explain_code

August 17, 2023

1 Công cụ hỗ trợ giao dịch với tín hiệu 2 đường EMA giao cắt

[ ]:
```
//+------------------------------------------------------------------+
//|                                              Bot_Click_To_Trade.mq5 |
//|                                  Copyright 2023, Vuong Pham. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2023, Vuong Pham."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

2 Thông số đầu vào

[ ]:
```
input int fMA_inp = 8;  // fast EMA
input int sMA_inp = 14; // slow EMA

input ENUM_TIMEFRAMES tframe_trade   = PERIOD_M1;   // khung thoi gian trade
input ENUM_TIMEFRAMES tframe_base_01 = PERIOD_M5;   // khung thoi gian co so 1
input ENUM_TIMEFRAMES tframe_base_02 = PERIOD_M30;  // khung thoi gian co so 2

input bool alert        = true; // cho phep alert
input bool all_signal   = true; // tat ca signal
input bool good_signal  = true; // signal dong pha 2 chart
input bool best_signal  = true; // signal dong pha 3 chart
input bool notification = true; // cho phep notification

input double sl_percent = 0.5;   // phan tram stoploss (%)
input int sl_point_gap  = 10;    // KC cong them vao dinh/day (point)

input bool is_break_even = true; // tu dong keo hoa von
input double ratio_rr    = 0.8;  // ty le (reward / risk) bat dau keo sl
```

2.1 Giải thích:

- Hai EMA lấy thông số chu kỳ 8 và 14. Vì sao chọn 2 EMA này?
  - Vì backtest chiến lược 2 EMA cắt nhau cho thấy thông số này tốt

  - Khoảng thời gian chu kỳ 2 EMA không lệch nhau lớn, để có tín hiệu giao cắt sớm nhất
  - Khoảng thời gian chu kỳ vừa đủ nhỏ, để đường EMA bám sát giá, phản ánh nhanh, nhưng không quá nhiễu loạn
  - Cặp số 8 & 14 cũng sát với 8 & 13 là những con số fibonacci
- Khung thời gian trade, khung thời gian cơ sở 1 và 2.
  - Các khung thời gian nên cách nhau là bội của 4, 5 or 6 lần.
  - Ví dụ: M1-M5-M30, M5-M30-H4, H1-H4-D1
- Cho phép mở hộp thoại Alert trên MT5
- Nếu cho phép alert, có 3 chế độ như sau:
  - CĐ1. Báo tất cả các tín hiệu xảy ra ở cả 3 khưng thời gian
  - CĐ2. Báo tín hiệu khi có sự đồng pha giữa khung trade và khung cơ sở 1
  - CĐ3. Báo tín hiệu khi có sự đồng pha giữa khung trade, khung cơ sở 1 và 2
- Cho phép báo notification tới MT5 trên điện thoại:
  - Trường hợp này chỉ post những tín hiệu ở chế độ 2 và 3.
- Phần trăm stoploss khi vào một lệnh, mặc định 0.5% tài khoản mỗi lệnh.
  - Có thể vào nhiều lệnh, nhồi lệnh thuận xu hướng
- Bot sẽ tự động tính giá stoploss và nới rộng thêm một khoảng là con số nhập thêm (point)
- Chế độ tự động kéo SL về hòa vốn, khi đạt được mức tỷ lệ nhất định
  - Tỷ lệ mặc định là Reward : Risk = 0.8, đưa lệnh về an toàn sớm

3 Class xử lý tín hiệu giao cắt từ 2 EMA

[ ]:
```
//+------------------------------------------------------------------+
// CLASS TWO EMA CROSS
//+------------------------------------------------------------------+

enum status_chart
{
    _down_    = -1,
    _sideway_ = 0,
    _up_      = 1,

};

class Two_EMA
{
private:
    int size, fhandle, shandle;
    double fMA[], sMA[];
    int f, s;
    double close_01, close_02;
    ENUM_TIMEFRAMES tframe;

    string name_label, text;
    int x, y, font_size;
    color clr;
    ENUM_BASE_CORNER corner;
```

```
public:
    // constructor
    Two_EMA(ENUM_TIMEFRAMES);

