**案例一 止盈止损**

本例中模板以止赢30跳，止损20跳为例，实际使用时可以转换为开仓价格的百分比或其它任何设置的变量进行止盈止损。

**分析**

要编写止盈止损的代码，关键是找准基准价格（本例以建仓价格MyEntryPrice为基准）和止盈止损价格

1. **止盈30跳**
   * 多仓情况，止盈条件可写为当最高价比建仓价格高30跳，表达式为：

最高价>= 建仓价格 + 止盈值（30跳表示的点位）

High>= MyEntryPrice + 30\* MinMove\*PriceScale

* + 空仓情况，止盈条件可写为当最低价比建仓价格低30跳，表达式为：

最低价<= 建仓价格 – 止盈值（30跳表示的点位）

Low<= MyEntryPrice - 30\* MinMove\*PriceScale

1. **止损20跳**
   * 多仓情况，止损条件可写为当最低价比建仓价格低20跳，表达式为：

最低价>= 建仓价格 - 止损值（20跳表示的点位）

Low <= MyEntryPrice - 20\* MinMove\*PriceScale

* + 空仓情况，止损条件可写为当最高价比建仓价格高20跳，表达式为：

最高价>= 建仓价格 + 止损值（20跳表示的点位）

high >= MyEntryPrice + 20\* MinMove\*PriceScale

1. 【说明】实际编写时，为了保证代码的可重复性，表达式中不直接使用数字30,20，而是定义数值型变量TakeProfitSet存放止盈设置，StopLossSet存放止损设置。这样当需要更改止盈止损的点位时，只需对变量重新赋值即可，代码中止盈止损的表达式则无需修改。
2. 如果止盈条件修改为价格上涨30%止盈，表达式可以写为：（以多仓为例） High>= MyEntryPrice + 0.3\* MyEntryPrice
3. **止盈止损的价格，考虑开盘价格跳空的情况**

止盈止损的价格，通常情况下直接按照止盈止损的要求即可，如果遇到开盘跳空的时候，需要进行相应的处理。

以多仓止盈，跳空为例，如果开盘价>止盈价，则使用开盘价进行止盈。

If(Open > MyExitPrice) MyExitPrice = Open

【说明】变量MyExitPrice也用作保存止盈价格

1. **公式主体结构设计**

由于止盈止损分为多仓和空仓两类不同的情况，所以采用嵌套的if……else语句

**代码：**

Vars

    Numeric MinPoint; *// 一个最小变动单位，也就是一跳*

    Numeric MyEntryPrice; *// 开仓价格，例中为开仓均价，可设置为某次入场价*

    Numeric **TakeProfitSet**(30); *// 止赢设置*

    Numeric **StopLossSet**(20); *// 止损设置*

    Numeric MyExitPrice; *// 平仓价格*

Events

**OnBar**(ArrayRef<Integer> indexs)

{

*//...*

    MinPoint = MinMove\*PriceScale;

    MyEntryPrice = AvgEntryPrice;

    If(MarketPosition == 1 And BarsSinceEntry >= 1) *// 有多仓的情况*

    {

        If(High >= MyEntryPrice + TakeProfitSet\*MinPoint) *// 止赢条件表达式*

        {

            MyExitPrice = MyEntryPrice + TakeProfitSet\*MinPoint;

*// 如果该Bar开盘价即跳空触发，用开盘价代替*

            If(Open > MyExitPrice) MyExitPrice = Open;

            Sell(0,MyExitPrice);

        }Else **If**(Low <= MyEntryPrice - StopLossSet\*MinPoint) *// 止损条件表达式*

        {

            MyExitPrice = MyEntryPrice - StopLossSet\*MinPoint;

*// 如果该Bar开盘价即跳空触发，则用开盘价代替*

            If(Open < MyExitPrice) MyExitPrice = Open;

            Sell(0,MyExitPrice);

        }

    }

    Else **If**(MarketPosition == -1 And BarsSinceEntry >= 1) *// 有空仓的情况*

    {

        If(Low <= MyEntryPrice - TakeProfitSet\*MinPoint) *// 止赢条件表达式*

        {

            MyExitPrice = MyEntryPrice - TakeProfitSet\*MinPoint;

            If(Open < MyExitPrice) MyExitPrice = Open;

            BuyToCover(0,MyExitPrice);

