○「　Win32API；EA にブレーク・ポイントを設定する　」

・アメンボです、
　EA のデバッグ時に「ブレーク・ポイント」を設定できたら良いと思ったことは
　ありませんか？　MT5 には存在する（？）様ですが、MT4 では専用機能が見つかりません。

・前々回の投稿「12. Win32API；任意フォルダ内のファイル・アクセス.doc」で
Win32API を調べ始めて以来、実に様々な機能が利用出来ことを理解しましたが、
イマイチ、「どうしても使いたい」と言う気になりませんでした。
（本当に必要な機能なら、MQL に取り込まれる筈ですし）
ただ、明らかに MQL が取りこぼしているチョットした機能は確かにあります。

・本稿では「アメンボがあたっらいいな！と思っていた機能の一つ」を
Win32API を利用して実現した例を投稿することとしました。

※本稿「MQL4 コード」はダウンロード用に同時掲載してあります。

１．作成する MQL4 コードの仕様

（１）「Bollin_EA_08.mq4」への修正
　①バックテスト（ビジュアル・モード）時に、チャートのプロファイルを切り替える。
　②売買の仕掛け（エントリーポイント）で一旦停止し、
　　希望する「データ値」をコメントとしてチャート上に表示する。
（２）修正後のコードは「Bollin_EA_08_win32_try.mq4」としました。

２．Win32API による機能拡張の概要

（１）基本ー利用する「API 機能」によって、inport する DLL を下記の３種類から選択します。

DLL ファイル名	主な API 機能	関数例
kernel32.dll	プロセス、メモリや周辺装置を管理	_lopen ( ) ・ファイル・オープン
user32.dll	ウインドウベースの ユーザー・インターフェース管理	PostWindowA(,WM_COMMAND,,) ・ウインドウへメッセージ送信
		keybd_event(0x13,,,) ・バーチャル・キーボード
ghi32.dll	文字列やグラフィックスの 描画に関するｻｰﾋﾞｽ提供	！未だ、使ったこと無し！

（2）MQL4 で「Win32API」を利用する際の注意

A．DLL から関数をインポート

・使用したい関数を下記の形式でインポートします。
例えば「kernel32.dll」に属する「_lopen ()」を使いたいのであれば、コードの頭部分で

```
#import "kernel32.dll"
    int _lopen (string path, int of);
#import
```

と、インポート（兼、プロトタイプ宣言）する必要があります。

B．「WinUser32.mqh」に注意

・MT4 の「**experts\include**」フォルダに入っている「**WinUser32.mqh**」はとても重要です、
プリントして参照する価値があります。（下記にその一部を示します）

```
#define     copyright "Copyright ｳ 2004, MetaQuotes Software Corp."
#define     link          "http://www.metaquotes.net/"
#import "user32.dll"
    //---- messages
    int         SendMessageA(int hWnd,int Msg,int wParam,int lParam);
    int         SendNotifyMessageA(int hWnd,int Msg,int wParam,int lParam);
    int         PostMessageA(int hWnd,int Msg,int wParam,int lParam);
    void        keybd_event(int bVk,int bScan,int dwFlags,int dwExtraInfo);
    void        mouse_event(int dwFlags,int dx,int dy,int dwData,int dwExtraInfo);
    //---- windows
    int         FindWindowA(string lpClassName ,string lpWindowName);
    int         SetWindowTextA(int hWnd,string lpString);
    ・・・・・・・・・・・・・・
    int         SwapMouseButton(int fSwap);
    int         GetAsyncKeyState(int vKey);
    bool        SetCursorPos(int X, int Y);

#import
```

重要である理由；

①「#include <WinUser32.mqh>」とインクルードしておけば、
このヘッダファイル内で、「#import "**user32.dll**"」が既に宣言されており、
「#import "**user32.dll**"」～「#import」間で宣言されている関数は、
「A.」で述べたような形で、改めて宣言する必要は無いのです！！

②しかし、この場合は、**特に注意**が必要です。
・例えば「PostMessageA()」ですが、マイクロソフトの C 言語仕様では「PostMessage()」
となります、つまり、「A」がありません。VB 言語仕様では「A」付きの様ですが！）
・何が言いたいかと言うと、
「#import "**user32.dll**"」～「#import」間で宣言されている関数は、
MQL4 では、そのままの書式で使用する必要があります。
例えば PostMessageA()は正常動作しますが、PostMessage()は動きません。
（　アメンボの理解です、間違っていたらすいません　）

