

## MT4 编程入门 6：市场数据取值

(2008-07-05 21:24:34)

转载

标签： 分类： [MT4 编程](#)

[理财](#)

[k 线](#)

[双精度](#)

[开盘价](#)

[最高价](#)

[杂谈](#)

一、预定义数组（MT4 已经定义好的，可以直接使用）

开盘价、最高价、最低价、收盘价、成交量等历史数据，每根 K 线蜡烛都各有一个，所以必须用数组来解决问题，MT4 有几个预定义数组：

开盘价 `Open[]`、最高价 `High[]`、最低价 `Low[]`、收盘价 `close[]`、成交量 `Volume[]`、所属时间 `Time[]`

类型为双精度 `double` 型（即精确小数）

这里有一个位置的问题，右边第一根 K 线蜡烛(即最新的蜡烛)的编号为 0，第二根蜡烛编号 1，第三根蜡烛编号 2，也就是说从右向左倒着数过去。

`Open[0]`、`High[0]`、`Low[0]`、`Close[0]`，表示最新的开盘价、最高价、最低价、收盘价

`Open[1]`、`High[1]`、`Low[1]`、`close[1]`，表示第 2 根 K 线蜡烛的开盘价、最高价、最低价、收盘价

`Open[2]`、`High[2]`、`Low[2]`、`close[2]`，表示第 3 根 K 线蜡烛的开盘价、最高价、最低价、收盘价

`Open[3]`、`High[3]`、`Low[3]`、`close[3]`，表示第 4 根 K 线蜡烛的开盘价、最高价、最低价、收盘价

`Open[i]`、`High[i]`、`Low[i]`、`close[i]`，表示第  $i+1$  根 K 线蜡烛的开盘价、最高价、最低价、收

盘价

以此类推。。。。。

注意：这些是数组，用的是方括号。

## 二、预定义变量

买入价、卖出价是实时价格，MT4 用预定义变量 Ask 和 Bid 表示，数值类型为 double 双精度

还有一些预定义变量，如：

Bars 表示图表中的蜡烛数，类型为 int 整数型

Digits 表示当前货币对的小数位，类型为 int 整数型，无日元币对为 4，有日元币对为 2，黄金石油等一般也为 2

Point 表示当前货币对的点值，类型为双精度 double 型，无日元币对为 0.0001，有日元币对为 0.01。与 Digits 正好相反。

## 三、指标函数

### 1、价格、成交量、时间

它们都有三个参数：货币对名、K 线周期、位置序号

开盘价：iOpen(symbol,timeframe,shift) 双精度 double 型

收盘价：iClose(symbol,timeframe,shift) 双精度 double 型

最高价：iHigh(symbol,timeframe,shift) 双精度 double 型

最低价：iLow(symbol,timeframe,shift) 双精度 double 型

成交量：iVolume(symbol,timeframe,shift) 双精度 double 型

所属时间：iTime(symbol,timeframe,shift) 日期时间 datetime 型

K 线周期为：1 分钟图(PERIOD\_M1)、5 分钟图(PERIOD\_M5)、15 分钟图(PERIOD\_M15)、30 分钟图(PERIOD\_M30)、

1 小时图(PERIOD\_H1)、4 小时图(PERIOD\_H4)、日线图(PERIOD\_D1)、周线图(PERIOD\_W1)、月线图(PERIOD\_W1)

例：

iOpen("USDJPY",PERIOD\_H1,0) 表示美元兑日元 1 小时图最新 K 线蜡烛的开盘价

iClose("EURUSD",PERIOD\_H4,2) 表示欧元兑美元 4 小时图第 3 根 K 线蜡烛的收盘价

iClose("GBPUSD",PERIOD\_H1,i) 表示英镑兑美元 1 小时图第 i+1 根 K 线蜡烛的收盘价

iHigh(NULL,0,0) 既不指定商品,也不指定 K 线周期,用在谁就是谁,用在哪就是哪

## 2、移动平均值。双精度 double 型

iMA(symbol, timeframe, period, ma\_shift, ma\_method, applied\_price, shift)

