○「　ＭＱＬ４；基礎の確認（その３）　」

・アメンボです、
　しばらくぶりにノウハウ集（基礎の確認）への追加です。
（既に知れ渡っている内容でしたら、すいません。アメンボはＭＱＬ新参者ですゆえ）

・本ノウハウは、「Bollin_EA_08.mq4」開発過程の試行錯誤の中で理解・確認した内容です。
つくづく感じるのは、ＭＱＬはＣ言語と同じで奥が深い、その割りに高度な解説書（紙版）が少ない。
「ＷＥＢ上の資料って、ぶつ切り状態か、はたまた巨大すぎて、意外と読みづらい」
と、感じるのはアメンボだけか？

・機能確認用のサンプルコード類もダウンロード用に掲載しておきました。

目次：

１．ファイル・アクセス可能フォルダについて

・今回は、実用的で且つ基本的なことだけをまとめることにしました。
（詳細にまで立ち入ると、奥が深くてアメンボの手には負えません）

（１）「ファイル・オープン関数」と「アクセス可能フォルダー」の関係

ファイル・オープン関数	アクセス可能な 「¥MetaTrader 4 at ﾌﾞﾛｰｶｰ名」 以下のフォルダー・パス名
FileOpen()	¥experts¥files
	¥tester¥files
FileOpenHistory()	¥history¥ﾌﾞﾛｰｶｰ server_name

・要約すれば、「FileOpen()」は実行チャート（モード）により決まる、
「¥experts¥files」か「¥tester¥files」フォルダ内のどちらかの、ファイルを
アクセスすることになります。

（２）「ＥＡ実行チャート（モード)」と「アクセス・フォルダ」の関係

・今回は最も実用的な「ＥＡ」実行によりアクセスする場合だけに限定します。

稼動	オープン関数	ﾘｱﾙ（ﾃﾞﾓ）ﾁｬｰﾄ	ﾊﾞｯｸﾃｽﾄ（ﾁｬｰﾄ）ﾓｰﾄﾞ	ｵﾌﾗｲﾝ（offline）ﾁｬｰﾄ
ＥＡ	FileOpen()	¥experts¥files	¥tester¥files　※２	¥experts¥files　※１
	FileOpenHistory()	¥history ¥ﾌﾞﾛｰｶｰ server_name	¥history ¥ﾌﾞﾛｰｶｰ server_name	¥history　※１ ¥ﾌﾞﾛｰｶｰ server_name

※１；init()は動きますが、start()は動かない。（tick が来ないから当然か！）
もっとも、アメンボのように「ｵﾌﾗｲﾝ・ﾁｬｰﾄ」で、ＥＡを動かしてみようなんて、
へそ曲がりはいないかもしれませんが。

※２；理由は不明ですが、「ﾊﾞｯｸﾃｽﾄ」では「PlaySound()」が動作せず。
（確かに動作しないほうが良いけど、理由が気になる）

（３）動作結果のファイル出力例

・ファイルを書き込みモードでオープンして、何かを書き込むと、
「experts¥files」フォルダの下に、指定した「write_test_EA.txt」ファイルが、
①存在していない場合は、新たにファイルが生成されて下記のように書込みがされる。
②すでに存在していた場合は、一旦内容がクリアされてから新たに書込みがされる。

（４）動作確認用ＥＡコード

```
//+------------------------------------------------------------------+
//|                                                write_test_EA1.mq4 |
//|                                                          amenbo   |
//|                                                                   |
//+------------------------------------------------------------------+
//
int count;
int outFile;
//
int init()
  {
     //「experts¥files」用のＯＫ品
     outFile=FileOpen("write_test_EA.txt",FILE_WRITE|FILE_CSV);
     //「history¥server_name」用ＯＫ品
     //outFile=FileOpenHistory("write_test_EA.txt",FILE_WRITE|FILE_CSV);
     //string str0="init()部は通過した";
     //FileWrite(outFile,str0);
     count=1;
     //
   return(0);
  }
//+------------------------------------------------------------------+
//| expert deinitialization function                                 |
//+------------------------------------------------------------------+
int deinit()
  {
   FileFlush(outFile);
   FileClose(outFile);
   return(0);
  }
//+------------------------------------------------------------------+
//| expert start function                                            |
//+------------------------------------------------------------------+
int start()
  {
   if(count==1)
   {
      string str1="01234567890123456789";
      string str2="abcdefghijklmnopqrstuvwxyz";
      string str3="これはＥＡです";

      FileWrite(outFile,str1);
      FileWrite(outFile,str2);
      FileWrite(outFile,str3);

      FileFlush(outFile);
      FileClose(outFile);
      //PlaySound("news.wav");
      count=count+1;
      PlaySound("news.wav");
    }//if(count==1)
      //PlaySound("news.wav");
      return(0);
  }
//+------------------------------------------------------------------+
```