    // method
    status_chart is_up_down();
    status_chart is_cross_up_down();
    status_chart is_cross();

    void alert_up_down();
    void alert_cross();
    void draw_status();
    void post_telegram();
};
```

3.1 Giải thích:

3.1.1 Ba trạng thái của thị trường suy ra từ vị trí tương đối của giá so với 2 EMA

- _down_: xu thế giảm, giá close nằm dưới cả EMA
- _sideway_: đi ngang, giá close nằm giữa 2 EMA
- _up_: xu thế tăng, giá close nằm trên cả 2 EMA

3.1.2 Thuộc tính

- Các biến sử dụng để xử lý handle của indicator iMA và copy_buffer
- Các biến sử dụng để vẽ label trạng thái thị trường lên chart

3.1.3 Các phương thức

- is_up_down(): trả về trạng thái giá đang close trên, dưới hay giữa 2 EMA
- is_cross_up_down(): trả về tín hiệu khi có sự chuyển trạng thái của giá so với 2 EMA
- is_cross(): trả về tín hiệu 2 EMA giao cắt nhau lên hay xuống
- alert_up_down(): báo alert khi có tín hiệu chuyển trạng thái của giá so với 2 EMA
- alert_cross(): báo alert khi có tín hiệu 2 EMA giao cắt nhau
- draw_status(): vẽ trạng thái của một khung thời gian ra chart
- post_telegram(): post tín hiệu qua telegram

3.2 Hàm khởi tạo giá trị cho class

[ ]:
```
// constructor
Two_EMA :: Two_EMA(ENUM_TIMEFRAMES _tframe)
{
    size = 0; fhandle = 0; shandle = 0;
    f = 0; s = 0;
    close_01 = 0; close_02 = 0;
    tframe = _tframe;
```

```
    text = "";
    corner = CORNER_RIGHT_UPPER;
    if (tframe == tframe_trade)
    {
        name_label = "tframe_trade";
        x = 220; y = 40;
        font_size = 12;
    }
    else if (tframe == tframe_base_01)
    {
        name_label = "tframe_base_01";
        x = 220; y = 70;
        font_size = 14;
    }
    else if (tframe == tframe_base_02)
    {
        name_label = "tframe_base_02";
        x = 220; y = 100;
        font_size = 14;
    }

    // draw on chart
    draw_status();
}
```

3.2.1 Giải thích:

- Ứng với mỗi khung thời gian (trade, cơ sở 1 và 2) thì sẽ có tọa độ (x, y) tương ứng để vẽ trên chart
  - font_size: kích thước chữ trong text
- Hàm draw_status() được thực thi khi khởi tạo instance

3.3 Phương thức `is_up_down()`

[ ]:
```
// method
status_chart Two_EMA :: is_up_down()
{
    size = 1;
    fhandle = iMA(_Symbol, tframe, fMA_inp, 0, MODE_EMA, PRICE_CLOSE);
    shandle = iMA(_Symbol, tframe, sMA_inp, 0, MODE_EMA, PRICE_CLOSE);

    f = CopyBuffer(fhandle, 0, 1, size, fMA);
    s = CopyBuffer(shandle, 0, 1, size, sMA);

    close_01 = iClose(_Symbol, tframe, 1);
```

```
    if (close_01 > fMA[0] && close_01 > sMA[0])
    {
        return _up_;
    }

    if (close_01 < fMA[0] && close_01 < sMA[0])
    {
        return _down_;
    }

    return _sideway_;
}
```

3.3.1 Giải thích:

- Lấy giá trị close của nến gần nhất, đồng thời lấy giá trị của fast EMA, slow EMA gần nhất
- So sánh với giá trị tương ứng giữa chúng
  - Nếu close nằm trên cả 2 EMA thì trả về _up_
  - Nếu close nằm dưới cả 2 EMA thì trả về _down_
  - Mặc định trả về _sideway_

3.4 Phương thức `is_cross_up_down()`

[ ]:
```
status_chart Two_EMA :: is_cross_up_down()
{
    size = 2;
    fhandle = iMA(_Symbol, tframe, fMA_inp, 0, MODE_EMA, PRICE_CLOSE);
    shandle = iMA(_Symbol, tframe, sMA_inp, 0, MODE_EMA, PRICE_CLOSE);

    f = CopyBuffer(fhandle, 0, 1, size, fMA);
    s = CopyBuffer(shandle, 0, 1, size, sMA);

    close_01 = iClose(_Symbol, tframe, 1);
    close_02 = iClose(_Symbol, tframe, 2);

    if (close_02 < sMA[0] && close_01 > fMA[1] && close_01 > sMA[1])
    {
        return _up_;
    }

