○「 ＭＱＬ５；翻訳まとめ（その２） ｉＭＡとｉＡＤＸ」　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．ＥＡやスクリプトで用いる場合（例）

（１）事前解説を要する関数

① CopyBuffer()関数

用途；インディケータ用のバッファから、コピー先（プログラムで使用可能な）配列にデータをコピーする。

戻り値（return）；
正常終了時はコピーしたデータの数、
エラーの場合は「-1」

指定方法は、３通り可能。(C++のオーバーライド)；
①CopyBuffer（ｲﾝﾃﾞｨｹｰﾀ･ﾊﾝﾄﾞﾙ、ﾊﾞｯﾌｧNo、ｽﾀｰﾄ･ﾎﾟｼﾞｼｮﾝ、個数、ｺﾋﾟｰ先の配列）
②CopyBuffer（ｲﾝﾃﾞｨｹｰﾀ･ﾊﾝﾄﾞﾙ、ﾊﾞｯﾌｧNo、ｽﾀｰﾄ日時、個数、ｺﾋﾟｰ先の配列）
③CopyBuffer（ｲﾝﾃﾞｨｹｰﾀ･ﾊﾝﾄﾞﾙ、ﾊﾞｯﾌｧNo、ｽﾀｰﾄ日時、ｽﾄｯﾌﾟ日時、ｺﾋﾟｰ先の配列）

（2） ｉＭＡデータを利用する

※MQL4 の場合とは異なり、「iMA」で直接に移動平均値が返されるのでは無いことに注意。

呼出手順（概要）；

①インディケータ・ハンドル、バッファ配列を定義
　iMA では、必要なハンドル数=1、必要なバッファ数＝1
　バッファ No=「0」のみ
②インディケータ用のハンドルを設定
③インディケータ・データをバッファ（配列）にコピーする
④配列の内容（データ）が利用可能となる
⑤インディケータ用のハンドルを開放する

コード基本構成（例）；

①グローバル領域

```
double ema[];
int Handle;
int period=21;
```

②OnInit()

```
{
        Handle=iMA(_Symbol,_Period,period,MODE_EMA,PRICE_CLOSE);
        //Handle=iMA(NULL,0,21, MODE_EMA,PRICE_CLOSE);  でも良い
}
```

③OnTick()、OnStart()など

```
{
        ArraySetAsSeries(ema,true);
        CopyBuffer(Handle,0,0,10000,ema);
        // CopyBuffer(ﾊﾝﾄﾞﾙ名,ﾊﾞｯﾌｧ No,ｽﾀｰﾄ位置,ｺﾋﾟｰ個数,配列名);
```

④配列データを利用する

```
        double ema_10=ema[10];
        ・・・・・
}
```

⑤OnDeinit()

```
{
        IndicatorRelease(Handle);
}
```

（3）ｉＡＤＸデータを利用する

※MQL4 の場合とは異なり、「iADX」で直接に移動平均値が返されるのでは無いことに注意。

呼出手順（概要）；

①インディケータ・ハンドル、バッファ（配列）を定義

iADX では、必要なハンドル数=1、必要なバッファ数＝3

バッファ No	ライン種類
0	MAIN_LINE
1	PLUSDI_LINE
2	MINUSDI_LINE

②インディケータ用のハンドルを設定

③インディケータ・データをバッファ（配列）にコピーする

④配列の内容（データ）が利用可能となる

⑤インディケータ用のハンドルを開放する

コード基本構成（例）；

①グローバル領域

```
double ADX[],DIp[],DIm[];
int Handle;
int period=14;
```

②OnInit()

```
{
Handle=iADX(_Symbol,_Period,period);//Handle=iADX(NULL,0,14);でも OK
}
```

③OnTick()、OnStart()など

```
{
ArraySetAsSeries(ADX,true);
ArraySetAsSeries(DIp,true);
ArraySetAsSeries(DIm,true);
CopyBuffer(Handle,0,0,10000,ADX);
CopyBuffer(Handle,1,0,10000,DIp);
CopyBuffer(Handle,2,0,10000,DIm);
// CopyBuffer(ﾊﾝﾄﾞﾙ名,ﾊﾞｯﾌｧ No,ｽﾀｰﾄ位置,ｺﾋﾟｰ個数,配列名);
```

④配列データを利用する

```
double adx_10=ADX[10];
double dip_10=DIp[10];
double dim_10=DIm[10];
・・・・・
}
```

⑤OnDeinit()

```
{
IndicatorRelease(Handle);
}
```

２．インディケータを表示する例

（１）事前解説を要する関数

① OnCalculate()関数

用途；インディケータの表示に使用する。（専用）

特徴；インディケータ表示に最低限必要なデータが、直接（１ステップで）呼び出せる様に準備されている。

指定方法は、２通り可能。（C++のオーバーライド）；

＜第１形式＞・・表示に必要なバッファ数が「１個」の場合専用

```
int OnCalculate (
        const int rates_total,          // size of the price[] array
        const int prev_calculated,      // bars handled on a previous call
        const int begin,        // where the significant data start from
        const double& price[]           // array to calculate
                ){・・・・・}
```

※引数の意味；

・rates_total　　；総足数に等しい

・prev_calculated　　；前回「OnCalculate()」がコールされた時に指標の計算が終わっていた足数

・begin　　；計算をスタートする足

・price[]　　；表示（計算）対象のデータが入った配列

＜第２形式＞・・表示に必要なバッファ数が「複数個」の場合（１個の場合も使える）

```
int OnCalculate (
        const int rates_total,      // size of input timeseries
        const int prev_calculated,  // bars handled in previous call
        const datetime& time[],     // Time
        const double& open[],       // Open
        const double& high[],       // High
        const double& low[],        // Low
        const double& close[],      // Close
        const long& tick_volume[],  // Tick Volume
        const long& volume[],       // Real Volume
        const int& spread[]         // Spread
                ) {・・・・・}
```