３．本稿で使用する Win32API について

（１）本稿のコードで使用した「コマンド」を解説

・全て、「user32.dll」に属する関数です

int　GetForegroundWindow();

・機能；フォアグラウンドウィンドウ（ 現在ユーザーが作業中のウィンドウ）ハンドルを返す、
　ウィンドウがフォーカスを失ったなどの特定の状況下で NULL になる場合もある。

int　PostMessageA(int hWnd,int Msg,int wParam,int lParam);

・機能；ウインドウのメッセージ・キューにメッセージを送る、
・成功すると；　「0」以外を返す
・失敗すると；　「0」を返す
・int hWnd；　　メッセージを届けたいウインドウのハンドル
・int Msg；　　ウインドウにポストするメッセージを指定、
　　「WM_COMMAND」や「WM_CLOSE」等がある。(WinUser32.mqh を参照)
・int wParam；　メッセージの第１情報を指定する（内容はメッセージに依存する）
・int lParam；　メッセージの第２情報を指定する（内容はメッセージに依存する）

※ウインドウへのメッセージは、余りに各種ありアメンボでは把握しきれません、
　本稿ではアメンボがとりあえず必要とした機能に関するものだけを記載します。
　（　正直言うと、Win32API に関してアメンボは超初心者なので、良くは理解していない　）

※北米の MQL4 フォーラムを覗くと、Win32API に関して様々な情報が観られます。

（例）メッセージ；int Msg＝「**WM_COMMAND**」の場合
・int wParam；　擬似的にクリック（選択）するメニュー項目
・int lParam；　通常「0」で使う

＜本稿；使用例＞

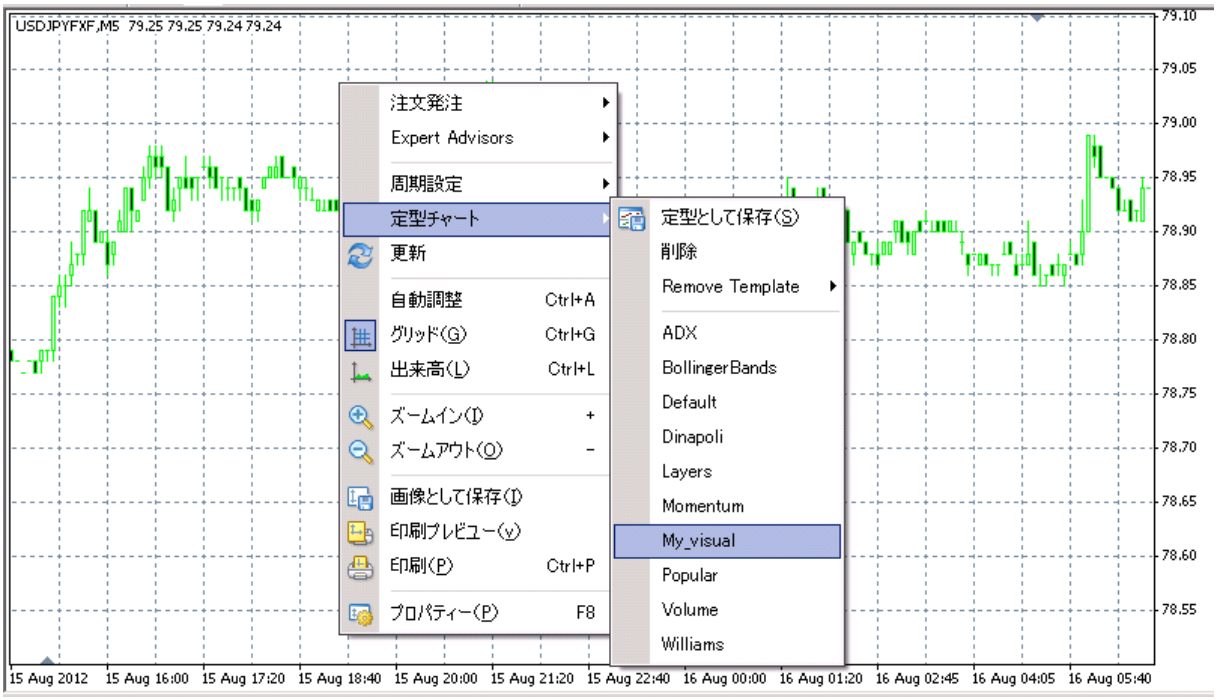
```
int hWnd=GetForegroundWindow();
//PostMessageA(hWnd,WM_COMMAND,34800,0);//チャートの第１テンプレートを選択
PostMessageA(hWnd,WM_COMMAND,34806,0);  //チャートの第７テンプレートを選択

int main=GetForegroundWindow();          //現在作業中のウィンドウ・ハンドル取得
PostMessageA(main,WM_COMMAND,0x57a,0);  //[Pause]を選択（押す）
```

