○「スクリプトの活用（その１）;外部アプリ起動とバックテスト起動」

・アメンボです、

EA の最適化やバックテストを自動で行うことを以前から考えています、

（理由その１）; 手作業で最適化を行うと、とにかく時間が掛かって仕方ないし、
時々パソコンをチェックするのがめんどくさい。

（理由その２）; ＥＡの実力値評価を、「extern 値」以外の外部要因を変化させて行いたい
ことがあるのですが、これも全て手作業で行うのはとてもカッタルイ。

・自動化と言えば、

小生なら「エクセルのマクロ」や「ＶＢスクリプト」を直ぐに思い浮かべるのですが、
しばらく使っていないと、すぐにコマンドや文法を忘れているし、
MQL4 と両方使っていくのも頭が混乱するし！！（歳のせい？）
と、言うことで、しばらくは手に付かない状況でした。

・そこで、ハット思いだしたのが、MQL4 のスクリプトでした。(特に Win32API による機能拡張)
これ自体、Ｃ言語ライクな立派なスクリプトですので活用しない手はなく、さっそく
資料調査と実機確認を始めましたので、途中経過の報告をすることとしました。

＜同時掲載資料＞・・ダウンロード用

・アメンボが動作確認に使用したスクリプトと参照ファイル一式;
「**shell_amenbo_set_01.zip**」を解凍して試してみてください

※「バックテストを起動する」を試すには、**MT4 を２セット**、ダウンロードしておくことが必要です。

※本稿は、「MT4 ; bild509」にて確認済みのものです。（しかし、509 は文字化け部分が多すぎる！）

目次：

１.「Win32API による機能拡張」の復習

・これまでも Win32API の使用例を何回か報告済みですので、その内容の大雑把な振り返りを行い、また今回使用する関数を以下に簡単にまとめます。

（１）MQL4 で「Win32API」を利用する際の書式

まず、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「shell32.dll」に属する「ShellExecuteA(....)」を使いたいのであれば、
コードの頭部分で、インポート（兼、プロトタイプ宣言）しておくと、続く MQL4 コード内で、
ShellExecuteA(....)が使えるようになります。

```
#import "shell32.dll"
    int ShellExecuteA(int hWnd,string lpVerb,string lpFile,string lpParameters,
                                                    string lpDirectory,int nCmdShow);
#import

int start()
  {
        ・・・・・・・・
       ShellExecuteA(NULL,.........,5);
        ・・・・・・・・
  }
```

（２）インポートする DLL 別の振り返りとまとめ

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・オープンなど
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！
＜今回使用＞ shell32.dll	シェル・ライブラリを提供、 アプリや、Web ページの起動	**ShellExecuteA(....) ・任意アプリソフトの起動**

※注意；アメンボは Win32API を初めて使おうとしたときに、上記の DLL を探してしまいました！
が、一個に纏まった DLL としての実体はなく、Windows システムがサポートしているので、
ただ、
「 #import"・・・" ・・ #import 」と書けば良いのだ、
と気が付くのに随分時間がかかりましたので、諸兄においても「ご注意」。

２．ShellExecuteA(・・)関数の使い方

※「理解できていない部分、実働での確認をしていない内容」が多々あるのですが、
諸兄による更なる調査が可能になるように記載します。

（１）定義

```
int ShellExecuteA(
    int hWnd,               //親ウィンドウのハンドル
    string lpVerb,          //ファイルの操作
    string lpFile,          //ファイル又はアプリケーションのパス
    string lpParameters,    //アプリケーションに渡す引数
    string lpDirectory,     //デフォルトのディレクトリ
    int nCmdShow            //アプリケーションが実行されたときのウィンドウの状態
    );
```

（２）パラメータ

・int hWnd；　　親ウインドウのハンドル、「0」でも「NULL」でも起動する。

・string lpVerb；　起動操作（ファイル操作）をする文字列。

操　作	説　明	備　考
edit	エディタを開く； *lpFile* で指定したファイルが文書ファイルではない場合、 この関数は失敗します。	未確認
explore	*lpFile* パラメータで指定したフォルダを選択して、 エクスプローラを起動する。	未確認
open	*lpFile* パラメータで指定したファイルを開く； *lpFile* パラメータで、 文書ファイルまたは実行可能ファイルを指定できる。 1 つのフォルダを指定することもできる。	アメンボ確認済
print	*lpFile* パラメータで指定したファイルを印刷する； *lpFile* パラメータで文書以外のファイルを指定すると、 この関数は失敗する。	未確認
properties	ファイルまたはフォルダのプロパティを表示する。	未確認

・string lpFile；　操作対象のファイル名またはフォルダ名へのパスを指定する。

・string lpParameters；

lpFile パラメータで実行ファイルを指定されたときに、
アプリケーションに引き渡すコマンドライン・パラメータを指定する。
それ以外は NULL を指定する。(MS-DOS プロンプトで渡す引数と同じ意味を持つ)