参数共 7 个, 分别为: 商品名称、K 线周期、均线周期、均线偏移、平均模式、价格种类、位置

均线周期: 10 天平均线的均线周期为 10, 20 日均线的均线周期为 20

均线偏移: 均线位置整体左右移动的位置偏移量

平均模式: 简单移动平均(MODE\_SMA)、指数移动平均(MODE\_EMA)、平滑移动平均线(MODE\_SMMA)、线性加权移动平均线(MODE\_LWMA)

价格种类: 收盘价(PRICE\_CLOSE)、开盘价(PRICE\_OPEN)、最高价(PRICE\_HIGH)、最低价(PRICE\_LOW)、中值(PRICE\_MEDIAN)、5(PRICE\_TYPICAL)、6(PRICE\_WEIGHTED)

例 1: iMA("EURUSD",PERIOD\_H1,20,0,MODE\_SMA,PRICE\_CLOSE,0)

表示: 欧元 1 小时图上, 以收盘价计算的, 20 小时简单移动平均线, 最新 K 线所对应位置的值

例 2: iMA(NULL,0,20,0,MODE\_EMA,PRICE\_CLOSE,2)

表示: 在当前商品、当前 K 线周期图表上, 以收盘价计算的, 20(天)指数移动平均线第 3 根 K 线所对应位置的值

其他如 MACD 指标、威廉指标、RSI、SAR、布林线等指标取值都与移动平均线指标相类似

## 3、在数组中求元素的移动平均值。双精度 double 型

iMAOnArray(数组名, 总数, 平均周期, 均线偏移, 平均模式, 位置)

这也与 iMA 函数差不多, 不过数据源变为数组

从数组中不但可以求得移动平均值, 还可以求得 RSI 等指标值

#### 4、求自定义指标的值

我们经常自己编一些自定义指标，可用 iCustom 函数来取得自定义函数的值

iCustom(商品名,K 线周期,自定义指标名称,自定义指标参数 1,参数 2,参数 3,,,自定义指标线编号,位置)

如果自定义指标只有一根指标线，则自定义指标线的编号为 0。

如果自定义指标有多根指标线，则第一条自定义指标线的编号为 0，第二条为 1，第三条为 2。。。

例如：iCustom(NULL,0,"mymacd",12,26,9,2,0) (12,26,9)为自定义指标 mymacd 的三个参数  
表示：求当前图表中，自定义指标 mymacd(12,26,9)的第 3 条指标线在最新位置的值

抛砖引玉，这里只是有代表性地列了几个函数，详细请查阅《MT4 编程手册》

## MT4 编程入门 9：MT4 自定义指标的结构

(2008-07-05 21:27:51)

转载

标签： 分类： [MT4 编程](#)

[mt4](#)

[编程](#)

[语言](#)

[杂谈](#)

MT4 自定义指标一般由四个部分构成：

(1)文件头部

(2)参数、变量和数组的定义

(3)初始化函数 init()

(4)主函数 start()

例：

### MACD指标源码

```
#property indicator_separate_window
#property indicator_buffers 2 文件头部分
#property indicator_color1 Silver
#property indicator_color2 Red 预处理程序
#property indicator_width1 2

extern int FMA=12;
extern int SMA=26; 定义参数变量
extern int SgMA=9;

double Buffer[]; 定义数组
double SBuffer[];

int init()
{
    SetIndexStyle(0,DRAW_HISTOGRAM) 设置指标线型
    SetIndexStyle(1,DRAW_LINE);
    SetIndexDrawBegin(1,SgMA); 设置划线起始位置
    IndicatorDigits(Digits+1); 设置小数位
    SetIndexBuffer(0,Buffer);
    SetIndexBuffer(1,SBuffer); 设置指标数组
    IndicatorShortName("MACD("+FMA+", "+SMA+
        设置指标名称 "+SgMA+"");
    SetIndexLabel(0,"MACD"); 设置指标线标签
    SetIndexLabel(1,"Signal");
    return(0);
}

int start()
{
    int limit; 主函数
    int counted_bars=IndicatorCounted();
    if(counted_bars>0) counted_bars--;
    limit=Bars-counted_bars;

    for(int i=0; i<limit; i++)
        Buffer[i]=iMA(NULL,0,FMA,0,1,0,i)
            -iMA(NULL,0,SMA,0,1,0,i);

    for(i=0; i<limit; i++)
        SBuffer[i]=iMAOnArray(Buffer,Bars,SgMA,0,0,i);

    return(0);
}
```