    if (close_02 > sMA[0] && close_01 < fMA[1] && close_01 < sMA[1])
    {
        return _down_;
    }

    return _sideway_;
}
```

3.4.1 Giải thích:

- Lấy 2 giá trị close gần nhất, và 2 giá trị của 2 EMA gần nhất, so sánh giá trị tại index tương ứng
  - Trả về _up_ khi giá chuyển sang trạng thái nằm trên 2 EMA
  - Trả về _down_ khi giá chuyển sang trạng thái nằm dưới 2 EMA
  - Mặc định trả về _sideway_

3.5 Phương thức `is_cross()`

[ ]:
```
status_chart Two_EMA :: is_cross()
{
    size = 2;
    fhandle = iMA(_Symbol, tframe, fMA_inp, 0, MODE_EMA, PRICE_CLOSE);
    shandle = iMA(_Symbol, tframe, sMA_inp, 0, MODE_EMA, PRICE_CLOSE);

    f = CopyBuffer(fhandle, 0, 1, size, fMA);
    s = CopyBuffer(shandle, 0, 1, size, sMA);

    if (fMA[0] < sMA[0] && fMA[1] > sMA[1])
    {
        return _up_;
    }

    else if (fMA[0] > sMA[0] && fMA[1] < sMA[1])
    {
        return _down_;
    }

    return _sideway_;
}
```

3.5.1 Giải thích:

- Lấy 2 giá trị gần nhất của 2 EMA, so sánh vị trí tương ứng giữa chúng
- Trả về trạng thái giao cắt _up_ hay _down_ giữa 2 EMA
- Mặc định _sideway_ tức chưa có tín hiệu giao cắt

3.6 Phương thức `alert_up_down()`

[ ]:
```
void Two_EMA :: alert_up_down()
{
    if (is_cross_up_down() == _up_)
    {
        Alert(StringFormat("%s >>> %s: close Higher ... ", _Symbol,␣
 ↪EnumToString(tframe)));
    }
    else if (is_cross_up_down() == _down_)
```

```
    {
        Alert(StringFormat("%s >>> %s: close Lower ... ", _Symbol,␣
 ↪EnumToString(tframe)));
    }
}
```

3.6.1 Giải thích:

- Nếu vừa có tín hiệu chuyển trạng thái của giá so với 2 EMA thì báo alert tương ứng

3.7 Phương thức `alert_cross()`

[ ]:
```
void Two_EMA :: alert_cross()
{
    if (is_cross() == _up_)
    {
        Alert(StringFormat("%s >>> %s: Up cross ... ", _Symbol,␣
 ↪EnumToString(tframe)));
    }
    else if (is_cross() == _down_)
    {
        Alert(StringFormat("%s >>> %s: Down cross ... ", _Symbol,␣
 ↪EnumToString(tframe)));
    }
}
```

3.7.1 Giải thích:

- Nếu vừa có tín hiệu giao cắt của 2 EMA thì báo alert tương ứng

3.8 Phương thức `draw_status()`

[ ]:
```
void Two_EMA :: draw_status()
{
    if (is_up_down() == _up_)
    {
        text = StringFormat("%s: _UP_", EnumToString(tframe));
        clr = clrBlue;
    }
    else if (is_up_down() == _down_)
    {
        text = StringFormat("%s: _DOWN_", EnumToString(tframe));
        clr = clrRed;
    }
    else if (is_up_down() == _sideway_)
    {
        text = StringFormat("%s: _SW_", EnumToString(tframe));
```

```
        clr = clrGray;
    }

    ObjectDelete(0, name_label);
    ObjectCreate(0, name_label, OBJ_LABEL, 0, 0, 0);
    ObjectSetInteger(0, name_label, OBJPROP_CORNER, corner);
    ObjectSetInteger(0, name_label, OBJPROP_XDISTANCE, x);
    ObjectSetInteger(0, name_label, OBJPROP_YDISTANCE, y);
    ObjectSetString(0, name_label, OBJPROP_TEXT, text);
    ObjectSetInteger(0, name_label, OBJPROP_FONTSIZE, font_size);
    ObjectSetInteger(0, name_label, OBJPROP_COLOR, clr);
}
```

3.8.1 Giải thích:

- Vẽ trạng thái của gía so với 2 EMA tùy mỗi khung thời gian lên chart.
  - Trạng thái tăng _UP_ vẽ màu xanh
  - Trạng thái giảm _DOWN_ vẽ màu đỏ
  - Trạng thái sideway _SW_ vẽ màu xám

4 Class xử lý thời gian nến

[ ]:
```
//+------------------------------------------------------------------+
// CLASS CHECK NEW CANDLE
//+------------------------------------------------------------------+

class New_Candle
{
private:
    ENUM_TIMEFRAMES tframe;
    datetime tcandle;

public:
    // constructor
    New_Candle(ENUM_TIMEFRAMES);

