○ＭＱＬ５；翻訳まとめ　「８つのデータ構造体（その１）」　翻訳のみ実施 2013.05.14

・アメンボです、
どうにも MQL5 は複雑（怪奇）と言うか、全体像が掴み難く苦闘中ですが、
裏を返せば、一度理解してしまうと良く出来たシステムなのかも知れません。
（アメンボは、翻訳を始めたばかりなので、到底まだそのレベルには達しませんが）
どうも、MQL5 はセミプロ・プログラマー以上の技量を持つトレーダーに
適したシステムの様にも思えます。
（しかし、「優れた入門・解説書」が出現すれば、普及が進むかも！？）

・ぼやいてばかりは止めて、解析を進めます。
アメンボは全体像が見えないとき、分類から入るのが好きなので、このシリーズも
分類から入って行くことにします。
さて、調査を進めると、MQL5 には「８つの定義済み（データ）構造体」があるようです。

注意；　　・本資料は、まだＭＴ５での動作・検証を行っていません、
・本編は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。
訳した資料がある程度たまったところで、ＭＴ５をダウンロードして確認して
いくつもりです。・・・アメンボは、まだＭＴ５は使ったことが無いのです！
（実機で未検証の内容ですので、誤訳があるかもしれません）
・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。
○本稿を「（その１）」としたのは、基本内容のみを記述したので、別の機会に応用や
実施例等を報告しようと考えているからです。

目次：

１．「８つの定義済み（データ）構造体」とは

・MQL5 に定義済みである、データを一纏めにして扱う構造体（即ちデータ構造体）は、
全部で「８個」あるようで、本稿では「４個」解説します。（下表）

	データ構造体	配列扱い	簡単な解説	備考
1	MqlDateTime	無？？	日時データ（date and time を扱う）の構造体	**本稿**
2	MqlParam	有	IndicatorCreate()でｲﾝﾃﾞｨｹｰﾀを作成する際に、ﾊﾟﾗﾒｰﾀ設定に使用	**本稿**
3	MqlRates	有	ヒストリカル・データ（含 price、volume、spread）を扱う	**本稿**
4	MqlBookInfo	有	Depth of Market（板表示）の情報取得に使用する	**本稿**
5	MqlTradeRequest		trade operations	
6	MqlTradeResult		trade request、OrderSend()	
7	MqlTradeTransaction		trade transaction（ﾄﾚｰﾄﾞ処理の流れ）記述情報を扱う	
8	MqlTick		現在の prices 情報を迅速に収集するために使用する	

・既にどこかで観た様な「構造体」もありますし、「MQL5 定義済みのデータ構造体」は、
これだけなんだ！、と言うのがアメンボの感想です。（もっと在りそうに思えるのですが）

２．解説１；MqlDateTime　構造体

（１）データ構造体定義

・本構造体は「日時データ」を一括して扱う構造体であり、８個の「int タイプ」から構成される。

```
struct MqlDateTime
  {
   int year;            // Year
   int mon;             // Month
   int day;             // Day
   int hour;            // Hour
   int min;             // Minutes
   int sec;             // Seconds
   int day_of_week;     // Day of week (0-Sunday, 1-Monday, ... ,6-Saturday)
   int day_of_year;     // Day number of the year
                        // (January 1st is assigned the number value of zero)
  };
```

※各メンバーの意味は、各行のコメントから判ると思いますので、特に解説しません。

（２）使用例

```
void OnStart()
  {
   datetime date1=D'2008.03.01';
   datetime date2=D'2009.03.01';

   MqlDateTime str1,str2;
   TimeToStruct(date1,str1);
   TimeToStruct(date2,str2);
   printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, day of year = %d",str2.day,str2.mon,str2.year,str2.day_of_year);
  }
        /*  Result:
          01.03.2008, day of year = 60
          01.03.2009, day of year = 59
        */
```

（３）補足＆参考；

－１．TimeToString 関数；

・「1970.01.01　00:00」を基点とした経過秒表示の時間を、"yyyy.mm.dd hh:mi"形式の文字列（時間表示）に変換する。

```
string  TimeToString(
   datetime  value,                          // 日時データ
   int       mode=TIME_DATE|TIME_MINUTES     // 出力形式
   );
```

パラメータ；

value　　[in]　「1970.01.01　00:00」からの経過時間（経過秒）

mode=TIME_DATE|TIME_MINUTES

[in] 文字列化した時間の表示形式

TIME_DATE　　指定時→　"yyyy.mm.dd"，・・デフォルト設定

TIME_MINUTES　指定時→　"hh:mm"，

TIME_SECONDS　指定時→　"hh:mm:ss".