        }

        Else **If**(High >= MyEntryPrice + StopLossSet\*MinPoint) *// 止损条件表达式*

        {

            MyExitPrice = MyEntryPrice + StopLossSet\*MinPoint;

            If(Open > MyExitPrice) MyExitPrice = Open;

            BuyToCover(0,MyExitPrice);

        }

    }

*//...*

}

**【注意事项】**

1. 因无法确定开仓Bar最高价、最低价和开仓价的先后顺序，因此以上写法忽略开仓Bar的处理。
2. 如果某个Bar最高价、最低价相差很大，可能会出现止盈止损同时满足的情况，这种情况下需要切换到更小的周期进行交易，或者扩大止盈、止损幅度。

**案例二 跟踪止损**

跟踪止损有很多种方式，本模板的规则如下：当盈利达到50跳之后启动第一级跟踪止损，止损的回撤值为30跳；当盈利达到80跳之后启动第二级的跟踪止损，止损的回撤值为20跳。原始止损条件为回撤50跳。也可以将这些固定的设置修改为盈利百分比，或者是某个价格的百分比。

**分析（以多仓为例）**

1. **第一个建仓位置到当前位置的Bar计数用以下函数表示**

BarsSinceEntry

| **说明** | **获得当前持仓的第一个建仓位置到当前位置的Bar计数。** |
| --- | --- |
| 语法 | Integer BarsSinceEntry() |
| 参数 | 无 |
| 备注 | 获得当前持仓的第一个建仓位置到当前位置的Bar计数，返回值为整型。 只有当MarketPosition != 0时，即有持仓的状况下，该函数才有意义，否则返回0。 注意：在开仓Bar上为0。 |

记录建仓以来最高最低价格。跟踪止损条件是采用这个最高最低价格和止损点之间进行比较判断。以多仓为例，需要记录开仓以来最高价格，使用序列变量HighestAfterEntry保存此值，在开仓bar上，比较开仓价和最新价格，记录较大的值；其他bar上，比较该bar的high和HighestAfterEntry的值，记录较大值。空仓反之，记录开仓以来的最低价格。

If(BarsSinceEntry == 0)

    {

        HighestAfterEntry = Close;

        LowestAfterEntry = Close;

        If(MarketPosition <> 0)

        {

            HighestAfterEntry = Max(HighestAfterEntry,AvgEntryPrice);

            LowestAfterEntry = Min(LowestAfterEntry,AvgEntryPrice);

        }

    }Else

    {

        HighestAfterEntry = Max(HighestAfterEntry,High);

        LowestAfterEntry = Min(LowestAfterEntry,Low);

    }

1. **公式结构**

跟踪止损和直接止损的区别是随着盈利的提高，突破某个值时，相应止损回撤值减少，即止损点不是固定的。因此需要使用多分支的嵌套语句分别进行判断。

1. **止损价格需考虑开盘跳空的情况**

If(Open < MyExitPrice) MyExitPrice = Open;

【说明】MyExitPrice为止损价格

**代码：**

Vars

    Numeric MinPoint;          *// 一个最小变动单位，也就是一跳*

    Numeric MyEntryPrice;      *// 开仓价格，本例为开仓均价，可设置为某次入场价*

    Numeric **TrailingStart1**(50);   *// 跟踪止损启动设置1*

    Numeric **TrailingStart2**(80);   *// 跟踪止损启动设置2*

    Numeric **TrailingStop1**(30);   *// 跟踪止损设置1*

    Numeric **TrailingStop2**(20);   *// 跟踪止损设置2*

    Numeric **StopLossSet**(50);    *// 止损设置*

    Numeric MyExitPrice;        *// 平仓价格*

    Series<Numeric> HighestAfterEntry;        *// 开仓后出现的最高价*

    Series<Numeric> LowestAfterEntry;         *// 开仓后出现的最低价*

Events

**OnBar**(ArrayRef<Integer> indexs)

{

*//...*

    If(BarsSinceEntry == 0)                *// 条件满足：开仓Bar*

    {

        HighestAfterEntry = Close;

        LowestAfterEntry = Close;        *// 赋初值为当前最新价格*

        If(MarketPosition <> 0)            *// 有持仓时执行以下代码*

        {

*// 开仓Bar，将开仓价和当时的收盘价的较大值保留到HighestAfterEntry*

            HighestAfterEntry = Max(HighestAfterEntry,AvgEntryPrice);