    // method
    bool is_new_candle();
};
```

4.0.1 Giải thích:

- Tạo instance đại diện cho mỗi khung thời gian và kiểm tra tín hiệu sang nến mới

4.1 Hàm khởi tạo

[ ]:
```
// constructor
New_Candle :: New_Candle(ENUM_TIMEFRAMES _tframe)
{
    tframe = _tframe;
    tcandle = iTime(_Symbol, _tframe, 0);
}
```

4.1.1 Giải thích:

- Tạo instance đại diện cho mỗi khung thời gian
- Khởi tạo giá trị cho biến thời gian tcandle

4.2 Phương thức `is_new_candle()`

[ ]:
```
// method
bool New_Candle :: is_new_candle()
{
    if (tcandle == iTime(_Symbol, tframe, 1))
    {
        tcandle = iTime(_Symbol, tframe, 0);
        return true;
    }
    return false;
}
```

4.2.1 Giải thích:

- Kiểm tra tín hiệu sang nến mới hay chưa?

5 Class xử lý liên quan đến Button

[ ]:
```
//+------------------------------------------------------------------+
// CLASS BUTTON
//+------------------------------------------------------------------+

#include <Trade\Trade.mqh>
#include <Trade\PositionInfo.mqh>
CTrade trade;
CPositionInfo position;

enum button_type
{
    _close_,
    _sell_,
    _buy_,
```

```
    _dollar_sl_,
    _points_,
    _vol_,
};

struct request
{
    double sl;
    double vol;
};

class Button
{
private:
    string name_button, text;
    int x, y, font_size;
    color clr;
    ENUM_BASE_CORNER corner;
    int width, height;
    button_type type;
    request req;
    double points, balance, dollar_sl;

public:
    // constructor
    Button(button_type);

    // method
    void create_button();
    void get_stoploss_volume();
    void execution();
    void update_button_info();
};
```

5.1 Giải thích:

- Sử dụng thêm 2 class có sẵn CTrade và CPositionInfo
- Tạo Enum các loại button và Struct chứa stoploss và volume của lệnh

5.1.1 Thuộc tính

- Các biến liên quan đến vẽ button lên chart
- Các biến để xử lý tính toán stoploss khi giao dịch

5.1.2 Phương thức

- create_button(): tạo nút button trên chart

- get_stoploss_volume(): tinh toán giá trị stoploss và volume tương ứng tại mỗi thời điểm dựa trên khung thời gian trade
- execution(): thực thi giao dịch
- update_button_info(): cập nhật thông tin stoploss theo dollar, stoploss theo points, và mức volume gợi ý

5.2 Hàm khởi tạo

[ ]:
```
// constructor
Button :: Button(button_type _type)
{
    corner = CORNER_RIGHT_UPPER;
    font_size = 16;
    width = 100; height = 50;
    type = _type;

    req.sl = 0;
    req.vol = 0.01;

    points = 0;
    balance = AccountInfoDouble(ACCOUNT_BALANCE);
    dollar_sl = balance * sl_percent / 100.0;

    if (_type == _close_)
    {
        name_button = "_close_";
        x = 220; y = 180;
        text = "CLOSE";
        clr = clrGray;
    }
    else if (_type == _sell_)
    {
        name_button = "_sell_";
        x = 220; y = 250;
        text = "SELL";
        clr = clrRed;
    }
    else if (_type == _buy_)
    {
        name_button = "_buy_";
        x = 220; y = 320;
        text = "BUY";
        clr = clrBlue;
    }
    else if (_type == _dollar_sl_)
    {
        name_button = "_dollar_sl_";
```

```
        x = 110; y = 180;
        text = StringFormat("sl: $%0.1f", dollar_sl);
        clr = clrGray;
        font_size = 12;
    }
    else if (_type == _points_)
    {
        name_button = "_points_";
        x = 110; y = 250;
        text = StringFormat("points: %0.0f", points);
        clr = clrGray;
        font_size = 12;
    }
    else if (_type == _vol_)
    {
        name_button = "_vol_";
        x = 110; y = 320;
        text = StringFormat("vol: %0.2f", req.vol);
        clr = clrGray;
        font_size = 12;
    }

