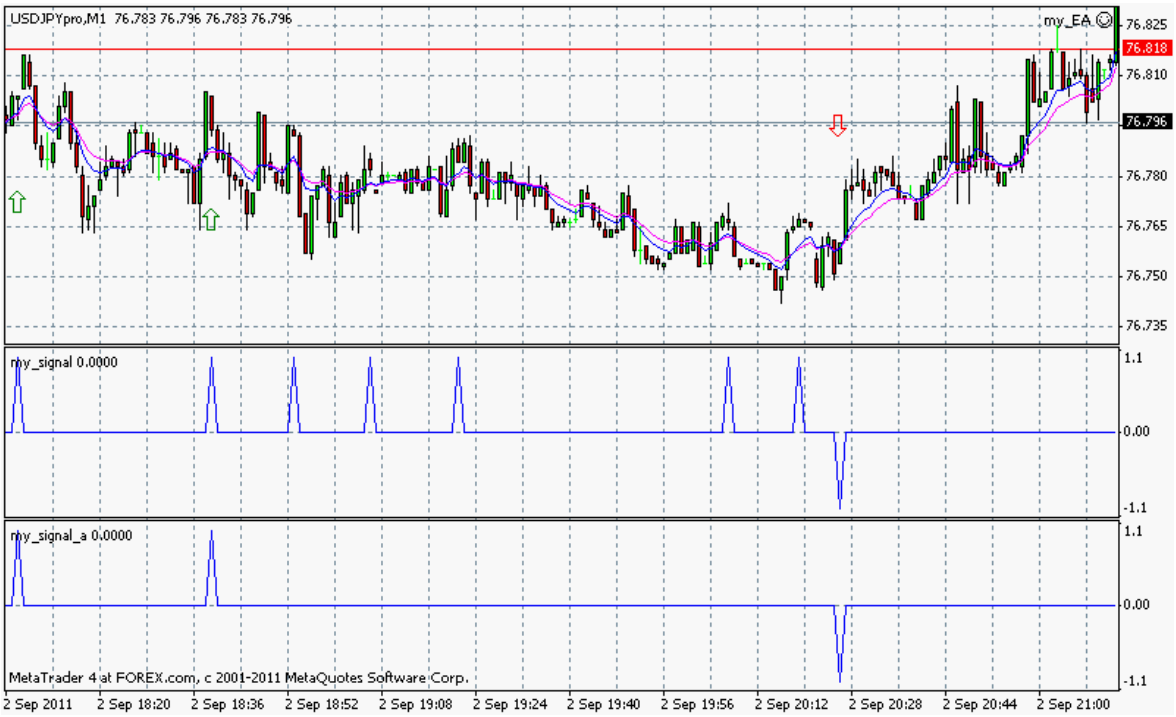
（３）表示例２（その他の組合せ表示）



※中段に「含み損益」を表示させた場合

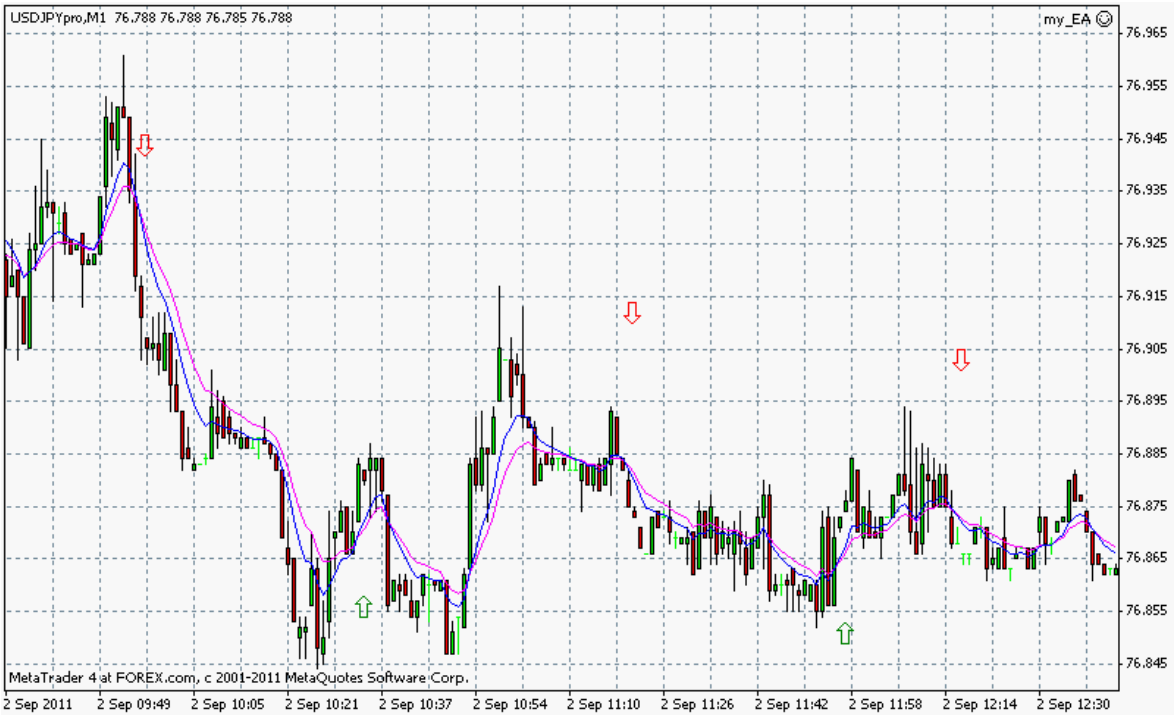
使用コード；「my_timing.mq4」、「my_EA.mq4」、「my_arrow.mq4」、「my_balance.mq4」