※wParam は１６進でも１０進使えます。（上記）

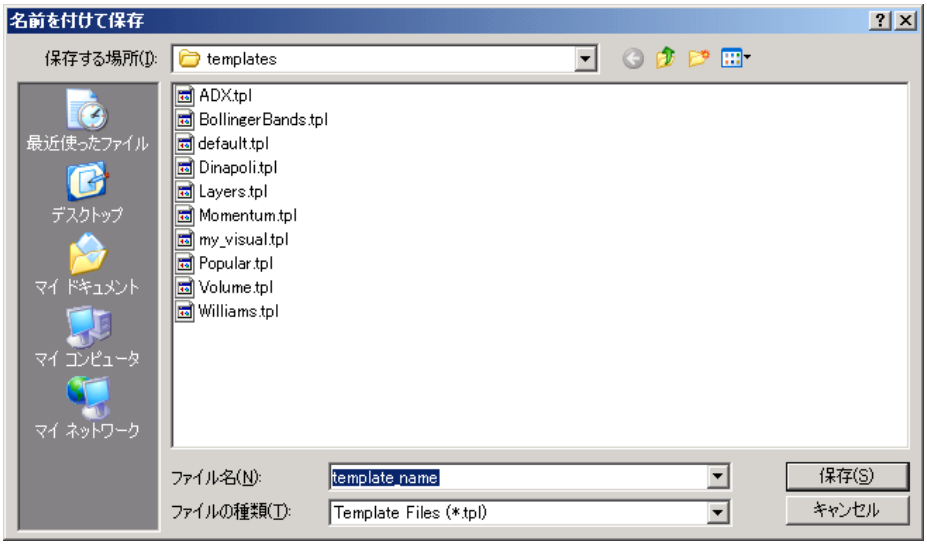
＜補足説明１＞

※「PostMessageA(hWnd,WM_COMMAND,34806,0);//チャートの第７テンプレートを選択」の意味；

・下記に示す様に、チャートのテンプレートの「アルファベット順で７番目」の、
テンプレートを選択することを意味します。
・アメンボが「バックテストのビジュアル・モード」用に設定（自作）して保存したものが、
「My_visual」です。



・チャート上で、［右クリック］－［定型チャート］－［定型として保存］としても確認可能。



＜補足説明２＞

※『　PostMessageA(hWnd,WM_COMMAND,メニューID,0);　』
北米のフォーラムを覗くと、上記で使用可能な「メニューID」が実に沢山調べられています、
しかし注意しなければならないことは、MT4 のバージョンアップによって「この ID」が
変更されてしまう可能性があることです。
また、余り乱用するとメタクウォーク社側が何らかの制限を加えてくるかも知れません。

・でも、色々知りたいというのが人情でしょうから、
各人の責任において対応することを条件として、情報を一つ！
北米の MQL4 フォーラム記事の
「#define's for known commands that can be used for PostMessageA()」
で、紹介されている Robert Strube 氏による「aswincmd.mqh」が参考になります。
（MQL4 関数でサポートされていない機能に限定して活用するのがポイントです）

void　keybd_event(int bVk,int bScan,int dwFlags,int dwExtraInfo);
・int bVk　　；仮想キーボード
・int bScan　　；ハードウエア・スキャンコード、通常「0」
・int dwFlags　；動作指定フラグ（0x0002＝キーを離す、指定しないとキーを押す操作となる）
・int dwExtraInfo；追加情報、通常「0」

※仮想キーボード一覧は「**５．仮想キーコード一覧**」に添付しました。（量が多いので！）

＜本稿；使用例＞

```
keybd_event(19,0,0,0);          ---  [Pause]を押す
Sleep(10);                      ---  10msec 休み
keybd_event(19,0,2,0);          ---  [Pause]を離す

keybd_event(0x13,0,0,0);        ---  [Pause]を押す
Sleep(10);                      ---  10msec 休み
keybd_event(0x13,0,2,0);        ---  [Pause]を離す
```

※bVk は１６進でも１０進使えます。（上記）

（２）コードの頭に必要な記述
・本稿のコードでは、「#include <WinUser32.mqh>」を宣言しているので、
改めて「PostMessageA」と「keybd_event」はインポート宣言する必要は有りません。
そこで、「GetForegroundWindow」のみ宣言すれば良いことになります。（下記を参照）

```
#include <WinUser32.mqh>
#import "user32.dll"
   int GetForegroundWindow();
#import
```