・string lpDirectory；

作業ディレクトリが格納されている文字列を指定する、
不要なときは「NULL」を指定することができる。

・int nCmdShow；

アプリケーションの表示方法を指示するフラグを指定する、

フラグの扱い方は、そのアプリケーションに依存する。

※次の値のいずれかを指定する；

指定整数 MQL4 で使用	C 言語での定義	説　明
0	SW_HIDE	ウィンドウを非表示にし、 他のウィンドウをアクティブにする。
1	SW_SHOWNORMAL	ウィンドウをアクティブにして、表示する。 ウィンドウが最小化または最大化されているときは、 位置とサイズを元へ戻す。 アプリケーションは、ウィンドウを最初に表示するときに このフラグを指定する必要あり。
2	SW_SHOWMINIMIZED	ウィンドウをアクティブにして、最小化する。
3	SW_SHOWMAXIMIZED	ウィンドウをアクティブにして、最大化する。
4	SW_SHOWNOACTIVATE	ウィンドウを直前の位置とサイズで表示する。 アクティブなウィンドウはアクティブな状態を維持する。
5	SW_SHOW	ウィンドウをアクティブにして、 現在の位置とサイズで表示する。
6	SW_MINIMIZE	指定されたウィンドウを最小化し、 奥行き方向で、指定されたウィンドウのすぐ奥にある ウィンドウをアクティブにする。
7	SW_SHOWMINNOACTIVE	ウィンドウを最小化する。 アクティブなウィンドウは、アクティブな状態を維持する。
8	SW_SHOWNA	ウィンドウを現在の状態で表示する。 アクティブなウィンドウはアクティブな状態を維持する。
9	SW_RESTORE	ウィンドウをアクティブにし、表示する。 指定されたウィンドウが最小化または最大化されていた場合、 元の位置とサイズに戻る。 最小化されたウィンドウを元のサイズへ戻す場合、 アプリケーションはこのフラグを指定する必要あり。
10	SW_SHOWDEFAULT	アプリケーションを起動したプログラムが 関数に渡す構造体の wShowWindow メンバで指定された SW_ フラグに基づいて、表示状態を設定する。 アプリケーションは、このフラグを指定して関数を呼び出し、 自らのメインウィンドウの初期の表示状態を設定する必要あり。
11	SW_MAXIMIZE	指定されたウィンドウを最大化する。

※MQL4 中でＣ言語と似た記述を行うには、「　#define SW_HIDE　 0　」などと定義すると良い。

※アメンボは、上表の内「0」と「5」しか動作確認していません。

（他の値の説明に間違いあれば、ご容赦）

（3）リターン値（返し値）

・「int r=ShellExecuteA(...) ;」として受けた場合、

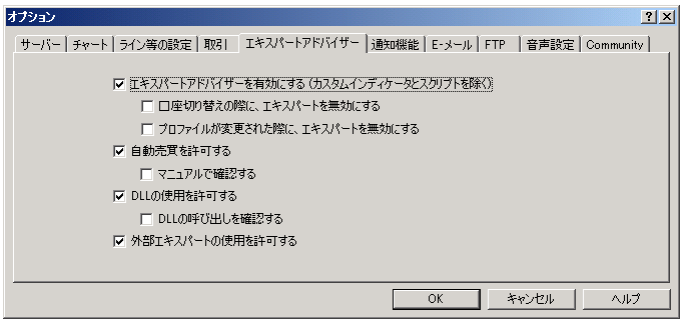
r≦32　であると、失敗

r≧33　ならば、成功

らしいのですが、それ以上は良く判らず未調査です。（手を抜いた）

３．ShellExecuteA(・・)関数使用例

※以下の使用例を動作させる前に、下記の設定（DLL の使用許可）をしておいてください。



※以下の MQL4 コードは、すべて「スクリプト」として書いたものです。

※注意；　「フォルダ、ファイル」へのパスを指定（記述）する場合、
日本語（２バイトコード）が入ると、動作しません。
１バイトコードのみを使うように注意が必要です。

（１）例１；ワードパッドでテキスト・ファイルを開く

－１．フォルダとファイルの配置

「C:¥amenbo」フォルダ内のファイル配置；

「Text3.txt」ファイル内容;

```
yomerunoka ？
読めるのか？
既定のディレクトリとは？？
```

※「TeraPad.exe」はフリーウエア、「wordpad.exe」は Windows に附属しています。
本稿では「shell_amenbo_set_01.zip」解凍後の amenbo フォルダーに wordpad.exe を入れましたが、TeraPad.exe は入れていません。
TeraPad.exe で確認したい場合は、諸兄にてネット上から入手してください。
（あるいは、他の適当な「テキスト・エディタ」で試してみてください。）

－２．MQL4 コード

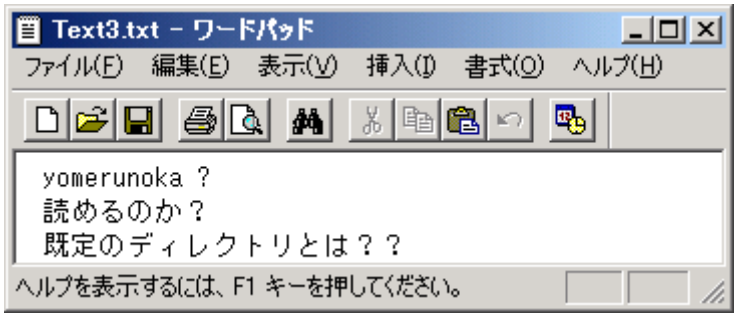
```
//+------------------------------------------------------------------+
//|                                                shell_amenbo_01.mq4 |
//|                                                            amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

//=======Win32API=========
// Win32API 使用宣言
//#include <WinUser32.mqh>
//#import "user32.dll"
#import "shell32.dll"
   //int ShellExecuteA(int hWnd,int lpVerb,string lpFile,int lpParameters,
                                    int lpDirectory,int nCmdShow);//①ＮＧ
   int ShellExecuteA(int hWnd,string lpVerb,string lpFile,string lpParameters,
                                    string lpDirectory,int nCmdShow);//②ＯＫ
#import
//========================

int start()
  {
   //ShellExecuteA(0,0,"TeraPad.exe",0,0,5);//①
   //ShellExecuteA(0,"open","C:¥¥TeraPad.exe","","",5);//②ＯＫ
   //ShellExecuteA(0,"open","C:¥¥TeraPad.exe","C:¥¥Text2.txt","",5);//②ＯＫ
   //ShellExecuteA(NULL,"open","C:¥¥TeraPad.exe","C:¥¥Text2.txt",NULL,5);//②ＯＫ
   ShellExecuteA(NULL,"open","wordpad.exe","Text3.txt","C:¥¥amenbo",5);
   //
   PlaySound("alert2.wav");
   return(0);
  }
//+------------------------------------------------------------------+
```