一、文件头部，也称为预处理程序

预处理程序以“#”开头，行尾无语句结束符“;”

常用的预处理程序有：

1、#property indicator\_chart\_window

把指标显示在主图。如：均线、SRA 等类指标用到此语句

## 2、#property indicator\_separate\_window

把指标显示在副图。如：MACD、RSI、威廉等类指标用到此语句

## 3、#property indicator\_buffers 3

显示 3 根指标线

## 4、#property indicator\_color1 Red

第 1 根指标线的颜色为 Red

## 5、#property indicator\_width1 1

第 1 根指标线的粗细分别为 1

## 6、#property indicator\_level1 0.00

在 0.00 值位置横划 1 条虚线

## 二、参数、变量和数组的定义

全局性的参数、变量、数组在此定义，局部变量可在 start()函数中定义

## 三、初始化函数 init()

init()在自定义指标加载时运行一次。

初始化函数的功能是“设置”。如果自定义指标需要划线，则必然用到此函数

## 四、主函数 start()

当数据有变动时，start()就被触发。数据变动一次，start()就运行一次。

自定义指标的编程主要依靠此函数进行。

start()函数的作用主要是取值和给指标线赋值，报警也在此函数内发起。

另外，还有一个反初始化函数 `deinit()`

`deinit()`在自定义卸载时运行一次，可用以去除指标加载时 `init()`所做的初始化操作。



要画指标线，只要在程序中写明以下几点就可以了：

第一、明确指标线所在窗口，是主图还是副图

第二、要建立数组，用以保存指标线在各个位置的值。

第三、要建立指标线与数组的对应关系，哪个数组对应哪条指标线

第四、要明确指标线的线型，是曲线还是柱线或者是箭头

第五、如果指标线是箭头，还要说明是哪种箭头

第六、给数组赋值

其中：

第一、二条写在文件头部中，

第三、四、五条写在 `init()` 函数中（`init` 函数仅在指标加载时运行一次）

第六条写在 `start()` 函数中（`start` 函数在数据发动变动时运行，变动一次运行一次）

下面以 MACD 为例说明

我们知道，MACD 指标由二条曲线和一组红绿柱线组成。（下图一）

其中：

白线是二根均线的差；

紫线是白线的移动平均线；

红绿柱线则是白线和紫线的差，白线上穿紫线，出现红柱，下穿则出现绿柱。



我们从简单入手，先去除紫线和红绿柱线，仅保留其中的那根白线，来看白线是怎样画出来的。

下面是全部语句：

```
#property indicator_separate_window
#property indicator_color1 White
#property indicator_level1 0

extern int FMA=12;
extern int SMA=26;

double   buf[];

int init()
{
    SetIndexBuffer(0,buf);
    SetIndexStyle(0,DRAW_LINE);
    return(0);
}
```

```
    }  
int start()  
{  
    int limit=Bars-IndicatorCounted();  
  
    for(int i=0; i<limit; i++)  
    {  
        buf[i]=  
            iMA(NULL,0,FMA,0,1,0,i)  
            -iMA(NULL,0,SMA,0,1,0,i);  
    }  
    return(0);  
}
```

说明如下：

## 单线 MACD 指标程序语句说明

```
#property indicator_separate_window  
#property indicator_color1 White  
#property indicator_level1 0  
extern int FMA=12;  
extern int SMA=26;  
double buf[];
```