４．「Bollin_EA_08_win32_try」実行結果

（１）「ビジュアル・テスト」モードで、チャートのテンプレートを切り替える

①通常のテンプレート；

・ビジュアル・モードでテストをすると、このモードから始まるので、
　アメンボはいつも一旦止めて、チャート設定をやり直していた。
（　どこかに設定するタブがあるのかも知れませんが、アメンボには判らなかった）

②init()部「PostMessageA(hWnd,WM_COMMAND,34806,0);」による切替結果

（２）myPause()部「keybd_event(0x13,0,0,0);」等による一時停止結果

・テストを再開するには、

の［＞＞］をクリックします。

５．MQL4 コード

※「**青書き**」部分が、「Bollin_EA_08.mq4」に追記した部分です。

```
//+------------------------------------------------------------------+
//|                                      Bollin_EA_08_win32_try.mq4 |
//|                                                          amenbo |
//|                                                                 |
//+------------------------------------------------------------------+
#property copyright "amenbo"

//
//=====Win32API========
// Win32API 使用宣言
#include <WinUser32.mqh>
#import "user32.dll"
   int GetForegroundWindow();
#import
//======================
//

#define Magic_ID 1930

extern double Lots=1;
extern int max_position=1;//最大保有ポジション数
// 損益設定
//「①②」は最適化の実施後での設定値
extern double profit=0.60;
extern double loss=0.60;
extern double profit_2=0.70;
extern double loss_2=0.50;
// ①Bollin の幅
extern int period_bollin=40;
extern double IKA=0.4;
extern double IZYOU=0.2;
// ②フィルター設定
extern int shortPeriod_buy=50;
extern int mediumPeriod_buy=120;
extern double short_buy=-0.0044;
extern double long_buy=-0.0006;
//     ・・sell 専用
extern int shortPeriod_sell=35;
extern int mediumPeriod_sell=180;
extern double short_sell=0.0044;
extern double long_sell=0.0006;
//-----------------------------
//  ③トレンド判断
extern int trendPeriod=300;//70
extern double _up=-0.00024;//-0.0004
extern double _down=0.00024;
extern double div_=0.04;
//====================================
static datetime lastbar;
datetime in_time,out_time;
int barsTotal;
//売買シグナル判定用の配列データ
```

```
double EMA_[10];
double Bol_hi[10];
double Bol_lo[10];
double Price_01[10000];
double Trend_[10000];
//////////////////////////
int init()
{
   //
   lastbar=Time[1];
   barsTotal = Bars;
   in_time=Time[20];
   //
   //=====Win32API========
   if(IsTesting() && IsVisualMode() && IsDllsAllowed())
   {
  //フォアグラウンドウィンドウ（ 現在ユーザーが作業しているウィンドウ）のハンドルを返す
      int hWnd=GetForegroundWindow();
      //PostMessageA(hWnd,WM_COMMAND,34800,0);//the first template
      PostMessageA(hWnd,WM_COMMAND,34806,0);//the 7th template
   }
   //
   keybd_event(0x13,0,2,0);//[Pause]を OFF する
   //
   Comment("");
   //=====================
   //
   return(0);
}
int deinit()
{
   //
   //=====Win32API========
   keybd_event(0x13,0,2,0);
   //=====================
   //
   return(0);
}
//////////////////////////////////////
int start()
{
   if(Bars<100 || IsTradeAllowed()==false) return;

   //  NewBar かチェック
   if(IsNewBar() && (Bars>barsTotal))
   {

      barsTotal=Bars;

      //＜ポジションが在る場合の処理＞
      if(OrdersTotal()>=1)
      {
         for(int i=0;i<OrdersTotal();i++)
         {
            if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==false) break;
            if(OrderMagicNumber()!=Magic_ID || OrderSymbol()!=Symbol()) continue;

            if(OrderMagicNumber()==Magic_ID && OrderSymbol()==Symbol())
```

```
      {
         bool ikisugi=(Time[0]-in_time)>=(300*Period()*60);