－３．実行結果

※スクリプト「shell_amenbo_01」をダブル・クリックすると、下記を表示します。



※ポイント；

①lpDirectory として「""」や「NULL」を指定すると、アプリやファイルは絶対パス指定が必要。

②lpDirectory として「"C:¥¥amenbo"」記述すると、これがデフォルト・ディレクトリーとなり、中に入っているアプリとファイルは相対パス指定でＯＫとなります。

※本件の MQL4 コードには、「うまく行かなかったコード」や「トライしてみた途中のコードも消さずに残しておきました。（何かの参考になるかと思い）

（２）例２；Web ページを開く

－１．MQL4 コード

```
//+------------------------------------------------------------------+
//|                                                shell_amenbo_02.mq4 |
//|                                                            amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

//=======Win32API=========
// Win32API 使用宣言
#import "shell32.dll"
   int ShellExecuteA(int hWnd,string lpVerb,string lpFile,string lpParameters,
                                                   string lpDirectory,int nCmdShow);
#import
//========================

int start()
  {
   ShellExecuteA(NULL,"open","http://www.benri.ne.jp/",NULL,NULL,5);//ＯＫ
   //
   PlaySound("alert2.wav");
   //
   return(0);
  }
//+------------------------------------------------------------------+
```

－２．実行結果・・お気に入りの部分は削除しました。（ご容赦、見られると恥ずかしい？？）

※スクリプト「shell_amenbo_02」をダブル・クリックすると、下記を表示します。

※「www.benri.ne.jp」は、たまたま amenbo が良く利用していただけ。

（３）例３；連続してアプリを起動する

※「wordpad.exe」と「Text3.txt」のフォルダ配置は、「shell_amenbo_01」の場合と同じです。
「calc.exe」は、実行ファイルまでの絶対パスで示しています。

－１．MQL4 コード

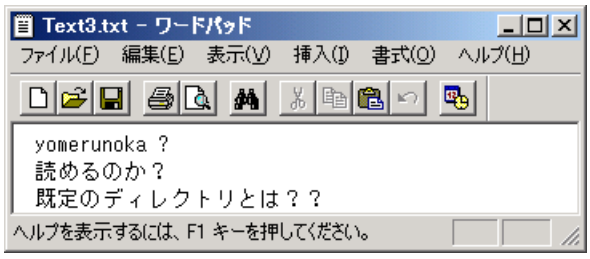
```
//+------------------------------------------------------------------+
//|                                                shell_amenbo_03.mq4 |
//|                                                            amenbo |
//|                                                  泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

#import "shell32.dll"
   int ShellExecuteA(int hWnd,string lpVerb,string lpFile,string lpParameters,
                                                    string lpDirectory,int nCmdShow);
#import
//=========================
int start()
  {
   ShellExecuteA(NULL,"open","wordpad.exe","Text3.txt","C:¥¥amenbo",5);//SH_SHOW
   //
   PlaySound("alert.wav");
   //
   ShellExecuteA(NULL,"open","c:¥¥windows¥¥system32¥¥calc.exe","","",5);//SH_SHOW
   //
   PlaySound("alert2.wav");
   return(0);
  }
```

－２．実行結果

※スクリプト「shell_amenbo_03」をダブル・クリックすると、下記の様に次々と起動します。

①ワードパッド（wordpad.exe）が起動して、ファイル内容（Text3.txt）を表示



②「"alert.wav"」音がした、と思ったら

③直ぐに、「calc.exe」で電卓が起動、と思ったら

④「alert2.wav」音がした。・・そして、ファイルと電卓は表示されたまま。

※後ほど「４．現状課題と次回検討事項」で触れますが、
「ShellExecuteA(...)」は次々とアプリを立ち上げてしまい、アプリの終了如何に係らず、
自身はその後にすぐ終了してしまう。

（４）例４；MT4 のバックテストを起動する

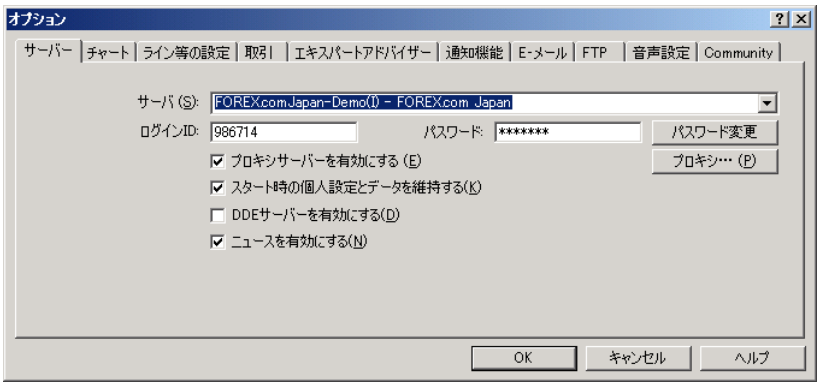
※スクリプトを使って、MT4 のバックテストを起動することを試みます。
結論を先に述べると、MT1 と MT2 の２セットの MT4 が在るとして、
◎MT1 でも MT2 でも、自身のスクリプトから自分自身のテスターを起動させることはＮＧでした。
◎一方、MT1 のスクリプトから、MT2 のテスターを起動させることはＯＫです。