指标放在副图  
指标线为白色  
在零轴画一条水平线  
定义参数变量  
定义数组

```
int init()  
{  
    SetIndexBuffer(0,buf);  
    SetIndexStyle(0,DRAW_LINE);  
    return(0);  
}
```

init 函数  
设置指标线数组  
设置指标线线型

```
int start()  
{  
    int limit=Bars-IndicatorCounted();  
    for(int i=0; i<limit; i++)  
    {  
        buf[i]=  
            iMA(NULL,0,FMA,0,1,0,i)  
            -iMA(NULL,0,SMA,0,1,0,i);  
    }  
    return(0);  
}
```

start 函数  
循环语句  
循环体  
将两条均线的差值赋值给数组

=====  
=====  
以下为上述语句的简要说明

#property indicator\_separate\_window

指标放在副图

#property indicator\_color1 White

第一条指标线为白色

```
#property indicator_level1 0
```

在副图中零值位置上画一条水平横线，

```
extern int FMA=12;
```

```
extern int SMA=26;
```

设立二个整数型变量，默认值为 12 和 26，允许外部修改值

```
double   buf[];
```

设立一个数组

```
int init()
```

初始化函数。该函数在指标加载时只运行一次。**init** 是系统默认的函数名，但使用时仍需要进行创设，所以要加定义符 **int**

```
{
```

```
    SetIndexBuffer(0,buf);
```

设置数组 **buf** 为第一条指标线

```
    SetIndexStyle(0,DRAW_LINE);
```

设置第一条指标线线型为连续曲线

```
    return(0);
```

函数结束语句

```
}
```

```
int start()
```

指标触发函数。与 **init** 函数不同，该函数在有数据变化时被触发，如果数据被不断更新，则该函数将不断执行。**start** 也是系统默认的函数名，但使用时也仍然需要进行创设，所以也要加定义符 **int**

```
{
```

```
    int limit=Bars-IndicatorCounted();
```

自定义一个变量 **limit**，并赋值

Bars 是图表中的柱数

IndicatorCounted()缓存中的柱数，就是已经计算过的有值的柱数

这样 limit 的值就是未经计算的柱数，这样就可以起到优化程序的作用。

```
for(int i=0; i<limit; i++)
```

循环语句。

循环从 i=0 开始，每循环一次 i 值增加 1，一直循环到 i<limit 不满足时结束循环

由于循环变量 i 为一个新变量，所以要先定义，加上整型变量定义符 int

下面大括号为循环体，此例中只一条语句

```
{  
    buf[i]=  
        iMA(NULL,0,FMA,0,1,0,i)  
        -iMA(NULL,0,SMA,0,1,0,i);  
}
```

给数组 buf 赋值，其值分别为相应位置上两条均线的差

i 是水平位置序号值，即烛柱从右到左的序号，右边第一个烛柱序号为 0

```
return(0);
```

start 函数结束

```
}
```

## MT4 编程入门 11：MT4 的报警

(2008-07-05 21:31:36)

转载

标签： 分类： [MT4 编程](#)

[mt4](#)

[编程](#)

[语言](#)

[杂谈](#)

报警功能是 MT4 的一大特色。

它可以在预定的条件达到时，发出警报。



与指标画线相比，报警语句显得非常简单，

只要在判断语句中加一个报警语句即可

报警方式有：弹出窗口报警、音乐报警、邮件报警等。

如果邮箱开通了手机短信通知，则邮件报警的内容会即时转发到手机上。

#### 1、弹出窗口报警：

当(条件达到)执行此语句时，以弹出窗口警告。

格式：Alert(内容 1,内容 2,内容 3,内容 4);

报警内容为字符串型，内容之间加逗号

例如：

```
Alert( Symbol(),"4 小时图 MACD 上穿零轴");
```

## 2、音乐报警：

当(条件达到)执行此语句时，播放一段音乐。

格式：PlaySound("音乐文件名.wav");