    // create button
    create_button();
}
```

5.2.1 Giải thích:

- Tương ứng với mỗi loại button sẽ có tọa độ (x, y), text, màu sắc và font_size tương ứng.
- Hàm tạo nút create_button() được chạy khi khởi tạo

5.3 Phương thức `create_button()`

[ ]:
```
// method
void Button :: create_button()
{
    ObjectDelete(0, name_button);
    ObjectCreate(0, name_button, OBJ_BUTTON, 0, 0, 0);
    ObjectSetInteger(0, name_button, OBJPROP_CORNER, corner);
    ObjectSetInteger(0, name_button, OBJPROP_XDISTANCE, x);
    ObjectSetInteger(0, name_button, OBJPROP_YDISTANCE, y);
    ObjectSetInteger(0, name_button, OBJPROP_XSIZE, width);
    ObjectSetInteger(0, name_button, OBJPROP_YSIZE, height);
    ObjectSetString(0, name_button, OBJPROP_TEXT, text);
    ObjectSetInteger(0, name_button, OBJPROP_FONTSIZE, font_size);
    ObjectSetInteger(0, name_button, OBJPROP_COLOR, clr);
    ObjectSetInteger(0, name_button, OBJPROP_STATE, false);
```

```
    ObjectSetInteger(0, name_button, OBJPROP_ZORDER, 0);
}
```

5.3.1 Giải thích:

- Tạo ra nút button tương ứng với chức năng

5.4 Phương thức `get_stoploss_volume()`

[ ]:
```
void Button :: get_stoploss_volume()
{
    Two_EMA _tf_trade(tframe_trade);
    double current_ask = SymbolInfoDouble(_Symbol, SYMBOL_ASK);
    double current_bid = SymbolInfoDouble(_Symbol, SYMBOL_BID);

    if (_tf_trade.is_up_down() == _up_)
    {
        int idx = iLowest(_Symbol, tframe_trade, MODE_LOW, 26, 0);
        req.sl = iLow(_Symbol, tframe_trade, idx) - sl_point_gap * _Point;

        points = (current_ask - req.sl) / _Point;
        req.vol = dollar_sl / points;
        req.vol = NormalizeDouble(req.vol, 2);
    }
    else if (_tf_trade.is_up_down() == _down_)
    {
        int idx = iHighest(_Symbol, tframe_trade, MODE_HIGH, 26, 0);
        req.sl = iHigh(_Symbol, tframe_trade, idx) + sl_point_gap * _Point;

        points = (req.sl - current_bid) / _Point;
        req.vol = dollar_sl / points;
        req.vol = NormalizeDouble(req.vol, 2);
    }
}
```

5.4.1 Giải thích:

- Tại khung thời gian trade, lấy trạng thái của giá close với với 2 EMA tương ứng
  - Nếu trạng thái là _up_, tính toán giá trị stoploss cho lệnh BUY, tìm giá Lowest trong vòng 26 thanh nến, và trừ đi khoảng cách sl_point_gap bù thêm
  - Nếu trạng thái là _down_, tính toán giá trị stoploss cho lệnh SELL, tìm giá Highest trong vòng 26 thanh nến, và cộng thêm khoảng cách sl_point_gap bù
- Dựa trên giá stoploss dự tính, tính toán ra volume tương ứng với mức loss theo phần trăm tài khoản

5.5 Phương thức `execution()`

[ ]:
```
void Button :: execution()
{
    double current_ask = SymbolInfoDouble(_Symbol, SYMBOL_ASK);
    double current_bid = SymbolInfoDouble(_Symbol, SYMBOL_BID);
    get_stoploss_volume();

    if (type == _buy_)
    {
        trade.Buy(req.vol, _Symbol, current_ask, req.sl);
    }
    else if (type == _sell_)
    {
        trade.Sell(req.vol, _Symbol, current_bid, req.sl);
    }
    else if (type == _close_)
    {
        int total = PositionsTotal();
        for(int i = 0; i < total; i++)
        {
            if (position.SelectByIndex(i))
            {
                trade.PositionClose(position.Ticket());
            }
        }
    }