返値；　　文字列

－２．日時データ（Date and Time）

・ＭＴ５（ＭＱＬ５）で扱う日時データ用の関数は以下の通り、
関数により「引数；あり、なし」の使い方あり。
（詳細解説は省略、実はアメンボも詳細には調べていません！）

返し値型	関数	動作
datetime	TimeCurrent()	最新のサーバー時間（datetime 形式）を返す。
datetime	TimeTradeServer()	クライアント・ターミナル内で計測した現在時間を返す。
datetime	TimeLocal()	ＰＣ（パソコン）上の時間を返す。
datetime	TimeGMT()	ＧＭＴ時間を返す。
int	TimeDaylightSavings()	夏時間用の補正値(int)を返す。
int	TimeGMTOffset()	ＧＭＴ時間とＰＣ時間の差分を返す。（夏時間補正を含む）
void	TimeToStruct()	datetime 型（1970.01.01 を基点とする経過秒数）のデータを、MqlDateTime 構造体に変換する。
datetime	StructToTime()	MqlDateTime 構造体の時間データを、datetime 型（1970.01.01 を基点とする経過秒数）データに変換する。

３．解説２；MqlParam　構造体

（１）データ構造体定義

※MqlParam の別名は「The Structure of Input Parameters of Indicators」であり、
「IndicatorCreate()」関数によって作成されるインディケータのインプット・パラメータを構造体形式で扱うためのもの。

```
struct MqlParam
  {
   ENUM_DATATYPE     type;          // type of the input parameter, value of ENUM_DATATYPE
   long              integer_value; // field to store an integer type
   double            double_value;  // field to store a double type
   string            string_value;  // field to store a string type
  };
```

・MQL5 では、大別して「２通りの方法」でインディケータを作成することが出来る。

－１．MQL4 で御馴染みの方法　　・・と言っても若干違いがありますが

－２．「IndicatorCreate()」関数による方法　・・この書式を使うメリットが未だ判らず！

IndicatorCreate()でインディケータを作成する際に、インプット・パラメータを設定するために、MqlParam 構造体の「配列」が使用されます。

(MqlParam を理解しようとしたら、芋蔓(いもずる)式に IndicatorCretae を調べることに成った！)

メンバー；

・ENUM_DATATYPE type　　；ｲﾝﾌﾟｯﾄ･ﾊﾟﾗﾒｰﾀのﾀｲﾌﾟ（下表；ENUM_DATATYPE から）を指定

・long integer_value　　；整数ﾀｲﾌﾟのﾊﾟﾗﾒｰﾀを設定するﾌｨｰﾙﾄﾞ

・long double_value　　；ﾀﾞﾌﾞﾙ･ﾀｲﾌﾟのﾊﾟﾗﾒｰﾀを設定するﾌｨｰﾙﾄﾞ

・long string_value　　；文字列ﾀｲﾌﾟのﾊﾟﾗﾒｰﾀを設定するﾌｨｰﾙﾄﾞ

※ENUM_DATATYPE

Identifier	Data type
TYPE_BOOL	bool
TYPE_CHAR	char
TYPE_UCHAR	uchar
TYPE_SHORT	short
TYPE_USHORT	ushort
TYPE_COLOR	color
TYPE_INT	int
TYPE_UINT	uint
TYPE_DATETIME	datetime
TYPE_LONG	long
TYPE_ULONG	ulong
TYPE_FLOAT	float
TYPE_DOUBLE	double
TYPE_STRING	string

・（例）インプットパラメータのタイプを指定する。

```
MqlParam params[];
params[2].type=TYPE_INT;
params[2].integer_value=MODE_EMA;
```

各「配列」の要素は、上記の様に直接に指定する必要がある。

・更に補足；（ﾎﾝﾄにｲﾓｽﾞﾙだ！）

※ENUM_MA_METHOD

ID	Description
MODE_SMA	Simple averaging
MODE_EMA	Exponential averaging
MODE_SMMA	Smoothed averaging
MODE_LWMA	Linear-weighted averaging