２．エクセル・ファイルにデータを書き込む

・データ間のデリミタを"¥t"（タブ）とすれば、エクセルファイルに出力することも可能です。
・以下にチェック用のサンプル例を示します。

（１）動作確認用コード（スクリプト）

```
//+------------------------------------------------------------------+
//|                                                   excel_write.mq4 |
//|                                                           amenbo  |
//|                                                                   |
//+------------------------------------------------------------------+
int start()
  {
      double data[6]={2.1,3.1,5.5,12.0,80.222,8823.11,7.7,8.8,9.9,0.0};
      int outFile=FileOpen("write_test.xls",FILE_WRITE|FILE_CSV,"¥t");

      for(int i=0;i<=9;i++)
      {
         FileWrite(outFile,i,data[i]);
      }

      FileFlush(outFile);
      FileClose(outFile);
      PlaySound("news.wav");
      return(0);
  }
```

（２）結果；

・「experts¥files」フォルダの下に「write_test.xls」ファイルが、
①存在していない場合は、新たにファイルが生成されて下記のように書込みがされる。
②すでに存在していた場合は、一旦内容がクリアされてから新たに書込みがされる。

	A	B	C	D
1	0	2.1		
2	1	3.1		
3	2	5.5		
4	3	12		
5	4	80.222		
6	5	8823.11		
7	6	7.7		
8	7	8.8		
9	8	9.9		
10	9	0		
11				
12				
13				

３．カウント・ダウン方式への対応ＥＡ

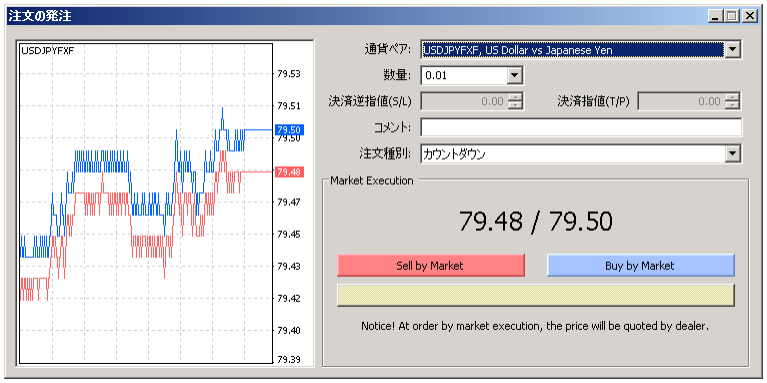
・ＥＡが「カウントダウン方式」に対応するための注意点を記述することにしました。
（理由は、最近はこの方式を採用する「ブローカー」が増えてきたためです、既に殆ど？？）

（１）ＥＡを作成する上での問題

・ブローカーの注文種別が「カウントダウン方式」の場合は、
ＥＡ中で、以下のタイプのオーダーを発行すると「エラー」になることはご存知と思います。

```
OrderSend(Symbol(),OP_BUY,1,Ask,3,Ask-25*Point,Ask+25*Point,"My_order",20120608,0,Green);
```

「スリップページ、損切りレベル、利食いレベル」を初めから注文に組込んでおくタイプです。
※カウントダウン方式では、この「スリップページ、損切りレベル、利食いレベル」を
受け付けてくれません。（と、言うか、概念が存在しないと言った方が正解）



・ＥＡで注文を発行しても、「１３０；INVALID_STOPS」エラーを返してきて動作しません。
例；　2012.06.08 23:28:08　　check_count_down USDJPYFXF,M5: order error : 130

（２）一つの解決方法

・下記のステップで注文を出します
①第１ステップ；
「スリップページ＝０、損切りレベル＝０、利食いレベル＝０」で注文を発行

```
ticket=OrderSend(Symbol(),OP_BUY,1,Ask,0,0,0,"My_order",20120608,0,Green);
```

②第２ステップ；
ティケットＮｏで上記の注文を選択

```
OrderSelect(ticket,SELECT_BY_TICKET);
```

③第３ステップ；
注文を修正する

```
OrderModify(OrderTicket(),OrderOpenPrice(),stop_loss,take_profit,0,Green);
```

※「スリップページ」の設定は出来ませんが、まあ良しとするしかないか。
※ところで北米のフォーラムでカウントダウン方式のことを検索しても、
「Straitght Through Processing bridge with Bank」　と表現しているようで、
「count dowm」で検索してもさっぱり要領を得ませんでした！、何故？

（３）実際に試してみる

・アメンボは、正直言いますと、このタイプの注文方法(ストップロス等を事前設定する)は殆ど使ったことが無いので、原理を確認するだけのコードを作って確認してみました。

（コードは、次頁以降のサンプル内容を参照ください）

手順；　①「check_count_down」ＥＡをセットする

②注文が発行されか、エラーがでたらＥＡを外す

③注文履歴を確認する（エラー発生は除く）

結果；

（－１）「カウント・ダウン方式のオーダーＥＡ」コードの結果・・注文ＯＫ

○○○○.com Japan

Account: 179420　Name: amenbo　Currency: JPY　2012 June 8, 23:18

Closed Transactions:

Ticket	Open Time	Type	Size	Item	Price	S / L	T / P	Close Time	Price	Commission	Taxes	Swap	Profit
10216601	2012.06.08 23:14	buy	1.00	usdjpyfxf	79.45	79.19	79.69	2012.06.08 23:17	79.46	0	0	0	1 000
										0	0	0	1 000
												Closed P/L:	**1 000**