文件类型为 wav 格式，并且保存在 C:\Program Files\MetaTrader4\sounds 目录中

文件名加引号

## 3、邮件报警：

当(条件达到)执行此语句时，发送一个邮件。

（收发件人地址在 MT4 系统中设置详见《[MT4 编程实例 1：一个简单的小程序，让你的手机摇身变成外汇行情接收机](#)》）

格式：SendMail(标题 1+标题 2, 内容 1+内容 2);

标题之间以加号连接，内容之间也以加号连接

邮件标题和邮件内容以逗号间隔

下面是《价格穿越某均线报警》举例

```
+-----  
  
#property indicator_chart_window  
  
extern int 警戒均线=20;  
  
int mark=0;  
  
int start()  
{  
    if( iHigh(0,0,0) >= iMA(0,0,警戒均线,0,MODE_SMA,PRICE_CLOSE,0)  
        && iHigh(0,0,1) < iMA(0,0,警戒均线,0,MODE_SMA,PRICE_CLOSE,1)  
        && mark != 1 )  
    {  
        Alert(Symbol(),"向上触及 30 均线");  
        mark = 1;  
    }  
}
```



```

if( iLow(0,0,0) <= iMA(0,0,警戒均线,0,MODE_SMA,PRICE_CLOSE,0)
    && iLow(0,0,1) > iMA(0,0,警戒均线,0,MODE_SMA,PRICE_CLOSE,1)
    && mark != 2 )
{
    Alert(Symbol(),"向下触及",警戒均线,"均线");
    mark = 2;
}
return(0);
}

```

+-----

部分语句说明：

#property indicator\_chart\_window

此句是把程序放在主图，当然这此例中放在副图也一样

extern int    定义一个外部参数变量，整数型，允许外部值修改

int            定义一个整数型变量

int start()    定义触发函数

if()           判断

iHigh()       最高价值函数

iLow()        最低价值函数

iMA()         移动平均线值函数

Alert()       报警函数

Symbol()      商品名称函数

&&            逻辑运算符“并且”

!=            逻辑运算符“不等于”

MODE\_SMA     简单移动平均模式

PRICE\_CLOSE   以收盘价计算

再说一下自定义变量 mark 的作用：

mark 的初值是 0，当上穿报警时给 mark 赋值 1，当下穿报警时给 mark 赋值 2。

这样当 mark 的值为 1 时，说明已经对上穿报过警了，就不能再次对上穿报警；

当 mark 的值为 2 时，说明已经对下穿报过警了，就不能再次对下穿报警。

这样就起到了消除重复报警的作用。

=====

语句简要解释如下：

=====

#property indicator\_chart\_window

指标放在主图

extern int 警戒均线=20;

设立一个自定义变量,允许外部值修改,整数形,变量名为"警戒均线",默认值 20

int mark=0;

设立一个自定义变量,整数型,变量名为 mark，并赋初值 0

此变量在后面用于记录是否报警，设计是这样的：

如果 mark=0，则从未报过警

如果 mark=1，则已经向上报过警

如果 mark=2，则已经向下报过警

int start()

设立触发函数 start。start 为系统规定函数名，函数内容自定义。当数据变动时，start 函数被触发

{

```
if( iHigh(0,0,0) >= iMA(0,0,警戒均线,0,MODE_SMA,PRICE_CLOSE,0)
    && iHigh(0,0,1) < iMA(0,0,警戒均线,0,MODE_SMA,PRICE_CLOSE,1)
    && mark != 1 )
```

条件判断语句。这里用到逻辑运算符&&，就是“并且”，条件有三个，三个条件要同时成立，则整个条件才成立

第一个条件：最高价大等于均线。iHigh 是烛柱最高价取值函数，iMA 是均线取值函数

第二个条件：前一烛柱最高价小于均线

第三个条件：mark 不等于 1。如果 mark 不等于 1，就说明指标没有对上穿报过警

```
{
    Alert(Symbol(),"向上触及 30 均线");
    mark = 1;
}
```

