○「　shared_memory.dll の使い方　」

・MQL4 は強力な言語で、かつ豊富な機能があり、特殊な機能も関数化すれば用は足りてしまうので、今までは正直言うと「DLL」を作る必要性は殆ど感じられませんでした。(中身のコードを非公開にしたいときぐらい、以外には！)
・しかし、最近 MQL4 では達成できない機能が必要になりました。何かと言うと EA 側のイベントをインディケータ側に伝える機能です、インディケータ側の情報は EA 側で参照出来ますが、その逆は「大域変数(GlobalVariable)」しか方法が無いように見えます。
・そこで以前、アメンボが MQL4 とは異なるトレードシステム用に作成した「メモリ共有」DLL を、MT4 に附属していたサンプルを参照して MQL4 用に作り直してみました。(何故か、以前作ったものでは動かなかった)
本稿では、この「shared_memory.dll」DLL の使い方と、MQL4 での実施例を解説します。

目次：

１．DLL 解説

（１）名称　　『　shared_memory.dll　』

（２）機能概要

・共有メモリ上で、1 個のレジスタ（r）と、3 個の配列（a、b、c）を MT4 のチャートに設定された「全ての EA、インディケータ」から、共有することが出来ます。
ただし、排他処理を入れていませんので取り扱いには注意が必要です。
・ベースの作成技術は同じですが、使用する場面の違いを意識できるように、レジスタと配列ではメモリ容量に差をつけました。
・共有メモリ（レジスタ、配列）を通して、EA やインディケータはデータのやり取りを行うことが出来ます。

（３）DLL の置場所

・「experts\liblaries\」フォルダ内に置きます。

（４）DLL の呼出し方

・使う機能のみを「#import」で宣言すれば OK です、例えば「配列 a 」のみを使うのであれば、下記のコードを先頭かヘッダファイルに書いておけば充分です。

```
#import "shared_memory.dll"
    double set_a();
    double write_a(double,int);
    double read_a(int);
    double close_a();
#import
```

（５）関数機能一覧

	関　数	機　能
レジスタ r	set_r()	レジスタ r 領域を設定します 使用する場合に宣言を実施
	write_r(データ,要素 No) ・データ；double 型 ・要素 No；int 型	要素 No にデータ（数値）を書込む 要素 No は「0～199」可能 （レジスタ数は 200 個）
	read_r(要素 No) ・要素 No；int 型	要素 No 中のデータを読出す
	close_r()	レジスタ r を閉じる
配列 a	set_a()	配列 a 領域を設定します
	write_a(データ,要素 No) ・データ；double 型 ・要素 No；int 型	要素 No にデータ（数値）を書込む 要素 No は「0～1999」可能 （要素数は 2000 個）
	read_a(要素 No) ・要素 No；int 型	要素 No 中のデータを読出す
	close_a()	配列 a を閉じる
配列 b	set_b()	配列 b 領域を設定します
	write_b(データ,要素 No) ・データ；double 型 ・要素 No；int 型	要素 No にデータ（数値）を書込む 要素 No は「0～1999」可能 （要素数は 2000 個）
	read_b(要素 No) ・要素 No；int 型	要素 No 中のデータを読み出す
	close_b()	配列 b を閉じる
配列 c	set_c()	配列 c 領域を設定します
	write_c(データ,要素 No) ・データ；double 型 ・要素 No；int 型	要素 No にデータ（数値）を書込む 要素 No は「0～3999」可能 （要素数は 4000 個）
	read_c(要素 No) ・要素 No；int 型	要素 No 中のデータを読出す
	close_c()	配列 c を閉じる

２．MQL4 実施例

（１）課題

・「EURJPY」の EA から Close データを、「USDJPY」のインディケータへ共有メモリ経由で渡して表示する。

・MQL4 では標準機能で異なる為替ペア・周期のデータを読み出せるので「USDJPY」チャート上に「EURJPY」データを表示することが可能ですが、「EA→インディケータ」方向のデータ授受は出来ません。（大域変数以外）

（２）手順

・「shared_memory.dll」を「experts\liblaries」フォルダにセット

・ＥＵＲＪＰＹ；①「myWrite_01.mq4（ＥＡ）」をセット

・ＵＳＤＪＰＹ；②「myRead_01.mq4（インディケータ）」をセット

（３）MQL コード

・データ送出側；

①myWrite_01.mq4（EA）

```
//   myWrite_01.mq4     EA
//
#import "shared_memory.dll"
     double set_a();
     double write_a(double,int);
     double read_a(int);
     double close_a();
#import

int init()
   {
     set_a();
     return(0);
```

```
  }

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

int start()
  {
   //set_a();

   for(int k=0;k<=500;k++)
   {
       double n=Close[k];
       write_a(n,k);
       //write_a(Ask,k);
   }

   return(0);
  }
```