Open Trades:

Ticket	Open Time	Type	Size	Item	Price	S / L	T / P		Price	Commission	Taxes	Swap	Profit
No transactions													
										0	0	0	0
												Floating P/L:	**0**

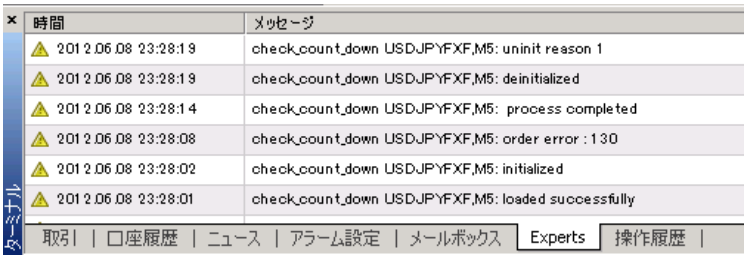
Working Orders:

Ticket	Open Time	Type	Size	Item	Price	S / L	T / P	Market Price
No transactions								

Summary:

Deposit/Withdrawal:	**0**	**Credit Facility:**	**0**		
Closed Trade P/L:	**1 000**	**Floating P/L:**	**0**	**Margin:**	**0**
Balance:	**485 000**	**Equity:**	**485 000**	**Free Margin:**	**485 000**

（－２）「従来方式のオーダーＥＡ」コードの結果・・注文ＮＧ



※エラー「１３０」で動作しません。

（４）動作確認用ＥＡコード　・・１つのコードで「コメントアウト」部位を変えて使用

＜カウント・ダウン方式のオーダーＥＡ＞

```
//+------------------------------------------------------------------+
//|                                            check_count_down.mq4 |
//|                                                          amenbo  |
//|                                                                  |
//+------------------------------------------------------------------+
//
double stop_loss;
double take_profit;
int count;
//
int init()
  {
   count=0;
   //Print("Point=",Point," : Digits=",Digits);
   return(0);
  }
//----------------
int deinit()
  {
   return(0);
  }
//----------------
int start()
  {

   if(count==0){
//int ticket=OrderSend(Symbol(),OP_BUY,1,Ask,3,Ask-25*Point,Ask+25*Point,"My_order",20120608,0,Green);
      int ticket=OrderSend(Symbol(),OP_BUY,1,Ask,0,0,0,"My_order",20120608,0,Green);//(A)セット
      //
      if(ticket>0){
         stop_loss=Ask-25*Point;
         take_profit=Ask+25*Point;
         OrderSelect(ticket,SELECT_BY_TICKET);//(A)セット
         OrderModify(OrderTicket(),OrderOpenPrice(),stop_loss,take_profit,0,Green);//(A)セット
         Print("order OK ");
      }else{
         Print("order error : ",GetLastError());
      }
      //
      count=count+1;
      PlaySound("email.wav");
   }else{
      PlaySound("alert.wav");
      Print(" process completed ");
   }

   //Print("Point=",Point," : Digits=",Digits);
   //PlaySound("alert.wav");
   //
   return(0);
  }
//==================================================================
//一回だけオーダーを出す
//「カウントダウン方式」と「従来方式」で、注文が通るかを確認する。
```

＜従来方式のオーダーＥＡ＞・・「１３０；INVALID_STOPS」エラーとなる

```
//+------------------------------------------------------------------+
//|                                            check_count_down.mq4 |
//|                                                          amenbo  |
//|                                                                  |
//+------------------------------------------------------------------+
//
double stop_loss;
double take_profit;
int count;
//
int init()
  {
   count=0;
   //Print("Point=",Point," : Digits=",Digits);
   return(0);
  }
//-----------------
int deinit()
  {
   return(0);
  }
//-----------------
int start()
  {

   if(count==0){
   int ticket=OrderSend(Symbol(),OP_BUY,1,Ask,3,Ask-25*Point,Ask+25*Point,"My_order",20120608,0,Green);
      //int ticket=OrderSend(Symbol(),OP_BUY,1,Ask,0,0,0,"My_order",20120608,0,Green);//(A)セット
      //
      if(ticket>0){
         //stop_loss=Ask-25*Point;
         //take_profit=Ask+25*Point;
         //OrderSelect(ticket,SELECT_BY_TICKET);//(A)セット
         //OrderModify(OrderTicket(),OrderOpenPrice(),stop_loss,take_profit,0,Green);//(A)セット
         Print("order OK ");
      }else{
         Print("order error : ",GetLastError());
      }
      //
      count=count+1;
      PlaySound("email.wav");
   }else{
      PlaySound("alert.wav");
      Print(" process completed ");
   }

   //Print("Point=",Point," : Digits=",Digits);
   //PlaySound("alert.wav");
   //
   return(0);
  }
//==================================================================
//一回だけオーダーを出す
//「カウントダウン方式」と「従来方式」で、注文が通るかを確認する。
```

以　上