（２）使用例・・MQL5 コード例

```
void OnStart()
  {
//
//  移動平均線を描くのに、
//  －１．通常の「MQL4 と似た」方法だと、
//        「 iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE);  」と書けば済む。
//  これを、
//  －２．「IndicatorCreate()」を使ってコードを書くと、以下の様になる。

   MqlParam params[];
   int      h_MA,h_MACD;
   ArrayResize(params,4);
//--- set ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- set ma_shift
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- set ma_method
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- set applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- create MA
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);

// 更に続けて、上記の「h_MA」を使って「MACD」を作る
//－１．通常の「MQL4 と似た」方法だと、「iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);」
//－２．「IndicatorCreate()」を使ってコードを書くと、以下の様になる。
   ArrayResize(params,4);
//--- set fast ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- set slow ma_period
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- set smooth period for difference
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- set indicator handle as applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- create MACD based on moving average
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- use indicators
//--- . . .
//--- release indicators (first h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

（３）補足＆参考；

※IndicatorCreate()関数；

・インディケータを設定する一方法で、パラメータを MqlParam 構造体の配列で渡す。

```
int  IndicatorCreate(
   string           symbol,                          // symbol name
   ENUM_TIMEFRAMES  period,                          // timeframe
   ENUM_INDICATOR   indicator_type,    // indicator type from the enumeration ENUM_INDICATOR
   int              parameters_cnt=0,                // number of parameters
   const MqlParam&  parameters_array[]=NULL,         // array of parameters
   );
```

パラメータ；

symbol　　[in] 為替ペアの名前、「NULL」は現在表示中のチャートの為替ペア。

period　　[in] インディケータを表示するタイム・フレーム（周期）、
「0；ｾﾞﾛ」は現在表示中のタイム・フレーム（周期）。

indicator_type　[in] 「ENUM_INDICATOR」（※１）の一つで指定するインディケータのタイプ。

parameters_cnt　[in] MqlParam 構造体配列で設定するパラメータの数で、MAX 255 個まで。
・「0；ｾﾞﾛ」を設定すると、パラメータを指定しないことを意味する、
・「0；ｾﾞﾛ」以外の場合は、必ず MqlParam 構造体配列を設定必要あり。

parameters_array[]=NULL　[in]　インディケータに適用する MqlParam 構造体配列を指定する。

返し値；　・成功　→　作成したインディケータのハンドルを返す。
・失敗　→「INVALID_HANDLE」を返す。（INVALID_HANDLE 内容は別の機会に！）

※１；ENUM_INDICATOR

Indicator	IndicatorCreate()を使う場合	参考；MQL4 と似た記述法では
Accelerator Oscillator	IND_AC	iAC
Accumulation/Distribution	IND_AD	iAD
Average Directional Index	IND_ADX	iADX
ADX by Welles Wilder	IND_ADXW	iADXWilder
Alligator	IND_ALLIGATOR	iAlligator
Adaptive Moving Average	IND_AMA	iAMA
Awesome Oscillator	IND_AO	iAO
Average True Range	IND_ATR	iATR
Bollinger Bands®	IND_BANDS	iBands
Bears Power	IND_BEARS	iBearsPower
Bulls Power	IND_BULLS	iBullsPower
Market Facilitation Index	IND_BWMFI	iBWMFI
Commodity Channel Index	IND_CCI	iCCI
Chaikin Oscillator	IND_CHAIKIN	iChaikin
Custom indicator	IND_CUSTOM	iCustom
Double Exponential Moving Average	IND_DEMA	iDEMA
DeMarker	IND_DEMARKER	iDeMarker

Envelopes	IND_ENVELOPES	iEnvelopes
Force Index	IND_FORCE	iForce
Fractals	IND_FRACTALS	iFractals
Fractal Adaptive Moving Average	IND_FRAMA	iFrAMA
Gator Oscillator	IND_GATOR	iGator
Ichimoku Kinko Hyo	IND_ICHIMOKU	iIchimoku
Moving Average	IND_MA	iMA
MACD	IND_MACD	iMACD
Money Flow Index	IND_MFI	iMFI
Momentum	IND_MOMENTUM	iMomentum
On Balance Volume	IND_OBV	iOBV
OsMA	IND_OSMA	iOsMA
Relative Strength Index	IND_RSI	iRSI
Relative Vigor Index	IND_RVI	iRVI
Parabolic SAR	IND_SAR	iSAR
Standard Deviation	IND_STDDEV	iStdDev
Stochastic Oscillator	IND_STOCHASTIC	iStochastic
Triple Exponential Moving Average	IND_TEMA	iTEMA
Triple Exponential Moving Averages Oscillator	IND_TRIX	iTriX
Variable Index Dynamic Average	IND_VIDYA	iVIDyA
Volumes	IND_VOLUMES	iVolumes
Williams' Percent Range	IND_WPR	iWPR

特記１；

・既に述べた様に、MQL5 では「２通り」のインディケータ記述方法がある。

－１．MQL4 で御馴染みの方法

－２．「IndicatorCreate()」関数による方法

「－２」方法を使うと、どのようなインディケータも「**統一された記述法**」で指定できる。

どちらの方法で作成されるインディケータも同一品であり、場合に応じて使い分けがベター。

（でも、どんな場合に「統一された記述法」が効果を発揮するのだろう？？）

特記２；

「 ENUM_INDICATOR indicator_type= IND_CUSTOM 」の場合、即ち、

Custom indicator (ｶｽﾀﾑ･ｲﾝﾃﾞｨｹｰﾀ) を指定する場合、必ず MqlParam 構造体配列の

第１要素に対して下記の設定が必要となる。

・MqlParam custum_params[]; //「custum_params」部は任意名称

・custum_params [0].type= TYPE_STRING;

・custum_params [0].string_value="カスタム・インディケータの名称";

４．解説３；MqlRates

・この構造体は、「ヒストリカル・データ；価格、ボリューム、スプレッド」情報を収録する

（１）データ構造体定義