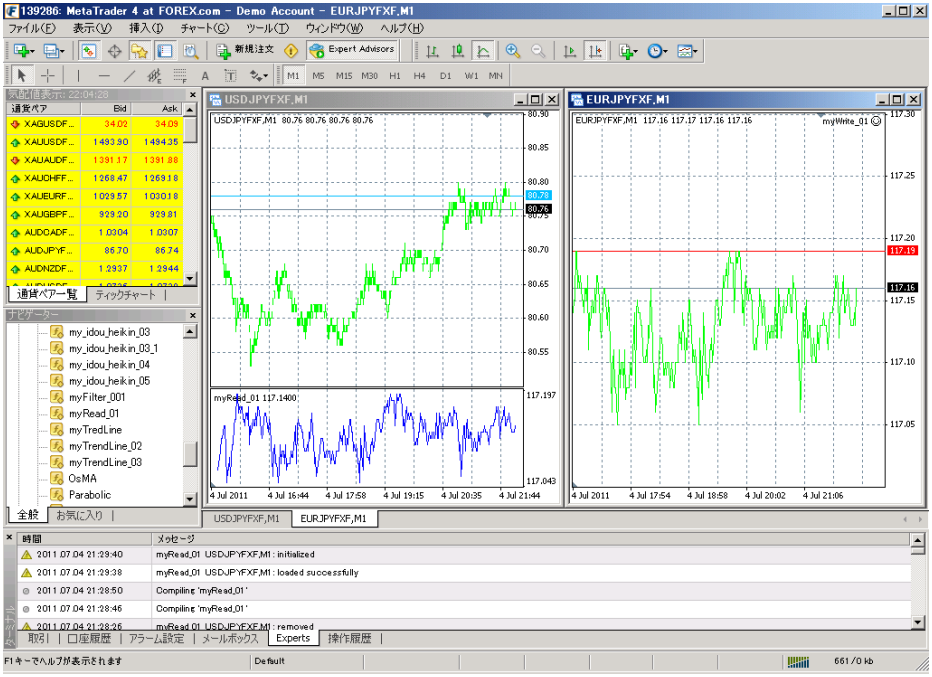
・データ受信側；

②myRead_01.mq4（インディケータ）

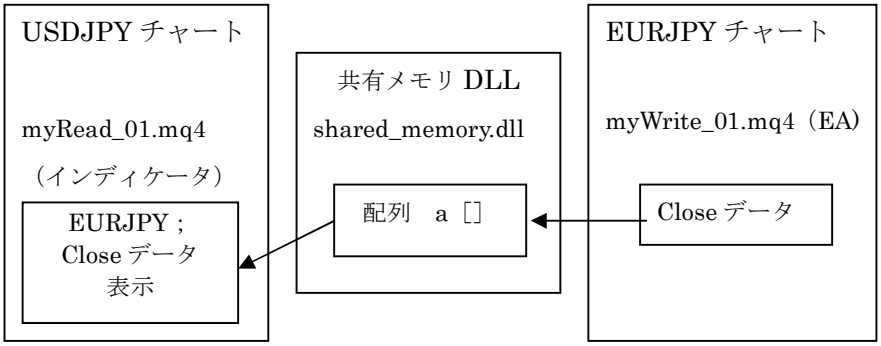
```
//        myRead_01.mq4       indicator
//
#import "shared_memory.dll"
    double set_a();
    double write_a(double,int);
    double read_a(int);
    double close_a();
#import

#property indicator_separate_window

double EUR_JPY[1000];

int init()
```

```
  {
   set_a();
   //
   SetIndexStyle(0,DRAW_LINE,STYLE_SOLID,1,Blue);
   SetIndexBuffer(0,EUR_JPY);
   return(0);
  }

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

int start()
  {
   int       counted_bars=IndicatorCounted();

   ArraySetAsSeries(EUR_JPY,true);
   //
  for(int i=0;i<=300;i++)
   {
       EUR_JPY[i]=read_a(i);
   }
   //Print("memory=",read_a(0));
   //close_a();

   return(0);
  }
```

（4）結果のチャート



※MQL コードとデータの流れ図



以　上