－１．準備（MT1 と MT2）

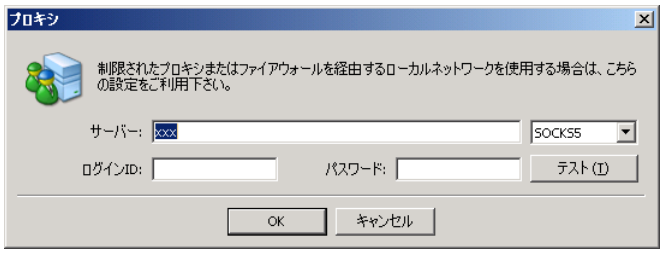
①先ず、MT4 を２セット（MT1 と MT2）、別ホルダーにダウンロードします。
MT1　→　MT4_main(bild509)
MT2　→　MT4_back_test(bild509)　とした。（本稿）

②MT2 側の処理１；
ここでは、**MT2 をバックテスト専用 MT4** と決め、下記の設定で OFF_LINE 状態にしておきます。

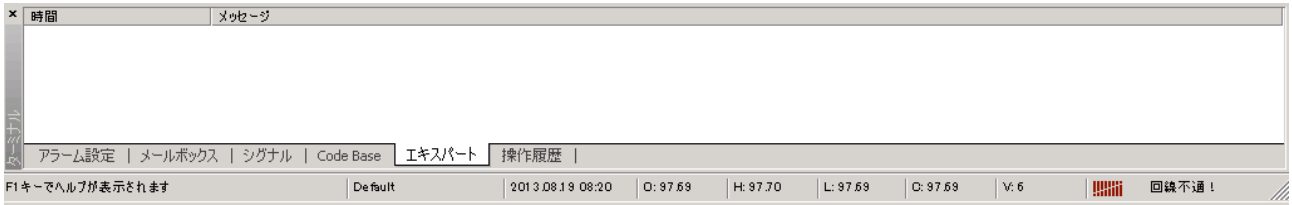
・［プロキシサーバーを有効にする］にチェック（レ）



・サーバーに在り得ない名前の「xxx」を入れた



・「回線不通」が右下に表示されればＯＫ

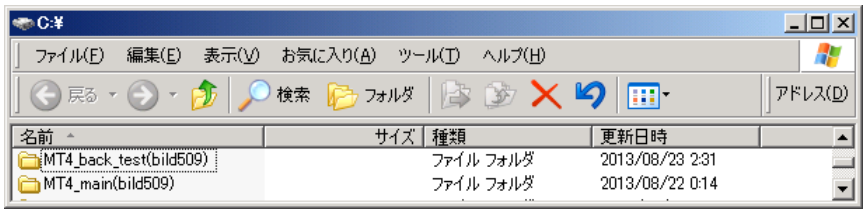


②MT2 側の処理２；

・「Strategy Tester」をクリックして、バックテストの［セッティング］画面を表示させてから、そのまま MT2 を終了しておきます。
（未だ、この処理は自動化出来ていません）

－２．フォルダと設定ファイルの配置

①本実験では「パス指定」が楽なので、下記の配置（C:¥　直下）としました。



②設定ファイルの配置・・何れも、MT2 側の「¥tester」フォルダ中に置く

「amnebo_text1.txt」・・バックテスト条件設定用（必須）

「empty01.set」・・バックテスト時の「extern 値」設定用（任意、無くてもＯＫ）

③設定ファイルの内容

「amenbo_test1.txt」ファイル；今回使用した内容（「;」部分はコメント・アウトされる）

```
; start strategy tester
TestExpert=Empty_EA_1
;TestExpertParameters=¥experts¥presets¥empty01.set;動かない
;「empty01.set」は¥tester に置いておくと、下記の設定で機能する
;TestExpertParameters=empty01.set
TestSymbol=USDJPYFXF
TestPeriod=M5
TestModel=0
;TestRecalculate=false
TestOptimization=false
;TestDateEnable=true
;TestFromDate=2013.03.22
;TestToDate=2013.05.15
TestReport=tester¥EmptyEA1Report
;TestReport=EmptyEA1Report;terminal.exe と同じ階層に作成されてしまう
;TestReplaceReport=true
TestReplaceReport=false
TestShutdownTerminal=false
```

「empty01.set」ファイル；

```
a=20
b=6.00000000
```

※ＥＡコード中に記述された「extern 値」とは異なる「値」を利用する場合に使う。
（従って、使わなくても動作する）

－３．ＥＡコード・バックテスト対象（ダミーＥＡ）、MT2 の「¥experts」フォルダ内配置（当然か）

```
//+------------------------------------------------------------------+
//|                                                  Empty_EA_1.mq4 |
//|                                                          amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

extern int a=10;
extern double b=3.0;
//
int init()
  {
   return(0);
  }
//
int deinit()
  {
   return(0);
  }
//-----------------------------------
int start()
  {
   for(int i=0;i<5;i++)
   {
      double c=b+a*1.2;
   }
   //
   Print("a=",a," : b=",b);
   return(0);
  }
//-----------------------------------
```