```
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
  };
```

※各メンバーの中で、「*tick_volume*;//Tick volume」と「*real_volume*;//Trade volume」の違いが、未だアメンボには釈然としないところあり？？

（２）使用例

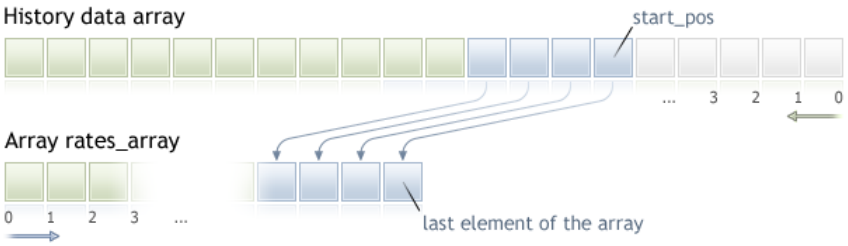
```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Bars copied: "+copied);
      string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
      string out;
      int size=fmin(copied,10);
      for(int i=0;i<size;i++)
        {
         out=i+":"+TimeToString(rates[i].time);
         out=out+" "+StringFormat(format,
                                  rates[i].open,
                                  rates[i].high,
                                  rates[i].low,
                                  rates[i].close,
                                  rates[i].tick_volume);
         Print(out);
        }
     }
   else Print("Failed to get history data for the symbol ",Symbol());
  }
```

（３）補足＆参考；

・またまた、コード例にイモズル式に解説しなければならない関数が出現しています、
本稿では、MQL4 から類推のつく関数は解説を省いて、最低限に留めます。

※CopyRates()関数；

・通常、チャート上のヒストリカル・データは「Series データ；現在足を配列要素[0]とする」として扱われています。
また、MqlRates 構造体配列にデータをセット（コピー）するには、CopyRates()関数を使用しますが、MqlRates 構造体配列にヒストリカル・データが格納される順番には、注意が必要です。（下図を参照）
ヒストリカル・データ（Historydata array）の最も古いデータが、MqlRates 構造体（Array rates_array）の配列要素[0]に収録されます。



・通常は、コピーすべきヒストリカル・データの量は不明の場合が多いので、MqlRates は、ダイナミック配列として指定することが多いです。→ MqlRates[];
・CopyRates（）関数には、「３通り」の呼出し方法（書式）があります。

＜書式１＞　コピーする「スタート位置」と、コピーする「足の数」を指定する

```
int  CopyRates(
   string           symbol_name,      // symbol name
   ENUM_TIMEFRAMES  timeframe,        // period
   int              start_pos,        // start position
   int              count,            // data count to copy
   MqlRates         rates_array[]     // target array to copy
   );
```

＜書式2＞　コピーする「スタート時間」と、コピーする「足の数」を指定する

```
int  CopyRates(
   string           symbol_name,      // symbol name
   ENUM_TIMEFRAMES  timeframe,        // period
   datetime         start_time,       // start date and time
   int              count,            // data count to copy
   MqlRates         rates_array[]     // target array to copy
   );
```

＜書式3＞　コピーする「スタート時間」と、コピーする「終了時間」を指定する