         if(OrderType()==OP_BUY)
         {
            double kachi_buy_price=OrderOpenPrice()+profit;
            double make_buy_price=OrderOpenPrice()-loss;
            //
            bool kachi_buy=(Bid>=kachi_buy_price);
            bool make_buy=(Bid<=make_buy_price);
            //
            if(kachi_buy || make_buy || ikisugi)
            {
               OrderClose(OrderTicket(),Lots,Bid,0,Blue);
               continue;
            }
         }

         if(OrderType()==OP_SELL)
         {
            double kachi_sell_price=OrderOpenPrice()-profit;
            double make_sell_price=OrderOpenPrice()+loss;
            //
            bool kachi_sell=(Ask<=kachi_sell_price);
            bool make_sell=(Ask>=make_sell_price);
            //
            if(kachi_sell || make_sell || ikisugi)
            {
               OrderClose(OrderTicket(),Lots,Ask,0,Blue);
               continue;
            }

         }
      }
   }
}//end_of_if(OrdersTotal()>=1)

//＜新規ポジションの設定と方針＞
//アップトレンドでは⇒buyIN のみ実施（順張り）
//ダウントレンドでは⇒sellIN のみ実施（順張り）
//レンジ相場では　　⇒buy と sell（逆張り？―
//if(OrdersTotal()<=max_position)
if(OrdersTotal()==0)
{

   //判断用の配列を準備
   ArraySetAsSeries(Bol_hi,true);
   ArraySetAsSeries(Bol_lo,true);
   //
   for(int j=0;j<=6;j++)
   {
     Bol_hi[j]=iBands(NULL,0,period_bollin,2,0,PRICE_OPEN,MODE_UPPER,j);  //2_σ
     Bol_lo[j]=iBands(NULL,0,period_bollin,2,0,PRICE_OPEN,MODE_LOWER,j);  //2_σ
   }

   //判断用の共通指標を準備
   bool Subtract=((IZYOU<(Bol_hi[0]-Bol_lo[0])) && ((Bol_hi[0]-Bol_lo[0])<IKA));
   bool hanareta=((Time[0]-in_time)>=(19*Period()*60));
```

```
         bool TREND_UP=((slope_kaiki(0,trendPeriod))<=(_up)) &&
(call_hensa(0,trendPeriod)<=div_);
         bool TREND_DOWN=((slope_kaiki(0,trendPeriod))>=(_down)) &&
(call_hensa(0,trendPeriod)<=div_);

         ////if(TREND_UP)
         ////{
            //USDJPY の上昇相場（円安）
            //買い条件は整ったか
            bool cross_up_1=((High[1]>Bol_hi[1]) && (Open[0]>Bol_hi[0]));
            bool cross_up_2=((Open[3]<Bol_hi[3]) && (Open[2]<Bol_hi[2]));
            bool cross_up_3=((Open[4]<Bol_hi[4]) && (Open[3]<Bol_hi[3]));
            bool cross_up_4=((Open[5]<Bol_hi[5]) && (Open[4]<Bol_hi[4]));
            bool cross_up_5=((Open[6]<Bol_hi[6]) && (Open[5]<Bol_hi[5]));
     bool cross_up=(cross_up_1 && (cross_up_2 || cross_up_3 || cross_up_4 || cross_up_5));
            //条件の補足
bool cross_up_rapid=((High[2]>Bol_hi[2]) && (High[1]>Bol_hi[1]) && (Open[0]>Bol_hi[0]));
            //Filter；本当に買いか
            double slope_s_buy=slope_kaiki(0,shortPeriod_buy);
            double slope_m_buy=slope_kaiki(0,mediumPeriod_buy);
            bool SLOPE_BUY=((slope_s_buy<=short_buy) && (slope_m_buy<=long_buy));
            //
            if(((cross_up || cross_up_rapid) && SLOPE_BUY) && Subtract && hanareta)
            {

OrderSend(Symbol(),OP_BUY,Lots,Ask,0,0,0,"my_Buy_order",Magic_ID,Green);
               in_time=Time[0];
               //=====Win32API========
               // for VisualMode debug
               Comment("");
               myPause("Buy: short_s_buy= ",slope_s_buy," : slope_m_buy= ",slope_m_buy);
                //======================
            }

         ////} else if(TREND_DOWN)
