○「　ＭＱＬ５；翻訳まとめ（その１）データ呼び出し　」　翻訳のみ実施　2011.11.15

注意；　・本資料は、まだＭＴ５での動作・検証を行っていません、
近々の検証用資料として、英文資料を意訳しながら纏めたもの（メモ）です。
・以上の状況を理解されたうえで、本稿内容を参照ください。

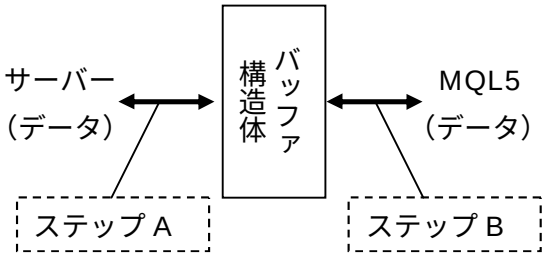
目次：

１．ＭＱＬ５とＭＱＬ４の基本的な相違

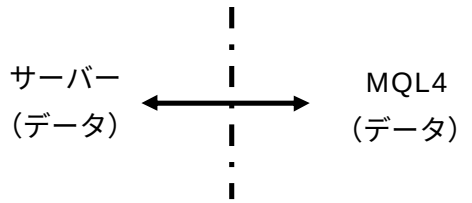
アメンボの理解（現状です！）；

・MQL5 では、ターミナル、サーバー間でデータをアクセスする際、
殆どの場合で「構造体かバッファ」を経由する仕様である。（例外あり）
このため、MQL4 では「１ステップ」でデータアクセス出来たものが、
MQL5 では「２ステップ」の手続きが必要になった。

＜イメージ図＞



（データ入手手順）	（売買命令手順）
ステップＡ； サーバーから データを 構造体に収納する ステップＢ； 構造体から データを取出す	ステップＢ； 売買条件を 構造体に収納 ステップＡ； 構造体から サーバーへ 売買条件を渡す



（データ入手手順）	（売買命令手順）
サーバーから いきなり データを取出す	売買条件を いきなり サーバーへ渡す

２．基本的な関数例

※MQL5 の場合に呼び出す手順概要を、大まかな手順「①、②、・・」で示す。

（１）EA、Scirpt で使用する「＊」印では「インディケータ表示」の場合は直接呼出せる

・Bars　----------

MQL5　① int bars=Bars(_Symbol,_Period);　(Bars(NULL,0);でも OK)

②以降は「bars」で呼び出せる

MQL4　「Bars」で直接呼出し

＊IndicatorCounted()----------

MQL5　① int counted_bars=BarsCalculated(int indicator_handle);

②以降「counted_bars」で呼び出せる

MQL4　「int counted_bars=IndicatorCounted();」で直接呼出し

・ASK、BID、・・・　----------

MQL5　① MqlTick last_tick;

SymbolInfoTick(_Symbol,last_tick);

②以降は「last_tick.ask、last_tick.bid」等で呼び出せる

MQL4　「ASK、BID、・・・」で直接呼出し

QL5 で使用する構造体；

```
struct MqlTick
  {
   datetime     time;          // Time of the last prices update
   double       bid;           // Current Bid price
   double       ask;           // Current Ask price
   double       last;          // Price of the last deal (Last)
   ulong        volume;        // Volume for the current Last price
  };
```

＊Close[i]、Open[i]、・・・Series データ　----------

MQL5　① MqlRate rates[];

CopyRate(NULL,0,0,100,rates);

②以降は「rates[i].close、rates[i].open」等で呼び出せる

MQL4　「Close[i]、Open[i]、・・・」で直接呼出し

QL5 で使用する構造体；

```
struct MqlRates
  {
   datetime     time;         // Period start time
   double       open;         // Open price
   double       high;         // The highest price of the period
   double       low;          // The lowest price of the period
   double       close;        // Close price
   long         tick_volume;  // Tick volume
   int          spread;       // Spread
   long         real_volume;  // Trade volume
  };
```

（2）インディケータ表示の場合「EA、Scirpt」の場合と、データ呼出手順が異なる場合あり
※インディケータ表示の「最小コード」例で示す

```
#property indicator_chart_window //別ｳｲﾝﾄﾞｳに表示する場合
#property indicator_buffers 1
#property indicator_plots   1
//  インディケータ用バッファ
double  Buffer[];
//
void OnInit()
  {
   SetIndexBuffer(0,Buffer,INDICATOR_DATA);
   //描画スタイルを記述可能→記述例（A）
  }
//
int OnCalculate(const  int  rates_total,
                const  int  prev_calculated,
                const  datetime  &time[],
                const  double  &open[],
                const  double  &high[],
                const  double  &low[],
                const  double  &close[],
                const  long  &tick_volume[],
                const  long  &volume[],
                const  int  &spread[] )
  {
//----
   for(int i=prev_calculated;i<rates_total;i++)
     {
      Buffer[i]=close[i]; //インディケータ用バッファに「終値データ」を設定
     }
//----  次のコールのために計算済み足数を返す
   return(rates_total);
  }
```

※インディケータ表示の場合「 OnCalculate（・・）{＊＊＊} 」を使用する。
この場合は、計算済み足数や、終値など、（・・）内に設定されたデータは、
下記の様に**直接呼び出すことができる**。

int counted_bars=prev_calculated;

double Close=close[i]、double Open=open[i]、int spread_=spread[i] など

※「rates_total」は足の総数、
「prev_calculated」は「以前に計算済みの足数」として使用可能
以前とは OnCalculate が一つ前にコールされた時、の意味。

※（A）描画スタイル記述例；

```
PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);       //Red 指定のこと
PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
```

以　上