```
int  CopyRates(
   string           symbol_name,      // symbol name
   ENUM_TIMEFRAMES  timeframe,        // period
   datetime         start_time,       // start date and time
   datetime         stop_time,        // end date and time
   MqlRates         rates_array[]     // target array to copy
   );
```

パラメータ；

symbol_name	[in]	Symbol name.
timeframe	[in]	Period.
start_time	[in]	Bar time for the first element to copy.
start_pos	[in]	The start position for the first element to copy.
count	[in]	Data count to copy.
stop_time	[in]	Bar time, corresponding to the last element to copy.
rates_array[]	[out]	Array of MqlRates type.

返し値；

成功；コピーした要素の数を返す

失敗；「－１」を返す

失敗例；指定した足数が、チャート上の足数を超えていたとき、

チャートに未だサーバーからデータがダウンロードされていないとき、など

補足；

・CopyRates()を「土曜日」に下記条件で実行すると、「０」が返される。

「週足」、「start_time=Last_Tuesday」、「stop_time=Last_Friday」

理由は、週足のタイム・フレームはいつも「日曜」から始まるのため。

・現在足（未確定中の「足」）のみをCopyRate()する（得る）には、

「書式１」を使って「start_pos=0」、「count=1」とする。

※fmin()関数；

・fmin()は下記の MathMin()と同じ働きをします。

```
double  MathMin(
   double  value1,     // first value
   double  value2      // second value
   );
```

パラメータ；

value1	[in]	First numeric value. (数値)
value2	[in]	Second numeric value. (数値)

返し値；　２つの値のうち、小さいほうのデータ

補足；　同様な「関数」関係として、「MathMax()」と「fmax()」がある。

５．解説４；MqlBookInfo　構造体

※本構造体は以前の投稿「OnBookEvent()の使い方、他（その１）」で解説済みですが再度記述。

本構造体（配列）は、「DOM(depth of market)；板情報」のデータを提供するものです。

（１）データ構造体定義

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE    type;       // Order type from ENUM_BOOK_TYPE enumeration
   double            price;      // Price   板上に表示された価格
   long              volume;     // Volume  板上に表示されたボリューム（枚数）
  };
```

DOMはシステムが提供するので、ユーザーは使用時にこの型の構造体を宣言するだけで良い。

「The DOM is available only for some symbols.」と言う記述があるのだが！？

幾つかの為替ペアのみで使用可能と言うことは？？利用できない通貨ペアがある？？

※そしてまた、イモズルの始まりです。

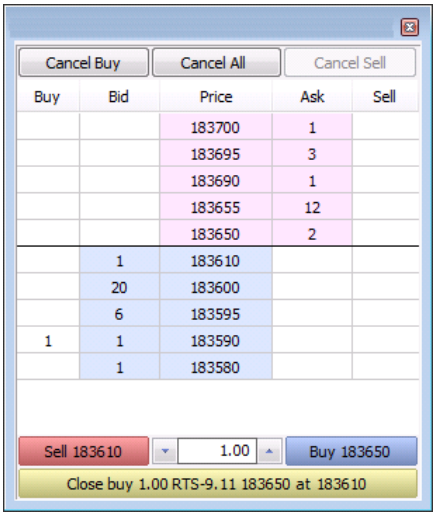
ENUM_BOOK_TYPE

Identifier	Description
BOOK_TYPE_BUY	Buy order (Bid)
BOOK_TYPE_SELL	Sell order (Offer)

・「ENUM_BOOK_TYPE」は、
入手したDOM（板）情報が「Bid」と「Ask」のどちらの側のデータであるかの判別に使う。
（たぶん）

・データ数が事前には不明なので、MqlBookInfo構造体（配列）は通常ダイナッミク配列で使用。

DOM(depth of market)；板情報



（２）使用例

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo for ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+":",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("Could not get contents of the symbol DOM ",Symbol());
     }
```

（３）補足＆参考；

※MarketBookGet()関数；

```
bool  MarketBookGet(
   string         symbol,      // symbol
   MqlBookInfo&   book[]       // reference to an array
   );
```

パラメータ；

symbol　[in]　Symbol name.（為替ペア名称）

book[]　[in]　Depth of Market（板情報）を集録する構造体（ダイナミック）配列

・通貨ペアを symbol()で指定すると、book[]で指定した構造体配列に、
板情報が（自動）集録される。

・この関数を利用する際は、前もって「MarketBookAdd()」関数で板情報を
開いておく必要のある場合がある。

以　上