         ////{
         ///+
            //USDJPY の下降相場（円高）
            //売り条件は整ったか
            bool cross_down_1=((Low[1]<Bol_lo[1]) && (Open[0]<Bol_lo[0]));
            bool cross_down_2=((Open[3]>Bol_lo[3]) && (Open[2]>Bol_lo[2]));
            bool cross_down_3=((Open[4]>Bol_lo[4]) && (Open[3]>Bol_lo[3]));
            bool cross_down_4=((Open[5]>Bol_lo[5]) && (Open[4]>Bol_lo[4]));
            bool cross_down_5=((Open[6]>Bol_lo[6]) && (Open[5]>Bol_lo[5]));
            bool  cross_down=(cross_down_1  &&  (cross_down_2  ||  cross_down_3  ||
cross_down_4 || cross_down_5));
            //条件の補足
            bool  cross_down_rapid=((Low[2]<Bol_lo[2])  &&  (Low[1]<Bol_lo[1])  &&
(Open[0]<Bol_lo[0]));
            //Filter；本当に売りか
            double slope_s_sell=slope_kaiki(0,shortPeriod_sell);
            double slope_m_sell=slope_kaiki(0,mediumPeriod_sell);
            bool SLOPE_SELL=(slope_s_sell>=short_sell) && (slope_m_sell>=long_sell);
            //
            if(((cross_down || cross_down_rapid) && SLOPE_SELL) && Subtract && hanareta)
            {
```

```
OrderSend(Symbol(),OP_SELL,Lots,Bid,0,0,0,"my_Sell_order",Magic_ID,Red);
                in_time=Time[0];
                //=====Win32API========
                // for VisualMode debug
                Comment("");
           myPause("Sell: short_s_sell= ",slope_s_sell," : slope_m_sell= ",slope_m_sell);
               //=====================
             }
          //*/
          ////} else if(TREND_UP==false && TREND_DOWN==false)
          ////{
             //レンジ相場
             //内容は検討中です
          ////}

       }//end_of_if(OrdersTotal()<=max_position)


    } else
    {
       //［５分足］内で起こる急変に対応する処理を記載する
       //  バックテストでは確認が困難な部分

       if(OrdersTotal()>=1)
       {
          for(int ii=0;ii<OrdersTotal();ii++)
          {
             if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==false) break;
             if(OrderMagicNumber()!=Magic_ID || OrderSymbol()!=Symbol()) continue;

             if(OrderMagicNumber()==Magic_ID && OrderSymbol()==Symbol())
             {

                if(OrderType()==OP_BUY)
                {
                   double i_kachi_buy_price=OrderOpenPrice()+profit_2;
                   double i_make_buy_price=OrderOpenPrice()-loss_2;
                   //
                   bool i_kachi_buy=(Bid>=i_kachi_buy_price);
                   bool i_make_buy=(Bid<=i_make_buy_price);
                   //
                   if(i_kachi_buy || i_make_buy)
                   {
                      OrderClose(OrderTicket(),Lots,Bid,0,Blue);
                      continue;
                   }
                }

                if(OrderType()==OP_SELL)
                {
                   double i_kachi_sell_price=OrderOpenPrice()-profit_2;
                   double i_make_sell_price=OrderOpenPrice()+loss_2;
                   //
                   bool i_kachi_sell=(Ask<=i_kachi_sell_price);
                   bool i_make_sell=(Ask>=i_make_sell_price);
                   //
                   if(i_kachi_sell || i_make_sell)
                   {
```

```
                        OrderClose(OrderTicket(),Lots,Ask,0,Blue);
                        continue;
                    }