－４．MQL4 スクリプト・コード・・・バックテスト起動用

※これは MT1 の「¥experts¥scripts」に配置すること、間違えないで！

```
//+------------------------------------------------------------------+
//|                                            shell_amenbo_11.mq4 |
//|                                                          amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

//=======Win32API=========
#import "shell32.dll"
   int ShellExecuteA(int hWnd,string lpVerb,string lpFile,string lpParameters,string
lpDirectory,int nCmdShow);
#import
//========================

int start()
  {
   ShellExecuteA(NULL,"open","C:¥¥MT4_back_test(bild509)¥¥terminal.exe",
                     "C:¥¥MT4_back_test(bild509)¥¥tester¥¥amenbo_test1.txt","",5);
   //
   PlaySound("alert2.wav");
   return(0);
  }
```

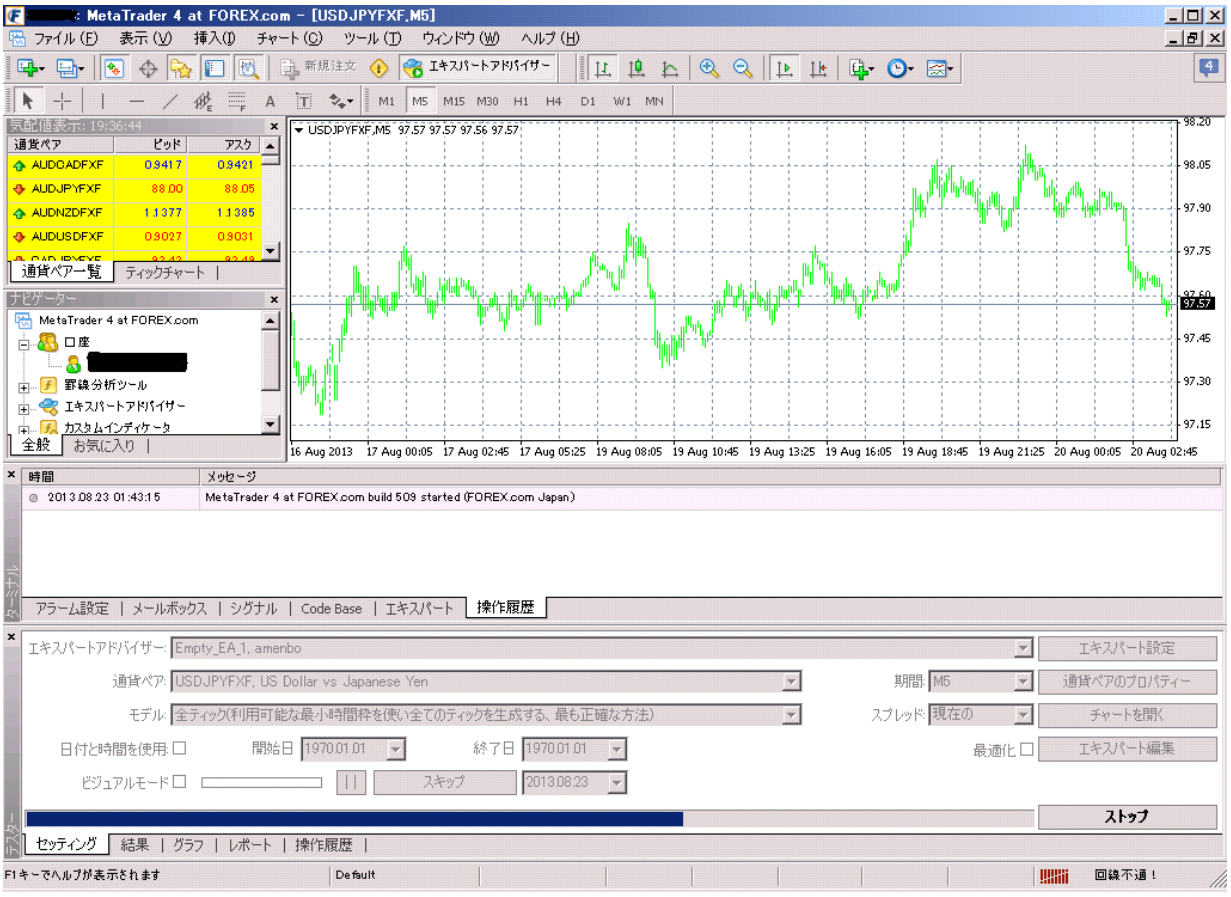
－５．実行結果

※ＭＴ１側で、スクリプト「shell_amenbo_11」をダブル・クリックすると、

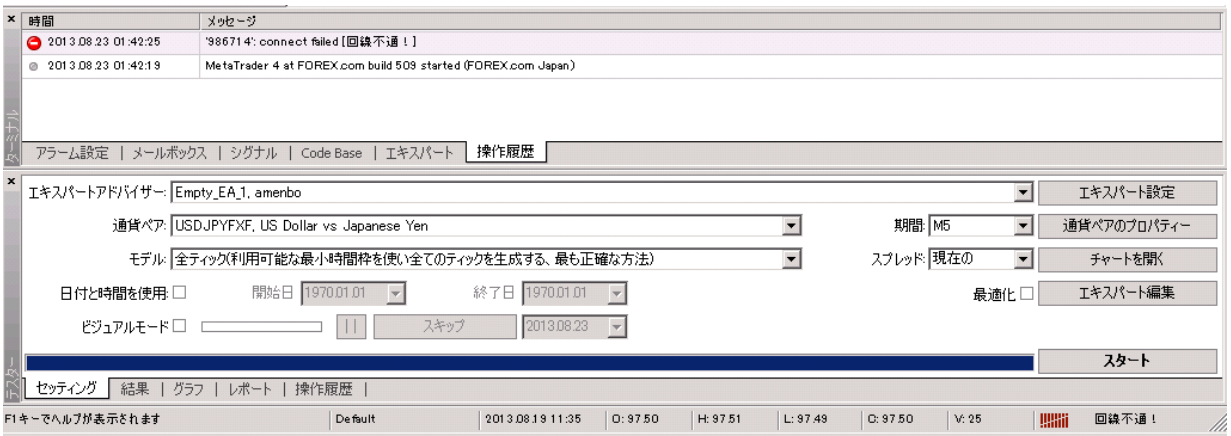
・下記の様に **MT2 が立上った後**、バックテストが実行される。(「プキュー」音もあり)