※引数の意味；

・「rates_total」と「prev_calculated」は、第１形式の場合と同様

・その他の「Close、volume」等は、指標計算に用いる「series」データ

EA、Script の場合と異なり、上記の引数データは直接呼び出すことができる。

② BarsCalculated()関数

用途；インディケータの準備が出来たかの判定に使用する。

```
int  BarsCalculated(
        int  indicator_handle,     // indicator handle
               );
```

（使用例）

```
int is_data_ready=BarCalculated(ｲﾝﾃﾞｨｹｰﾀ･ﾊﾝﾄﾞﾙ);
```

※戻り値；・インディケータ・ハンドルが設定された直後のインディケータ・バッファ中の計算済みデータ数。
・まだ計算が始まっていない場合は「-1」を返す。

（2） ｉＭＡインディケータを表示する

・・「EA、Scirpt」の場合とは、データ呼出手順が異なる場合あり

呼出手順（概要）；

①インディケータ・ハンドル、表示用バッファ配列を定義
iMA では、必要なハンドル数=1、必要なバッファ数＝1（バッファ No=「0」のみ）
②インディケータ用のバッファを設定、「Series」データとして設定、
ハンドルを設定
③インディケータ・データをバッファ（配列）にコピーする
表示を円滑にするためには、計算済みデータの再計算を回避する必要あり
④配列の内容がインディケータとして表示される

コード基本構成（例）；

※インディケータ表示の「最小コード」例

```
//#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・ハンドル・MA 計算周期
double  MABuffer[];
int handle;
int period=21;
//
void OnInit()
  {
```

```
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   ArraySetAsSeries(MABuffer,true);
   handle=iMA(NULL,0,period,0,MODE_EMA,PRICE_CLOSE);
   return(0);
  }
//
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[] )
  {
//----  指標表示のための最小記述
//（A）全てのデータを表示に度に計算する場合例
        int copied=CopyBuffer(handle,0,0,rates_total,MABuffer);
//----  次のコールのために計算済み足数を返す
   return(rates_total);
  }
```

※（A）部；計算済みデータの再計算を行わないためには、例えば下記の記述方法あり

```
if(BarsCalculated(handle)<rates_total)  return(0);
//
int to_copy;
if(prev_calculated>rates_total ¦¦ prev_calculated<=0)  to_copy=rates_total;
        else
        {
                to_copy=rates_total-prev_calculeted;
                to_copy++;
        }
//
        CopyBuffer(handle,0,0,to_copy,MABuffer);
//
```

※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

（3）ｉＡＤＸインディケータを表示する

・・「EA、Scirpt」の場合とは、データ呼出手順が異なる場合あり

呼出手順（概要）；

①インディケータ・ハンドル、表示用バッファ配列を定義
iADX では、必要なハンドル数=1、必要なバッファ数＝3

バッファ No	ライン種類
0	MAIN_LINE
1	PLUSDI_LINE
2	MINUSDI_LINE

②インディケータ用のバッファをそれぞれ設定、
「Series」データとしてそれぞれ設定、ハンドルは１つ設定
③インディケータ・データをそれぞれバッファ（配列）にコピーする
表示を円滑にするためには、計算済みデータの再計算を回避する必要あり
④配列の内容がインディケータとして表示される

コード基本構成（例）；

※インディケータ表示の「最小コード」例

```
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//  インディケータ用バッファ・ハンドル・ADX 計算周期
double  ADX_Buffer[];
double  DIp_Buffer[];
double  DIm_Buffer[];
int handle;
int period=14;
//
void OnInit()
  {
   SetIndexBuffer(0,ADX_Buffer,INDICATOR_DATA);
   SetIndexBuffer(0,DIp_Buffer,INDICATOR_DATA);
   SetIndexBuffer(0,DIm_Buffer,INDICATOR_DATA);
   //
   ArraySetAsSeries(ADX_Buffer,true);
   ArraySetAsSeries(DIp_Buffer,true);
   ArraySetAsSeries(DIm_Buffer,true);
   //
   handle=iADX(NULL,0,period);
   return(0);
  }
//
```

```
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[] )
  {
//----  指標表示のための最小記述
//（A）全てのデータを表示に度に計算する場合例
        CopyBuffer(handle,0,0,rates_total,ADX_Buffer);
        CopyBuffer(handle,1,0,rates_total,DIp_Buffer);
        CopyBuffer(handle,2,0,rates_total,DIm_Buffer);
//----  次のコールのために計算済み足数を返す
   return(rates_total);
  }
```

※（A）部；計算済みデータの再計算を行わないためには、例えば下記の記述方法あり

```
if(BarsCalculated(handle)<rates_total)  return(0);
//
int to_copy;
if(prev_calculated>rates_total ¦¦ prev_calculated<=0)  to_copy=rates_total;
        else
        {
                to_copy=rates_total-prev_calculeted;
                to_copy++;
        }
//
        CopyBuffer(handle,0,0,to_copy,ADX_Buffer);
        CopyBuffer(handle,1,0,to_copy,DIp_Buffer);
        CopyBuffer(handle,2,0,to_copy,DIm_Buffer);
//
```

以　上