*// 开仓Bar，将开仓价和当时的收盘价的较小值保留到LowestAfterEntry*

            LowestAfterEntry = Min(LowestAfterEntry,AvgEntryPrice);

        }

    }Else      *// 非开仓Bar时进行以下运算*

    {

*// 记录下当前Bar的最高点，用于下一个Bar的跟踪止损判断*

        HighestAfterEntry = Max(HighestAfterEntry,High);

*// 记录下当前Bar的最低点，用于下一个Bar的跟踪止损判断*

        LowestAfterEntry = Min(LowestAfterEntry,Low);

    }

    Commentary("HighestAfterEntry = "+Text(HighestAfterEntry));

    Commentary("LowestAfterEntry = "+Text(LowestAfterEntry));

    MinPoint = MinMove\*PriceScale;

    MyEntryPrice = AvgEntryPrice;

    If(MarketPosition == 1 And BarsSinceEntry >= 1)  *// 有多仓的情况*

    {

*// 第二级跟踪止损的条件表达式*

        If(HighestAfterEntry[1] >= MyEntryPrice + TrailingStart2\*MinPoint)

        {

            If(Low <= HighestAfterEntry[1] - TrailingStop2\*MinPoint)

            {

                MyExitPrice = HighestAfterEntry[1] - TrailingStop2\*MinPoint;*// 如果该Bar开盘价即跳空触发，则用开盘价代替*

                If(Open < MyExitPrice) MyExitPrice = Open;

                Sell(0,MyExitPrice);

            }

        }Else **If**(HighestAfterEntry[1] >= MyEntryPrice + TrailingStart1\*MinPoint) *// 第一级跟踪止损的条件表达式*

        {

            If(Low <= HighestAfterEntry[1] - TrailingStop1\*MinPoint)

            {

                MyExitPrice = HighestAfterEntry[1] - TrailingStop1\*MinPoint;

*// 如果该Bar开盘价即跳空触发，则用开盘价代替*

                If(Open < MyExitPrice) MyExitPrice = Open;

                Sell(0,MyExitPrice);

            }

        }Else **If**(Low <= MyEntryPrice - StopLossSet\*MinPoint) *//可在此写初始止损处理*

        {

            MyExitPrice = MyEntryPrice - StopLossSet\*MinPoint;

*// 如果该Bar开盘价即跳空触发，则用开盘价代替*

            If(Open < MyExitPrice) MyExitPrice = Open;

            Sell(0,MyExitPrice);

        }

    }Else **If**(MarketPosition == -1 And BarsSinceEntry >= 1)   *// 有空仓的情况*

    {

*// 第二级跟踪止损的条件表达式*

        If(LowestAfterEntry[1] <= MyEntryPrice - TrailingStart2\*MinPoint)

        {

            If(High >= LowestAfterEntry[1] + TrailingStop2\*MinPoint)

            {

                MyExitPrice = LowestAfterEntry[1] + TrailingStop2\*MinPoint;

                If(Open > MyExitPrice) MyExitPrice = Open;

                BuyToCover(0,MyExitPrice);

            }

        }Else **If**(LowestAfterEntry[1] <= MyEntryPrice - TrailingStart1\*MinPoint)*// 第一级跟踪止损的条件表达式*

        {

            If(High >= LowestAfterEntry[1] + TrailingStop1\*MinPoint)

            {

                MyExitPrice = LowestAfterEntry[1] + TrailingStop1\*MinPoint;

                If(Open > MyExitPrice) MyExitPrice = Open;

                BuyToCover(0,MyExitPrice);

            }

        }Else **If**(High >= MyEntryPrice + StopLossSet\*MinPoint) *//可在此写初始止损处理*

         {

            MyExitPrice = MyEntryPrice + StopLossSet\*MinPoint;

            If(Open > MyExitPrice) MyExitPrice = Open;

            BuyToCover(0,MyExitPrice);

        }

    }

*//...*

}

**【注意事项】**

* 因无法确认开仓Bar最高价、最低价和开仓价的先后顺序，因此以上写法一般忽略开仓Bar的处理。
* 如果某个Bar最高价、最低价相差很大，可能出现创新高之后跟踪止损的情况，但系统无法确认最高价和最低价的先后顺序，因此本模板只用前一个Bar的最高价、最低价计算最大盈利位置。