※「amenbo_test1.txt」内の下記の記述による効果

```
TestExpert=Empty_EA_1
TestSymbol=USDJPYFXF
TestPeriod=M5
TestModel=0
TestOptimization=false
・・・・
TestShutdownTerminal=false
```



↓

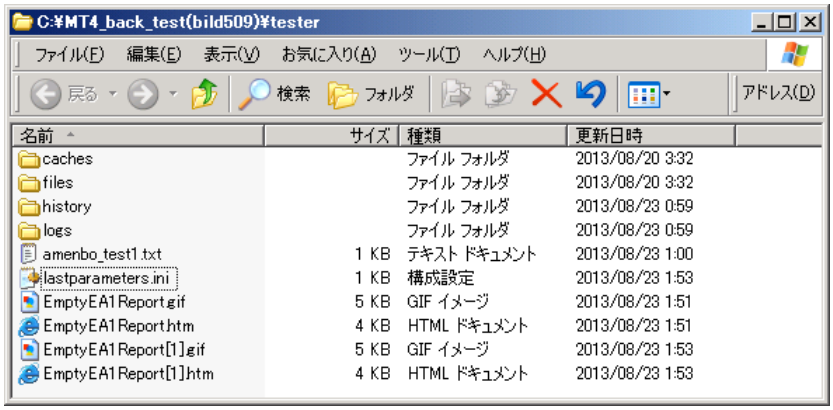


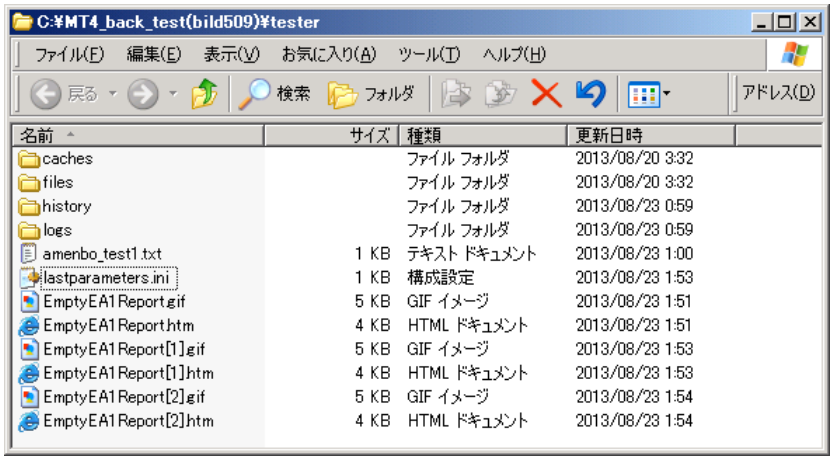
－６．「¥experts」下には、ＥＡのレポートファイルが生成される。

※「amenbo_test1.txt」内の下記の記述による効果

```
TestReport=tester¥EmptyEA1Report
TestReplaceReport=false
```

・・何度も実行すると、「EmptyEA1Report[*]」が増えていく





－７．捕捉；「TestExpertParameters」の機能

※「amenbo_test1.txt」内に、「TestExpertParameters」の指定は無くても動作する。
「TestExpertParameters=empty01.set」設定の有無で、何が異なるのか？

再確認；「empty01.set」ファイル

```
a=20
b=6.00000000
```

再確認；「Empty_EA_1.mq4」

```
//+------------------------------------------------------------------+
//|                                                 Empty_EA_1.mq4 |
//|                                                         amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

extern int a=10;
extern double b=3.0;
//
・・・・・・・・以下、略・・・・・・・・
```

¥tester¥logs フォルダ内の「日付.log」ファイルの内容を確認してみる；

①「;TestExpertParameters=empty01.set」とコメント・アウトした場合

```
01:54:16 2013.08.14 23:55  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
01:54:16 2013.08.14 23:55  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
01:54:16 2013.08.14 23:55  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
01:54:16 2013.08.14 23:55  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
01:54:16 2013.08.14 23:55  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
01:54:16 2013.08.14 23:56  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
01:54:16 2013.08.14 23:56  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
01:54:16 2013.08.14 23:56  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
01:54:16 2013.08.14 23:56  Empty_EA_1 USDJPYFXF,M5: a=10 : b=3
```