花括中为条件执行语句。Alert 是报警语句，Symbol()是商品名称取值函数

报警语句执行后，给 mark 赋值 1，这样就记录了已经向上报过警了

```
if( iLow(0,0,0) <= iMA(0,0,警戒均线,0,MODE_SMA,PRICE_CLOSE,0)
    && iLow(0,0,1) > iMA(0,0,警戒均线,0,MODE_SMA,PRICE_CLOSE,1)
    && mark != 2 )
```

```
{
    Alert(Symbol(),"向下触及",警戒均线,"均线");
    mark = 2;
}
```

```
return(0);
```

start 函数结束语句，返回零值

```
}
```

# 如果挂单成交了用什么函数判断!

如果挂单成交了用什么函数判断!

**bull** 发表于 2008-4-21 13:42

直接判断的函数没有，

建议采用如下逻辑：

- 1、循环比较挂单数量，如果少于你设定的数目，则认为其中的挂单成交
- 2、进一步查找持仓单，获得新入场的订单，此订单就是挂单入场的那个订单，对他进行你想做的处理就行了

**changkangwei** 发表于 2009-1-16 16:20

哈哈 老帖真的很有用

**changkangwei** 发表于 2009-1-17 14:48

```
[code]void DoUp()
{
int cnt, ticket, total;
double TicketA,TicketB,TicketC,Openprice;
total = OrdersTotal();
Openprice = OrderOpenPrice();
//+开仓+
if(total == 0 )
{ TicketA=OrderSend(Symbol(),OP_BUY,Lots,Ask,Slip,Ask-SL*Point,Ask+TP*Point,"",20090
110,0,Red);
if(TicketA>0)
{ if(OrderSelect(TicketA,SELECT_BY_TICKET,MODE_TRADES))Print("Buy:",OrderOpenPric
e()); }
}
//+第一次锁仓+
for(cnt=0;cnt<total;cnt++)
{ OrderSelect(cnt, SELECT_BY_POS, MODE_TRADES);
if( total == 1 )
{ TicketB=OrderSend(Symbol(),OP_SELLSTOP,Lots,NormalizeDouble(Openprice+70,Digi
ts),Slip,
NormalizeDouble(Openprice+70,Digits)+SL*Point,NormalizeDouble(Openprice+70,Digits
)-TP*Point,"",0,1,Red);
}
}
//+第二次锁仓+
```

```

for(cnt=0;cnt<total;cnt++)
{ OrderSelect(cnt, SELECT_BY_POS, MODE_TRADES);
  if( total = 2 )
  { TicketB=OrderSend(Symbol(),OP_BUYSTOP,Lots,NormalizeDouble(Openprice,Digits),Slip,
    NormalizeDouble(Openprice,Digits)+SL*Point,NormalizeDouble(Openprice,Digits)-TP*Point,"",0,1,Red);
  }
}
return(0);
}[/code]

```

请版们帮我看写的对不对

---

```

int iHighest(string symbol,    int timeframe,    int type,    void count,
             void start)

```

根据类型返回最大值转移的一个具体数字。

参量:

<b>symbol</b>	- 需应用到计算指标的货币对数据 NULL 意味当前货币对名称。。
<b>timeframe</b>	- 时间周期。可以是 <a href="#">时间周期列举</a> 的任意值。0 意味着当前图表的时间周期。
<b>type</b>	- 系列数组的识别符。它可以是 <a href="#">系列数据识别符列举</a> 的任意值。
<b>count</b>	- 周期数字。
<b>start</b>	- 移动显示与当前相关的柱，采取数据。

示例:

```

double val;

// 在范围内 20 个连续柱计算最大值

// 在当前图表上从第 4 个至第 23 个的索引

val=High[iHighest(NULL,0,MODE_HIGH,20,4)];

```

[iHigh iLow](#) →