「my_fpl.mq4」を追加



※「売買タイミング」と「売買シグナル」の違いを示す

・中段が「売買タイミング」、下段が加工後の「売買シグナル」



※ema（長短）のクロスで売買シグナル矢印を表示、

・ただしクロス点から近すぎる次のクロス点は無視している、

・同時に「買い」と「売り」が交互に来るように加工している。

２．MQL4 コード内容一覧

（１）売買タイミング発生＜ｲﾝﾃﾞｨｹｰﾀ＞　「my_timing.mq4」

```
//        ＜インディケータ＞
//        移動平均を2本引く:長短(EMA)
//        CrossUp、Crossown 信号を共有メモリに書き込む
//
//
#import "shared_memory.dll"
    //double set_a();
    //double write_a(double,int);
    //double read_a(int);
    //double close_a();
    double set_b();
    double write_b(double,int);
    double read_b(int);
    double close_b();
#import

#property show_inputs                    //設定時にインプット・タブを表示する

#property indicator_chart_window    //チャート上に描画する
#property indicator_buffers 2       //インディケータは2個
#property indicator_color1 Magenta  //長 ema はピンク
#property indicator_color2 Blue          //単 ema は青

//指標バッファ
double Buf0[];
double Buf1[];

//外部パラメータ
extern int Long_Period = 13;
extern int Short_Period = 8;
extern int shift_ES = 0;            //移動平均の表示「右シフト」数

//初期化
int init()
{
    //set_a();
    set_b();
    //指標バッファ割り当て
    SetIndexBuffer(0,Buf0);
    SetIndexBuffer(1,Buf1);
    //表示する指標数
    IndicatorBuffers(2);                 //いくつ表示するかを指定
    //
    for(int k=0;k<=1000;k++)
    {
        //write_a(0.0,k);
        write_b(0.0,k);
    }
    //
    return(0);
}

int deinit()
```

```
  {
   //close_a();
   close_b();
   return(0);
  }

//スタート関数
int start()
{
//
    for(int k=0;k<=1000;k++)
    {
        //write_a(0.0,k);
        write_b(0.0,k);
    }
    //
    int limit = Bars-IndicatorCounted();

    for(int i=limit-1;i>=0;i--)
    {
            Buf0[i]=iMA(NULL,0,Long_Period,0,MODE_EMA,PRICE_CLOSE,i);        //長 13ema の計算
            Buf1[i]=iMA(NULL,0,Short_Period,0,MODE_EMA,PRICE_CLOSE,i);       //単  8ema の計算
    }
////
    for(int j=0;j<=1000;j++)
    {
    //CrossUp
   if(Buf1[j]>Buf0[j]  &&  Buf1[j+1]>Buf0[j+1]  &&  Buf1[j+2]<Buf0[j+2]  &&  Buf1[j+3]<Buf0[j+3]  &&
Buf1[j+20]<Buf0[j+20])
    {
                        //write_a(+1.0,j);
                        write_b(+1.0,j);
    }
    else  if(Buf1[j]<Buf0[j] && Buf1[j+1]<Buf0[j+1] && Buf1[j+2]>Buf0[j+2] && Buf1[j+3]>Buf0[j+3] &&
Buf1[j+20]>Buf0[j+20])
    {
                        //write_a(-1.0,j);
                        write_b(-1.0,j);
    }
    else
    {
            //write_a(0.0,j);
            write_b(0.0,j);
    }

    }

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

（２）売買シグナル発生＜EA＞　　「my_EA.mq4」

※本コードの仕様と課題；