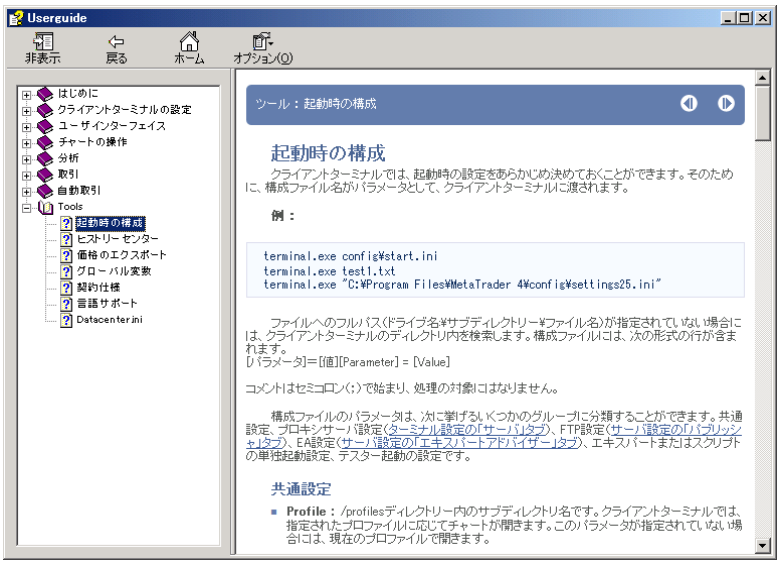
②「TestExpertParameters=empty01.set」と機能を活かした場合

```
02:15:19 2013.08.20 03:20  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
02:15:19 2013.08.20 03:20  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
02:15:19 2013.08.20 03:20  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
02:15:19 2013.08.20 03:21  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
02:15:19 2013.08.20 03:21  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
02:15:19 2013.08.20 03:21  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
02:15:19 2013.08.20 03:23  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
02:15:19 2013.08.20 03:24  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
02:15:19 2013.08.20 03:24  Empty_EA_1 USDJPYFXF,M5: a=20 : b=6
```

※つまり、「TestExpertParameters=empty01.set」と設定すると、
「Empty_EA_1.mq4」コード内の「extern 値；a、b」は、「empty01.set」内に記述される値に置き換わる。

－８．「起動時の構成」資料

※MT4（bild482）に付随している「Terminal_japanese.chm」から、
「冒頭部分」と「テスターの起動設定」のみを抜粋した。
！理由は、何と「bild509」には日本語版が付いて無かった、もので！（英語版はあるけど）



起動時の構成

クライアントターミナルでは、起動時の設定をあらかじめ決めておくことができます。そのために、構成ファイル名がパラメータとして、クライアントターミナルに渡されます。

例　：

```
terminal.exe config¥start.ini
terminal.exe test1.txt
terminal.exe "C:¥Program Files¥MetaTrader 4¥config¥settings25.ini"
```

ファイルへのフルパス（ドライブ名:¥サブディレクトリー¥ファイル名）が指定されていない場合には、クライアントターミナルのディレクトリ内を検索します。構成ファイルには、次の形式の行が含まれます。

[パラメータ]＝[値][Parameter] = [Value]

コメントはセミコロン（;）で始まり、処理の対象にはなりません。

構成ファイルのパラメータは、次に挙げるいくつかのグループに分類することができます。共通設定、プロキシサーバ設定（ターミナル設定の「サーバ」タブ）、FTP 設定（サーバ設定の「パブリッシャ」タブ）、EA 設定（サーバ設定の「エキスパートアドバイザー」タブ）、エキスパートまたはスクリプトの単独起動設定、テスター起動の設定です。

・・・・・・・途中は省略・・・・・・・・・・・・・

テスターの起動設定

- TestExpert ： テストの目的で起動するエキスパートの名前です。このパラメータが指定されていない場合には、テストは開始されません。
- TestExpertParameters ： パラメータを含むファイル（¥tester ディレクトリ）の名前です。このファイルは、テスト用エキスパートの「プロパティ」ウィンドウで「入力」タブの「保存」ボタンを押すことにより作成されます。通常は、デフォルトのパラメータと異なるパラメータを保存するときに使われます。テスト用エキスパートの「テスティング」及び「最適化」タブ内のその他のパラメータ（及びこのパラメータが指定されていない場合には「入力」タブ内のパラメータ）については、最後のテスト後に¥tester¥[エキスパート名].ini ファイルに自動保存された値が入力されます。t
- TestSymbol ： エキスパートのテストで使用する通貨ペアの名前です。このパラメータが指定されていない場合には、テスターで最後に使用された値が使われます。
- TestPeriod ： チャートの周期（1 分、5 分、15 分、30 分、1 時間、4 時間、1 日、1 週間、月）です。このパラメータが指定されていないときは、1 時間を使用します。
- TestModel ： テスティングモデルの種類（「レートごと」、「コントロールポイント」、「オープン価格のみ」）によって、0、1、または 2 のいずれかとなります。このパラメータが指定されていないときは、0（Every tick）を使用します。
- TestRecalculate ： 「再計算」フラグを有効／無効にします。「true」または「false」のいずれかの値をとります。このパラメータが指定されていない場合は、「false」の値が使われます。
- TestOptimization ： 最適化を有効／無効にします。「true」または「false」のいずれかの値をとります。このパラメータが指定されていない場合は、「false」の値が使われます。
- TestDateEnable ： 「日付を使用」フラグを有効／無効にします。「true」または「false」のいずれかの値をとります。このパラメータが指定されていない場合は、「false」の値が使われます。
- TestFromDate ： テストを開始する日付です。YYYY.MM.DD の形式になります。このパラメータが指定されていない場合には、この日付は 1970.01.01 となります。
- TestToDate ： テストを終了する日付です。YYYY.MM.DD の形式になります。このパラメータが指定されていない場合には、この日付は 1970.01.01 となります。
- TestReport ： テストレポートファイルの名前です。このファイルはクライアントターミナルのディレクトリ内に作成されます。例えば「tester¥MovingAverageReport」といった相対パスを指定することができます。ファイル名の中で拡張子が指定されていない場合には、「.htm」が自動的に付加されます。このパラメータが指定されていない場合には、テストレポートは作成されません。
- TestReplaceReport ： レポートファイルの反復記録を有効／無効にします。「true」または「false」のいずれかの値をとります。「false」の値を指定した場合で、同じ名前のレポートファイルが既に存在するときは、ファイル名の後に角括弧で括った数字が付加されます。例えば、「MovingAverageReport[1].htm」となります。このパラメータが指定されていない場合は、「false」の値が使われます。
- TestShutDownTerminal ： テスト終了後のターミナルのシャットダウンを有効／無効にします。「true」または「false」のいずれかの値をとります。このパラメータが指定されていない場合は、「false」の値が使われます。ユーザが「停止」ボタンを押した場合には、制御がユーザに移るため、このパラメータの値は「false」にフラッシュされます。