**案例三 加仓减仓**

本例仅以做多为例，做空类似。模板以首次开仓2手后每盈利30跳加仓一次，每次1手，最多加仓3次；开仓后每亏损30跳减仓1手。也可以转换为开仓价格的百分比值，或波动率的百分比等其它任何设置的变量进行处理。

**分析：**

1. **加仓三次如何表示，需要用到CurrentEntries函数**

CurrentEntries

| **说明** | **获得当前持仓的建仓次数。** |
| --- | --- |
| **语法** | Integer CurrentEntries() |
| **参数** | 无 |
| **备注** | 获得当前持仓的建仓次数，返回值为整型。 只有当MarketPosition != 0时，即有持仓的状况下，该函数才有意义，否则返回0。 |

通过判断条件CurrentEntries<4得知是否加仓三次

1. **条件设置，以盈利为例，每赢利30跳加仓一次**

最高价格>= 最后一次建仓价格 + 30跳盈利

High >= LastPrice + AddSet\*MinPoint

【说明】为了方便调整盈利设置，这里不直接使用常数30，而是用变量AddSet来保存。

1. **公式结构**

因为有仓位时需要一直判断是否满足加仓减仓的条件，所以要使用循环语句。总的来说是分支结构，其中嵌套循环语句。

**代码：**

Vars

    Numeric MinPoint;            *// 一个最小变动单位，也就是一跳*

    Series<Numeric> FirstPrice;    *// 第一次开仓价格*

    Series<Numeric> LastPrice;    *// 最后一次开仓价格*

    Numeric **AddSet**(30);         *// 加仓设置*

    Numeric **SubSet**(30);         *// 减仓设置*

    Bool FirstEntryCon;         *// 首次开仓条件*

Events

**OnBar**(ArrayRef<Integer> indexs)

{

*//FirstEntryCon = ...            // 设置首次开仓条件*

    MinPoint = MinMove\*PriceScale;

    If(MarketPosition == 0)        *// 空仓时*

    {

        If(FirstEntryCon)

        {

            FirstPrice = Open;

            LastPrice = FirstPrice;

            Buy(2,FirstPrice);        *// 条件满足，首次开仓2手*

        }

    }Else **If**(MarketPosition == 1 And BarsSinceEntry >= 1)    *// 有多仓的情况*

    {

        While(CurrentEntries < 4 && High >= LastPrice + AddSet\*MinPoint)     *// 加仓*

        {

            LastPrice = LastPrice + AddSet\*MinPoint;

            If(Open > LastPrice) LastPrice = Open;

            Buy(1,LastPrice);

        }

        While(CurrentEntries > 0 && Low <= FirstPrice - SubSet\*MinPoint)     *// 减仓*

        {

            FirstPrice = FirstPrice - SubSet\*MinPoint;

            If(Open < FirstPrice) FirstPrice = Open;

            Sell(1,FirstPrice);

        }

    }

*//...*

}

**【注意事项】**

1. 因无法确认开仓Bar最高价、最低价和开仓价的先后顺序，忽略开仓Bar的加减仓处理。
2. 如果某个Bar最高价、最低价相差很大，可能出现加仓减仓同时满足的情况，这种情况下需要切换到更小的周期进行交易，或者扩大加仓、减仓幅度设置。

**案例四 多品种交易**

多品种交易，指的是图表中叠加多个商品，公式中对多个商品同时进行交易。通常情况下，如果公式中没有指明交易的商品，而且也没有在商品交易属性中设置委托偏移，仅针对第一个商品Data0进行交易。

模板以常用的双均线系统为例，对叠加的各个商品分别进行交易。

**代码：**

Params

    Numeric **FastLength**(5);

    Numeric **SlowLength**(20);

Vars

    Series<Numeric> AvgValue1;

    Series<Numeric> AvgValue2;

Events

**OnBar**(ArrayRef<Integer> indexs)

{

    Range[0:1]

    {

        AvgValue1 = AverageFC(Close,FastLength);

        AvgValue2 = AverageFC(Close,SlowLength);

        PlotNumeric("MA1",AvgValue1);

        PlotNumeric("MA2",AvgValue2);

    }

    Range[0:1]

    {

        If(MarketPosition <>1 && AvgValue1[1] > AvgValue2[1])

        {

            Buy(1,Open);

        }

        If(MarketPosition <>-1 && AvgValue1[1] < AvgValue2[1])

        {

            SellShort(1,Open);