```
//
//            ＜EA＞
//       売買シグナルを加工して、売買タイミングを抽出する
//--------------------------------------
#import "shared_memory.dll"
    double set_a();
    double write_a(double,int);
    double read_a(int);
    double close_a();
    double set_b();
    double write_b(double,int);
    double read_b(int);
    double close_b();
#import

//------
int open_position;
//--------------------------------------
int init()
  {
    set_a();
    set_b();
    //
    open_position=0;
    return(0);
  }
//--------------------------------------
int deinit()
  {
    close_a();
    close_b();
    return(0);
  }
//--------------------------------------
int start()
  {
    //
    //Print("start(): spread= ",spread,": Bars=",Bars);
    //for(int i=Bars-1000;i>=0;i--)//OK
    //for(int i=0;i<=Bars-1000;i++)//OK
    //open_position=0;
    for(int k=1000;k>=0;k--)
    {
        write_a(0.0,k);
    }
    //
    for(int i=1000;i>=0;i--)
    {
          //
          if(open_position==0)
          {
                  //
                  if(read_b(i)==+1.0)
```

```
		{
			//Up_arrow
			write_a(+1.0,i);
			open_position=1;
		}
		else if(read_b(i)==-1.0)
		{
			//Down_arrow
			write_a(-1.0,i);
			open_position=-1;
		}
		else if(read_b(i)==0.0)
		{
			write_a(0.0,i);
			open_position=0;
		}
	}
	//
	if(open_position==1)
	{
		if(read_b(i)==-1.0)
		{
			//
			  write_a(-1.0,i);
			open_position=0;
			//
		}else if(read_b(i)==+1.0)
		{
			//
			//write_a(0.0,i);
			//open_position=1;
			//
		}else if(read_b(i)==0.0)
		{
			//write_a(0.0,i);
			//open_position=1;
		}
	}
	//
	if(open_position==-1)
	{
		if(read_b(i)==1.0)
		{
			//
			write_a(+1.0,i);
			open_position=0;
			//
		}else if(read_b(i)==-1.0)
		{
			//
			//write_a(0.0,i);
			//open_position=-1;
			//
		}else if(read_b(i)==0.0)
		{
			//write_a(0.0,i);
			//open_position=-1;
		}
```

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

（３）売買シグナル矢印表示＜ｲﾝﾃﾞｨｹｰﾀ＞　「my_arrow.mq4」

```
//          ＜インディケータ＞
//       売買タイミングを矢印で示す
//--------------------------------------
#import "shared_memory.dll"
    double set_a();
    double write_a(double,int);
    double read_a(int);
    double close_a();
#import
//------
#property indicator_chart_window          //チャート上に描画する
//--------------------------------------
int init()
  {
    set_a();
    //
    return(0);
  }
//--------------------------------------
int deinit()
  {
    close_a();
    ObjectsDeleteAll(0);
    return(0);
  }
//--------------------------------------
int start()
  {
 //
     ObjectsDeleteAll(0);
    //
   //int limit = Bars-IndicatorCounted();

    //for(int i=Bars-1000;i>=0;i--)//OK
    //for(int i=0;i<=Bars-1000;i++)//OK
    int up_c=1,down_c=1;

    for(int i=1000;i>=0;i--)//OK
    {
            //
            if(read_a(i)==+1.0)
            {
            //Up_arrow
                        string up_a=up_c;
                        ObjectCreate("up_arrow"+up_a,OBJ_ARROW,0,Time[i],(Low[i]-0.02));
                        //ObjectCreate("up_arrow"+up_a,OBJ_ARROW,0,0,0);
```