                }
            }
        }
    }//end_of_if(OrdersTotal()>=1)

}

return(0);
}
//////////////////////  以下は関数類  ////////////////////////////////////
bool IsNewBar()
{
datetime curbar = Time[0]; // Open time of current bar
if (lastbar!=curbar)
   {
    lastbar=curbar;
    return (true);
   }
  return(false);
}
//-----------------------------------------
double slope_kaiki(int start_s,int period_s)
{
   ArraySetAsSeries(Price_01,true);
   Price_01[0]=Open[0];
   for(int j_=(start_s+1);j_<=((start_s+1)+(2*period_s));j_++)
   {
         Price_01[j_]=(1.0/4.0)*(Open[j_]+High[j_]+Low[j_]+Close[j_]);
   }
   double slope=kaiki_sen(Price_01,start_s,period_s,0);
   //
   return(slope);
}
//
double call_hensa(int start_h,int period_h)
{
   double koubai_M=kaiki_sen(Price_01,start_h,period_h,0);
   double seppen_M=kaiki_sen(Price_01,start_h,period_h,1);
   //
   double ooM=0;
   double sigma=0;
   for(int pM=start_h;pM<=(start_h+period_h);pM++)
   {
      Trend_[pM]=koubai_M*ooM+seppen_M;
      ooM=ooM+1;
      //
      double div_=(Price_01[pM]-Trend_[pM])*(Price_01[pM]-Trend_[pM]);
      sigma=sigma+div_;
   }
   double div=sigma/100;//100 足あたりの値
   //Print 内容は、log ファイルにも記録されるので、バックテスト時に異常解析の邪魔になる
  // Print("勾配は＝　",koubai_M);
  // Print("回帰線からの偏差＝　",div);
  // Comment("勾配は＝ ",koubai_M,":回帰線からの偏差＝　",div);
```

```
//---
   return(div);
}
//----------------------------------------
//回帰直線の傾き、切辺を返す
double kaiki_sen(double& price_[],int start_,int period_,int what_)
{
   double X_av=0.0,X_i=0.0;
   double Y_av=0.0,Y_i=0.0;
   double i_D;
   //
   ArraySetAsSeries(price_,true);
   //まず、平均値を求める
   for(int i=start_;i<=(start_+period_);i++)
   {
      i_D=i*1.0;
      X_i=X_i+i_D;
      Y_i=Y_i+price_[i];
   }
      X_av=(X_i/(period_+1));
      //X_av=(start_+(start_+period_))/period_;
      Y_av=(Y_i/(period_+1));
   //加算
   double bunshi=0.0,bunbo=0.0;
   double koubai_=0.0,seppen_=0.0;
   double j_D;
   //
   for(int j=start_;j<=(start_+period_);j++)
   {
      j_D=j*1.0;
      bunshi=bunshi+(j_D-X_av)*(price_[j]-Y_av);
      bunbo=bunbo+(j_D-X_av)*(j_D-X_av);
   }
      koubai_=bunshi/bunbo;
      //////seppen_=Y_av-(koubai_*X_av);//良い値が出ない
      seppen_=price_[start_];
   //
   if(what_==0) return(koubai_);
   if(what_==1) return(seppen_);
   //
}
//+------------------------------------------------------------------+
// Wun32API を利用した一時停止 (ただし、ビジュアル・モード)
void myPause(string short_s,double s_data,string short_m,double m_data)
{

   if(IsTesting() && IsVisualMode() && IsDllsAllowed())
   {
      //ＯＫコード；その１
      //keybd_event(19,0,0,0);
      //Sleep(10);
      //keybd_event(19,0,2,0);
      //Print(short_s,s_data,short_m,m_data);
      //Print("Ask= ",Ask," : Bid= ",Bid);
      //Comment(short_s,s_data,short_m,m_data,"¥n","Ask= ",Ask," : Bid= ",Bid);

      //ＯＫコード；その２
      keybd_event(0x13,0,0,0);
```

```
        Sleep(10);
        keybd_event(0x13,0,2,0);
        //Print(short_s,s_data,short_m,m_data);
        //Print("Ask= ",Ask," : Bid= ",Bid);
        Comment(short_s,s_data,short_m,m_data,"¥n","Ask= ",Ask," : Bid= ",Bid);

        //ＯＫコード；その３
        //int main=GetForegroundWindow();
        //PostMessageA(main,WM_COMMAND,0x57a,0);