        }

    }

}

**案例五 跨周期**

案例实现在5分钟K线上调用日线指标。

**分析**

1. 本案例的核心在于如何得到日线指标，TB新公式系统支持同一策略单元内跨周期数据源调用。只需将5分钟周期数据源和日周期数据源加入同一策略单元，便可非常方便地在5分钟K线上调用日线指标。
2. 考虑引用日线指标时引用未来数据问题，对于当日来讲，日线指标还没有计算出来就要引用，这种方式写技术分析指标是可以的，但用来进行自动交易就会出问题，为了更准确合理的使用跨周期数据，日线指标最多引用到前一日。

**代码：**

新建一个公式应用My5MinMA，得到日线指标并显示，详细代码如下：

Params

    Numeric **Length**(10);

Vars

    Series<Numeric> MA;

events

**OnBar**(ArrayRef<Integer> indexs)

{

    Data[1].MA = Data[1].AverageFC(Data[1].Close,Length);*//计算日线指标*

    Data[0].PlotNumeric("MA", Data[1].MA[1]);    *//在5分钟周期中引用日线指标*

    Data[1].PlotNumeric("MAday", Data[1].MA[1]);    *//在日线上同时显示指标*

}

在超级图表中依次插入5分钟商品和日线商品，将编译成功的公式My5MinMA插入超级图表,为了对比方便，将支持多数据源显示的10周期均线公式MA\_Mlayer\_10 插入超级图表中，显示如下：

**【注意事项】**

为了更准确合理的使用跨周期数据，建议引用数据源上一个bar的数据或变量，以避免使用未来数据或变化的数据。

**案例六 收盘平仓**

**分析**

收盘平仓是指每个交易日收盘前平掉持有的仓位。由于真正收盘后，交易指令是无法成交的，而太早发出平仓指令，实际平仓价又会和真实的收盘价有一定的差异。因此，实际交易时只能通过在真正收盘之前，提前一定的时间，发出平仓指令，近似地达到收盘平仓的目标。因此，平仓信号分为两种情况，一种是（非当天）的收盘平仓信号，另一种是当天的收盘平仓信号。

1. **历史的收盘平仓信号：以该交易日的收盘价平仓（近似效果）**

历史收盘平仓信号，一定产生于过去某个交易日的最后一根Bar。所以，编程的主要任务是找到该交易日的最后一根bar，然后在此bar上执行平仓代码。这里又分为两种情况：

* + 非最后交易日的收盘平仓

找到两个bar日期都为有效值，并且前一根Bar日期与后一根bar的日期不同，由此可得出前一根Bar是那天的最后一个Bar，应该进行收盘平仓。

TradingDate[-1] != InvalidInteger && TradingDate!= TradingDate[-1]

* + 最后交易日（但并不是今日）的收盘平仓

既然是最后交易日，那收盘BAR一定是最后一根。因此，可通过后一个bar是否为无效值来判断。但是最后Bar的日期是小于系统当前日期的值，说明已经不是当天了，也要进行收盘平仓。（但要注意和下面一种情况的区别是，不是交易当天，所以不需要对时间进行判断）

TradingDate[-1] == InvalidInteger && TradingDate< CurrentDate

1. **当天的收盘平仓（选择14：59分平仓）**

当天的收盘平仓和历史的收盘平仓信号的不同之处在于，当天的收盘平仓是在接近收盘时才产生，所以需要增加一个时间的判断。以商品期货15:00收盘，5分钟周期为例，找到当日最后一个bar，即time为14:55的那个bar，再判断系统当前时间是否超过14:59分，满足条件之后做收盘平仓操作。

TradingDate== CurrentDate && Time == 0.1455 && CurrentTime >= 0.1459

**代码:**

Events

**OnBar**(ArrayRef<Integer> indexs)

{

*//...*

    If((TradingDate[-1] != InvalidInteger && TradingDate!= TradingDate[-1]) || (TradingDate[-1] == InvalidInteger && TradingDate< CurrentDate))

    {

        Sell(0,Close);

        BuyToCover(0,Close);

    }Else **If**(TradingDate== CurrentDate && Time == 0.1455 && CurrentTime >= 0.1459)

    {

        Sell(0, Close);

        BuyToCover(0, Close);

    }

*//...*

}

**【注意事项】**

1. 本例是以国内商品期货交易所收市时间举例，股指期货或其他市场需调整写法。
2. 本例是针对5分钟周期的收盘平仓所写，针对不同的周期需改写为合适的最后Bar时间。