例：

```
; start strategy tester
TestExpert=Moving Average
TestExpertParameters=ma0.set
TestSymbol=EURUSD
TestPeriod=H1
TestModel=2
TestRecalculate=false
TestOptimization=false
TestDateEnable=true
TestFromDate=1970.01.01
TestToDate=2006.06.06
TestReport=MovingAverageReport
TestReplaceReport=false
TestShutdownTerminal=true
```

４．現状課題と次回検討事項

※「ShellExecuteA(...)」は、とても使いやすく、スクリプトの動作チェックには重宝するのですが、アプリを起動した後、アプリ終了如何に係らず、自身は終了してしまいます。（アプリ終了待たず）この動作は、実はスクリプトによる自動化を検討する際に「障害」になる場合があります。

※ MT2 側の「Strategy Tester」を一旦クリックして、バックテストのタブを表示させておかないと、起動したときにバックテストが進行する様子を表示しない。

（何か別に方法があるような気がするが！）

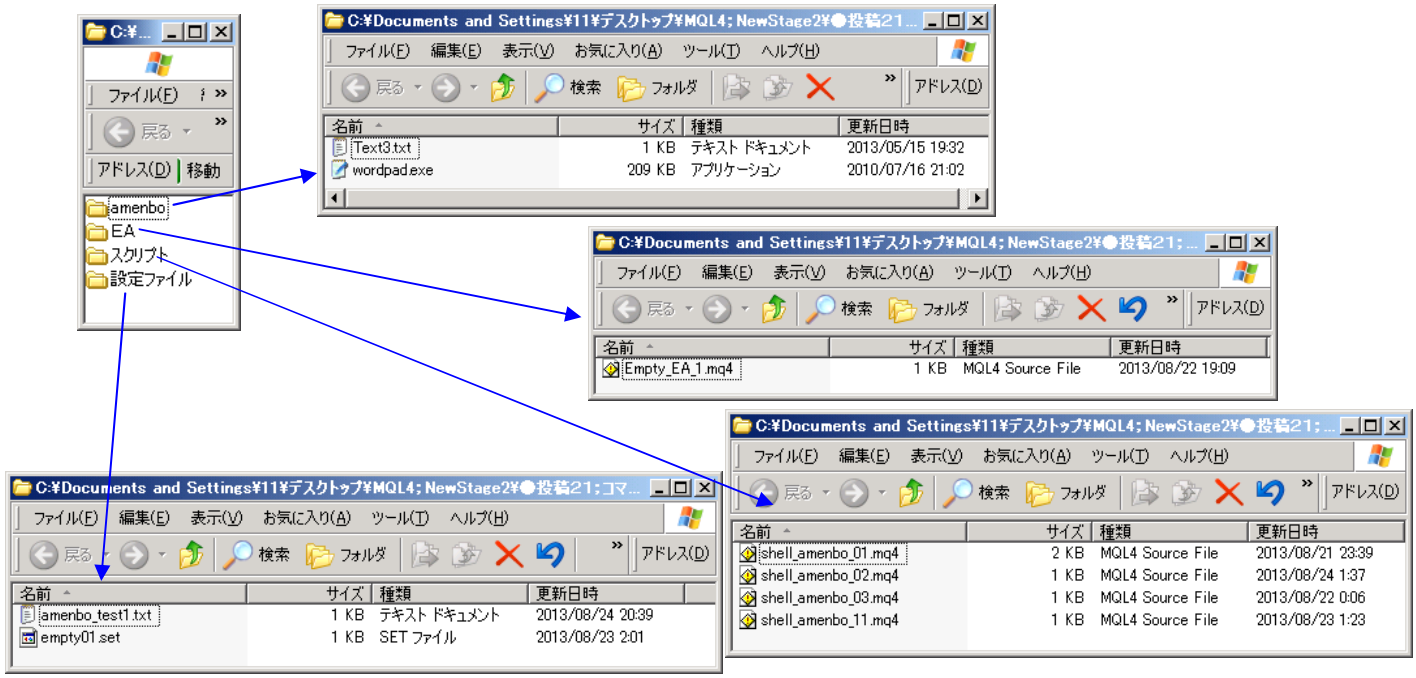
⇒**上記「２点」**は、次回への課題とします。

※次回検討事項；　「起動時の構成」について、更に調査予定です。

（目標は、自動化によるＥＡ開発の工数削減です）

５．添付スクリプトの解凍と内容について・・本稿で使用したファイル一式

※添付「shell_amenbo_set_01.zip」を解凍すると、下記のフォルダとファイルが作成されます。



以　上