        //Print(short_s,s_data,short_m,m_data);
        //Print("Ask= ",Ask," : Bid= ",Bid);
        //Comment(short_s,s_data,short_m,m_data,"¥n","Ask= ",Ask," : Bid= ",Bid);
    }

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

＜特記＞

・「OK コード；その１」、「OK コード；その２」または「OK コード；その３」の何れかで、同一の機能（一時ストップ）が可能です。
使用するコード部分の「／／；コメント・アウト」を外してください。

５．仮想キーコード一覧

値（16 進数）	意　味
01	マウス左ボタン
02	マウス右ボタン
03	[Cancel]
04	マウス中央ボタン
05	**Windows 2000/XP：** マウス X1 ボタン
06	**Windows 2000/XP：** マウス X2 ボタン
07	未定義
08	[Back space]
09	[Tab]
0A ～ 0B	予約
0C	[Clear]
0D	[Enter]
0E ～ 0F	未定義
10	[Shift]
11	[Ctrl]
12	[Alt]
13	[Pause]
14	[Caps Lock]
15	IME カナモード
15	IME ハングルモード
16	未定義
17	IME Junja モード
18	IME ファイナル モード
19	IME Hanja モード
19	IME 漢字モード
1A	未定義
1B	[Esc]
1C	IME 変換
1D	IME 無変換
1E	IME 使用可能
1F	IME モード変更

20	スペースキー
21	[Page Up]
22	[Page Down]
23	[End]
24	[Home]
25	[←]
26	[↑]
27	[→]
28	[↓]
29	[Select]
2A	[Print]
2B	[Execute]
2C	[Print Screen]
2D	[Insert]
2E	[Delete]
2F	[Help]
30 ～ 39	[0] ～ [9] (ASCII コード '0' ～ '9' と同じ)
3A ～ 40	未定義
41 ～ 5A	[A] ～ [Z] (ASCII コード 'A' ～ 'Z' と同じ)
5B	左の Windows キー
5C	右の Windows キー
5D	アプリケーションキー
5E	予約
5F	コンピュータスリープキー
60 ～ 69	テンキーの [0] ～ [9]
6A	テンキーの [*]
6B	テンキーの [+]
6C	テンキーの [Enter]
6D	テンキーの [-]
6E	テンキーの [.]
6F	テンキーの [/]
70	[F1]
71	[F2]

72	[F3]
73	[F4]
74	[F5]
75	[F6]
76	[F7]
77	[F8]
78	[F9]
79	[F10]
7A	[F11]
7B	[F12]
7C	[F13]
7D	[F14]
7E	[F15]
7F	[F16]
80	[F17]
81	[F18]
82	[F19]
83	[F20]
84	[F21]
85	[F22]
86	[F23]
87	[F24]
88 ～ 8F	割り当てなし
90	[Num Lock]
91	[Scroll Lock]
92 ～ 96	OEM 固有コード
97 ～ 9F	割り当てなし
A0	左の [Shift]
A1	右の [Shift]
A2	左の [Ctrl]
A3	右の [Ctrl]
A4	左の [Alt]
A5	右の [Alt]

A6	**Windows 2000/XP：** ブラウザの「戻る」キー
A7	**Windows 2000/XP：** ブラウザの「次へ」キー
A8	**Windows 2000/XP：** ブラウザの「更新」キー
A9	**Windows 2000/XP：** ブラウザの「中止」キー
AA	**Windows 2000/XP：** ブラウザの「検索」キー
AB	**Windows 2000/XP：** ブラウザの「お気に入り」キー
AC	**Windows 2000/XP：** ブラウザの「ホーム」キー
AD	**Windows 2000/XP：** ボリュームのミュートキー
AE	**Windows 2000/XP：** ボリュームダウンキー
AF	**Windows 2000/XP：** ボリュームアップキー
B0	**Windows 2000/XP：** 「次のトラック」キー
B1	**Windows 2000/XP：** 「前のトラック」キー
B2	**Windows 2000/XP：** 「メディア停止」キー
B3	**Windows 2000/XP：** 「メディア Start / Stop 」キー
B4	**Windows 2000/XP：** 「メール開始」キー
B5	**Windows 2000/XP：** 「メディア選択」キー
B6	**Windows 2000/XP：** 「アプリケーション 1 起動」キー
B7	**Windows 2000/XP：** 「アプリケーション 2 起動」キー
B8 ～ B9	予約
BA	さまざまな文字のために使用できます。 **Windows 2000/XP：** U.S. 標準キーボードでは [:;]
BB	**Windows 2000/XP：** [+]
BC	**Windows 2000/XP：** [,]
BD	**Windows 2000/XP：** [-]
BE	**Windows 2000/XP：** [.]
BF	さまざまな文字のために使用できます。 **Windows 2000/XP：** U.S. 標準キーボードでは [/?]
C0	さまざまな文字のために使用できます。 **Windows 2000/XP：** U.S. 標準キーボードでは [`~]
C1 ～ D7	予約
D8 ～ DA	割り当てなし
DB	さまざまな文字のために使用できます。 **Windows 2000/XP：** U.S. 標準キーボードでは [[{]
DC	さまざまな文字のために使用できます。 **Windows 2000/XP：** U.S. 標準キーボードでは [¥|]

DD	さまざまな文字のために使用できます。 **Windows 2000/XP：** U.S. 標準キーボードでは [ ]}]
DE	さまざまな文字のために使用できます。 **Windows 2000/XP：** U.S. 標準キーボードでは [’″]
DF	さまざまな文字のために使用できます。
E0	予約
E1	OEM 固有コード
E2	**Windows 2000/XP：** RT 102-key キーボードの角カッコまたはバックスラッシュ
E3 ～ E4	OEM 固有コード
E5	**Windows 95/98/Me/NT 4.0/2000/XP：** IME Process
E6	OEM 固有コード
E7	**Windows 2000/XP：** Unicode 文字がキーストロークであるかのように通すために使用されます。
E8	割り当てなし
E9 ～ F5	OEM 固有コード
F6	Attn
F7	CrSel
F8	ExSel
F9	Erase EOF
FA	Play
FB	Zoom
FC	予約
FD	PA1
FE	Clear

以　上