**案例七 A函数下单、撤单以及全局变量操作**

案例实现的功能是每天收盘前N分钟时自动撤掉超级图表中多个商品的挂单，并全部平仓。代码中通过A\_SendOrder进行下单，A\_DeleteOrder进行撤单。

**分析**

1. A函数仅对实时行情有效，所以，需要使用之前用“BarStatus == 2”进行限定；
2. 根据TB程序运行机制，实时行情时，当前bar每个tick都会触发程序的执行，为了避免A函数每个tick重复撤单平仓，公式中使用全局变量HasSendOrder来保存上一次程序运行之后撤单平仓标志。
3. 公式中的平仓是根据A\_BuyPosition()的返回值来确定的，但是在收盘平仓之前有撤挂单的操作，或者因为成交回报不及时而不能得到准确的A\_BuyPosition()值，因此，设置撤单之后延时5个tick再平仓。（注意：这里假定撤单后5个Tick委托状态能同步成功，实际情况中因网络延时等原因并不一定能够成功，因此实际策略中请根据情况调整）

本例中用公式中使用全局变量DeleteOrderTickCounter来记录延时的tick数。当撤单执行后，DeleteOrderTickCounter赋值为1，开始计数，每次累加1，判断DeleteOrderTickCounter小于5，没有达到5个tick的时候直接return，直到5个tick之后再继续后续操作，平仓。

值得注意的是DeleteOrderTickCounter的初值如何赋值，代码中给出的是999，实际上只要是比5大的数都可以，这是为了保证到了收盘平仓时间，如果没有挂单可以直接进行平仓，即：使得条件DeleteOrderTickCounter< 5不成立，程序可执行平仓操作。

**代码：**

Params

    Numeric **offSet**(1);                *// 委托价格偏移*

    Numeric **BeforeMins**(10);           *// 收盘前几分钟开始操作*

Vars

    Numeric tempPos;                  *// 仓位*

    Global Numeric dataIndex;         *// 商品索引*

    Global Integer DeleteOrderTickCounter;

    Global Integer HasSendOrder;

Events

**OnBar**(ArrayRef<Integer> indexs)

{

    If(BarStatus == 0)    *// 第一个Bar，初始化Tick计数器、撤单标志，存于全局变量中*

    {

        DeleteOrderTickCounter = 9999;

        HasSendOrder = 0;

        SetGlobalVar(0,DeleteOrderTickCounter);

        SetGlobalVar(1,HasSendOrder);

    }Else         *// 其他Bar，从全局变量中读取撤单Tick计数器、撤单标志的值*

    {

        DeleteOrderTickCounter = GetGlobalVar(0);

        HasSendOrder = GetGlobalVar(1);

    }

*// 收盘前N分钟，且撤单标志为0，即还未撤单时*

    If(CurrentTime > (0.1459 - 0.0001\*(BeforeMins - 1)) && BarStatus == 2 && gValue[0] == 0)

    {

        For dataIndex=0 To DataSourceSize -1

        {

            If(Data[dataIndex].Close != InvalidNumeric && Data[dataIndex].A\_GetOpenOrderCount() > 0)

            {

                Data[dataIndex].A\_DeleteOrder();

*// Tick开始计数，为了延迟5个Tick后做平仓用的*

               DeleteOrderTickCounter = 1;

            }

            DeleteOrderTickCounter = DeleteOrderTickCounter + 1;

            SetGlobalVar(0,DeleteOrderTickCounter);

            If(DeleteOrderTickCounter < 5) Return; *// 撤单后需要延迟几个Tick才平仓*

            Data[dataIndex].tempPos = Data[dataIndex].A\_BuyPosition();

            If(Data[dataIndex].tempPos > 0) *// 平多单*

            {

              Data[dataIndex].A\_SendOrder(Enum\_Sell,Enum\_Exit, Data[dataIndex].tempPos, Data[dataIndex].Q\_BidPrice - offSet\* Data[dataIndex].MinMove\* Data[dataIndex].PriceScale);

            }

            Data[dataIndex].tempPos = Data[dataIndex].A\_SellPosition();