```
                ObjectSet("up_arrow"+up_a,OBJPROP_ARROWCODE,SYMBOL_ARROWUP);
                ObjectSet("up_arrow"+up_a,OBJPROP_COLOR,Green);
                ObjectSet("up_arrow"+up_a,OBJPROP_WIDTH,2);
                //ObjectMove("up_arrow"+up_a,0,Time[i],(Low[i]-0.02));
                up_c=up_c+1;
          //
          }else if(read_a(i)==-1.0)
          {
          //Down_arrow
                string down_a=down_c;
                ObjectCreate("down_arrow"+down_a,OBJ_ARROW,0,Time[i],(High[i]+0.04));
                //ObjectCreate("down_arrow"+down_a,OBJ_ARROW,0,0,0);
                ObjectSet("down_arrow"+down_a,OBJPROP_ARROWCODE,SYMBOL_ARROWDOWN);
                ObjectSet("down_arrow"+down_a,OBJPROP_COLOR,Red);
                ObjectSet("down_arrow"+down_a,OBJPROP_WIDTH,2);
                //ObjectMove("down_arrow"+down_a,0,Time[i],(High[i]+0.02));
                down_c=down_c+1;
          //
          }else if(read_a(i)==0.0)
          {
          //
          }
          //
      }
     //
   return(0);
  }
```

（４）資産カーブ描画＜ｲﾝﾃﾞｨｹｰﾀ＞　　「my_balance.mq4」

```
//          ＜インディケータ＞
//        資産カーブ描画
//--------------------------------------
#import "shared_memory.dll"
    double set_a();
    double write_a(double,int);
    double read_a(int);
    double close_a();
#import
//------
#property indicator_separate_window
//#property indicator_maximum 501000
//#property indicator_minimum 495000
#property indicator_buffers 2
//------
double BALANCE_C[];
double ZERO_LINE[];
double balance,profit,equity,spread,ASK_,BID_;
double lots=1.0;
int open_position;
//--------------------------------------
int init()
  {
    set_a();
    //
    SetIndexStyle(0,DRAW_LINE,STYLE_SOLID,1,Blue);
    SetIndexBuffer(0,BALANCE_C);
    SetIndexStyle(1,DRAW_LINE,STYLE_SOLID,1,Black);
```

```
    SetIndexBuffer(1,ZERO_LINE);
    IndicatorBuffers(2);
    //
    //BALANCE_C[801]=AccountBalance();//なぜかここではダメ
    profit=0.0;
    open_position=0;
    return(0);
  }
//-------------------------------------
int deinit()
  {
    close_a();
    Comment("");
    return(0);
  }
//-------------------------------------
int start()
  {
    //
          int limit = Bars-IndicatorCounted();
    //
        //spread=Ask-Bid;
        spread=0.02;
        //Print("start(): spread= ",spread,": Bars=",Bars);

        BALANCE_C[1001]=AccountBalance();
    //for(int i=Bars-1000;i>=0;i--)//OK
    //for(int i=0;i<=Bars-1000;i++)//OK
    for(int i=1000;i>=0;i--)//OK
    {
          //
          if(open_position==0)
          {
                BALANCE_C[i]=BALANCE_C[i+1];
                //
                if(read_a(i)==+1.0)
                {
                      ASK_=iClose(NULL,0,i)+spread;
                      open_position=1;
                }
                else if(read_a(i)==-1.0)
                {
                      BID_=iClose(NULL,0,i);
                      open_position=-1;
                }
                else if(read_a(i)==0.0)
                {
                  //
                }
          }
          //
          if(open_position==1)
          {
                if(read_a(i)!=-1.0)
                {
                      BALANCE_C[i]=BALANCE_C[i+1];
                }else if(read_a(i)==-1.0)
                {
```

```
                        BALANCE_C[i]=BALANCE_C[i+1]+(iClose(NULL,0,i)-ASK_)*10000;
                        open_position=0;
                        ASK_=0.0;
                }
        }
        //
        if(open_position==-1)
        {
                if(read_a(i)!=1.0)
                {
                        BALANCE_C[i]=BALANCE_C[i+1];
                }else if(read_a(i)==1.0)
                {
                        BALANCE_C[i]=BALANCE_C[i+1]+(BID_-(iClose(NULL,0,i)+spread))*10000;
                        open_position=0;
                        BID_=0.0;
                }
        }
        //
   }
    //
   return(0);
  }
```

（５）含み損益カーブ描画＜ｲﾝﾃﾞｨｹｰﾀ＞　　「my_fpl.mq4」