            If(Data[dataIndex].tempPos > 0) *// 平空单*

            {

              Data[dataIndex].A\_SendOrder(Enum\_Buy,Enum\_Exit, Data[dataIndex].tempPos, Data[dataIndex].Q\_AskPrice + offSet\* Data[dataIndex].MinMove\* Data[dataIndex].PriceScale);

            }

            HasSendOrder = 1;

            SetGlobalVar(1,HasSendOrder);

        }

    }

}

**【注意事项】**

1. 本例是以国内商品期货交易所收市时间举例，股指期货或其他市场需调整写法。
2. 本例假设撤单后5个Tick委托状态能同步成功，实际情况中因网络延时等原因并不一定能够成功。

**案例八 平仓延迟反手**

本案例实现平仓之后，再反手开仓。

**分析**

使用buy、sellshort这类反手交易指令建仓时，如果持有反向仓位，将会先平再开。实际交易过程中，平仓和开仓两个指令会同时发出，交易所不一定先撮合成交哪个指令，所以一般使用这种反手交易指令时，要求客户的账户上有交易数量2倍的资金。如果客户资金有限，希望先执行平仓操作，资金返回账户之后再反手开仓，那么编写公式时需要注意将平仓和开仓操作分开，保证平仓成交之后再进行开仓。本例实现的思路是先平仓，平仓之后延时N个tick再开仓。

实现方法：（**以平空仓反手开多为例**）

1. 定义参数DelayTicks，保存延时的tick数；
2. 定义变量TickCounter记录tick数，初始值为0，且最新Bar第一次生成时，重新开始计数；
3. 持空仓时如果需要平仓开多，不直接采用buy，而是先执行BuyToCover平仓，同时TickCounter开始tick计数；
4. 比较TickCounter与DelayTicks，达到了延时时间，再执行buy开仓。

**代码：（只应用在单数据源）**

Params

    Numeric **FastLength**(5);

    Numeric **SlowLength**(20);

    Numeric **DelayTicks**(5);

Vars

    Series<Numeric> AvgValue1;

    Series<Numeric> AvgValue2;

    Numeric LastBarTime;

    Numeric TickCounter;

    Numeric dataIndex;

Events

**OnBar**(ArrayRef<Integer> indexs)

{

    AvgValue1 = AverageFC(Close,FastLength);

    AvgValue2 = AverageFC(Close,SlowLength);

    LastBarTime = GetGlobalVar(0);

    TickCounter = GetGlobalVar(1);

*// 最新Bar第一次生成时，Tick重新开始计数*

    If(BarStatus == 2 && gValue[0]!= Time)

    {

        LastBarTime = Time;

        TickCounter = 0;

    }

    If(MarketPosition <> 1 && AvgValue1[1] > AvgValue2[1])

    {

        If(MarketPosition == 0 || BarStatus != 2)

*// 无持仓，直接买多仓*

*// 持空仓且Bar不是实时行情，平空仓，买多仓*

        {

            Buy(1,Open);

        }Else    *// 持空仓，Bar实时行情，平空仓，通过TickCounter计数，延迟反手*

        {

            BuyToCover(1,Open);

            If(TickCounter == 0)

            {

               TickCounter = 1;

            }Else **If**(TickCounter < DelayTicks)

            {

               TickCounter = TickCounter + 1;

            }Else

            {

                Buy(1,Open);

            }

        }

    }

    If(MarketPosition <> -1 && AvgValue1[1] < AvgValue2[1])

    {

        If(MarketPosition == 0 || BarStatus != 2)

        {

            SellShort(1,Open);

        }Else     *// 持多仓且Bar为实时行情，平多，延迟反手*

        {

            Sell(1,Open);

            If(TickCounter == 0)

            {

                TickCounter = 1;

            }Else **If**(TickCounter < DelayTicks)

            {

                TickCounter = TickCounter + 1;

            }Else

            {

                SellShort(1,Open);

            }

        }

    }

    SetGlobalVar(0,LastBarTime);

    SetGlobalVar(1,TickCounter);

}

**【注意事项】**

1. TB程序每个tick触发，所以可以实现延时几个tick在反手开仓，但是用户要考虑gValue[1]重置为0的时机，本例采用的是新bar生成的时候重置，这样设置对于一些长线周期没问题，但是对于短周期（tick级，秒级），会出现延时tick还未达到的，新bar生成时将gValue[1]的情况。
2. 平仓延迟反手除了延时的办法，还可以直接判断平仓是否成交的方式，用户可根据实际情况选择。