```
//         ＜インディケータ＞
//       共有メモリを読み込んで、データを表示する
//--------------------------------------
#import "shared_memory.dll"
    double set_a();
    double write_a(double,int);
    double read_a(int);
    double close_a();
#import
//------
#property indicator_separate_window
#property indicator_buffers 2
//------
double PROFIT_C[];
double ZERO_LINE[];
double balance,profit,equity,spread,ASK_,BID_;
double lots=1.0;
int open_position;
//--------------------------------------
int init()
  {
    set_a();
    //
    SetIndexStyle(0,DRAW_LINE,STYLE_SOLID,1,Blue);
    SetIndexBuffer(0,PROFIT_C);
    SetIndexStyle(1,DRAW_LINE,STYLE_SOLID,1,Black);
    SetIndexBuffer(1,ZERO_LINE);
    IndicatorBuffers(2);
    //
    profit=0.0;
    open_position=0;
    return(0);
```

```
  }
//+------------------------------------------------------------------+
int deinit()
  {
   close_a();
   Comment("");
   return(0);
  }
//+------------------------------------------------------------------+
int start()
  {
      //spread=Ask-Bid;
      spread=0.02;

   //for(int i=Bars-1000;i>=0;i--)//OK
   //for(int i=0;i<=Bars-1000;i++)//OK
   for(int i=1000;i>=0;i--)//OK
   {
      ZERO_LINE[i]=0.0;
   //
        if(open_position==0)
        {
              PROFIT_C[i]=0.0;
              //
              if(read_a(i)==+1.0)
              {
                    ASK_=iClose(NULL,0,i)+spread;
                    open_position=1;
              }
              else if(read_a(i)==-1.0)
              {
                    BID_=iClose(NULL,0,i);
                    open_position=-1;
              }
        }
        //
        if(open_position==1)
        {
              if(read_a(i)!=-1.0)
              {
                    PROFIT_C[i]=(iClose(NULL,0,i)-ASK_)*10000;
              }else if(read_a(i)==-1.0)
              {
                    //
                    open_position=0;
                    ASK_=0.0;
              }
        }
        //
        if(open_position==-1)
        {
              if(read_a(i)!=1.0)
              {
                    PROFIT_C[i]=(BID_-(iClose(NULL,0,i)+spread))*10000;
              }else if(read_a(i)==1.0)
              {
                    //
                    open_position=0;
```

```
                                BID_=0.0;
                        }
                }
                //
        }
         //
        return(0);
    }
```

（６）売買シグナルパルス表示＜ｲﾝﾃﾞｨｹｰﾀ＞「my_signal_a.mq4」

```
//          ＜インディケータ＞
//        共有メモリ a[]を読み込んで、データを表示する
//
#import "shared_memory.dll"
     double set_a();
     double write_a(double,int);
     double read_a(int);
     double close_a();
#property indicator_separate_window
#property indicator_buffers 1

double MEMORY_D[1000];

int init()
   {
     set_a();
     //
     SetIndexStyle(0,DRAW_LINE,STYLE_SOLID,1,Blue);
     SetIndexBuffer(0,MEMORY_D);
     IndicatorBuffers(1);
     return(0);
   }

int deinit()
   {
     close_a();
     return(0);
   }

int start()
   {
     //
     //int limit = Bars-IndicatorCounted();
     //
     for(int k=0;k<=1000;k++)
     {
          MEMORY_D[k]=read_a(k);
     }
     //
     return(0);
   }
```

（７）売買タイミング表示＜ｲﾝﾃﾞｨｹｰﾀ＞　　「my_signal.mq4」

```
//         ＜インディケータ＞
//       共有メモリb[]を読み込んで、データを表示する
//
#import "shared_memory.dll"
     double set_b();
     double write_b(double,int);
     double read_b(int);
     double close_b();
#import

#property indicator_separate_window
#property indicator_buffers 1

double MEMORY_D[1000];

int init()
   {
     set_b();
     //
     SetIndexStyle(0,DRAW_LINE,STYLE_SOLID,1,Blue);
     SetIndexBuffer(0,MEMORY_D);
     IndicatorBuffers(1);
     return(0);
   }

int deinit()
   {
     close_b();
     return(0);
   }

int start()
   {
     //
           //int limit = Bars-IndicatorCounted();
     //
     for(int k=0;k<=1000;k++)
     {
          MEMORY_D[k]=read_b(k);
     }
     //
     return(0);
   }
```

以　上