    // release state button
    ObjectSetInteger(0, name_button, OBJPROP_STATE, false);
}
```

5.5.1 Giải thích:

- Thực thi giao dịch khi có tín hiệu Click chuột vào nút tương ứng với chức năng
  - nút _buy_ vào lệnh BUY
  - nút _sell_ vào lệnh SELL
  - nút _close_ để đóng tất cả lệnh trên cặp tiền đang chạy
- Giải phóng trạng thái ấn nút sau khi thực thi xong

5.6 Phương thức `update_button_info()`

[ ]:
```
void Button :: update_button_info()
{
    get_stoploss_volume();

    if (type == _dollar_sl_)
```

```
    {
        text = StringFormat("sl: $%0.1f", dollar_sl);
        ObjectSetString(0, name_button, OBJPROP_TEXT, text);
    }
    else if (type == _points_)
    {
        text = StringFormat("points: %0.0f", points);
        ObjectSetString(0, name_button, OBJPROP_TEXT, text);
    }
    else if (type == _vol_)
    {
        text = StringFormat("vol: %0.2f", req.vol);
        ObjectSetString(0, name_button, OBJPROP_TEXT, text);
    }

    // release state button
    ObjectSetInteger(0, name_button, OBJPROP_STATE, false);
}
```

5.6.1 Giải thích:

- Gọi hàm tính toán giá trị stoploss và volume
- Cập nhật text hiển thị lên chart, nhằm mục đích tham khảo
- Giải phóng trạng thái ấn nút

6 Khởi tạo tất cả Instance sử dụng trong công cụ

[ ]:
```
//+------------------------------------------------------------------+
// CREATE ALL INSTANCE
//+------------------------------------------------------------------+

Two_EMA tf_trade(tframe_trade);
Two_EMA tf_base01(tframe_base_01);
Two_EMA tf_base02(tframe_base_02);

New_Candle time_trade(tframe_trade);
New_Candle time_base01(tframe_base_01);
New_Candle time_base02(tframe_base_02);

Button btn_close(_close_);
Button btn_sell(_sell_);
Button btn_buy(_buy_);
Button btn_dollar_sl(_dollar_sl_);
Button btn_points(_points_);
Button btn_vol(_vol_);
```

6.0.1 Giải thích:

- Ba instance cho 3 khung thời gian (trade, cở sở 1 và 2)
- Ba instance đại diện cho xử lý thời gian nến ở 3 khung tương ứng
- Sáu instance cho 6 nút button trên chart

7 Hàm báo alert tín hiệu khi đồng pha 2 khung thời gian

[ ]:
```
//+------------------------------------------------------------------+
// ALERT GOOD SIGNAL
//+------------------------------------------------------------------+

string msg_up   = StringFormat("%s >>> %s: Up cross ... ", _Symbol,␣
 ↪EnumToString(tframe_trade));
string msg_down = StringFormat("%s >>> %s: Down cross ... ", _Symbol,␣
 ↪EnumToString(tframe_trade));

void alert_good_signal()
{
    if (tf_trade.is_cross() == _up_ && tf_base01.is_up_down() == _up_)
    {
        Alert(msg_up);
    }
    else if (tf_trade.is_cross() == _down_ && tf_base01.is_up_down() == _down_)
    {
        Alert(msg_down);
    }
}
```

7.0.1 Giải thích:

- Báo alert khi có sự giao cắt của 2 EMA trên khung thời gian trade, với điều kiện tại khung thời gian cơ sở 1 có sự đồng pha tương ứng với tín hiệu UP or DOWN

8 Hàm báo alert tín hiệu khi đồng pha 3 khung thời gian

[ ]:
```
//+------------------------------------------------------------------+
// ALERT BEST SIGNAL
//+------------------------------------------------------------------+

void alert_best_signal()
{
    if (tf_trade.is_cross() == _up_ && tf_base01.is_up_down() == _up_ &&␣
 ↪tf_base02.is_up_down() == _up_)
    {
        Alert(msg_up);
```

```
    }
    else if (tf_trade.is_cross() == _down_ && tf_base01.is_up_down() == _down_␣
 ↪&& tf_base02.is_up_down() == _down_)
    {
        Alert(msg_down);
    }
}
```

8.0.1 Giải thích:

- Báo alert khi có sự giao cắt của 2 EMA trên khung thời gian trade, với điều kiện tại cả khung thời gian cơ sở 1 và 2 có sự đồng pha tương ứng với tín hiệu UP or DOWN

9 Hàm notification tới MT5 trên điện thoại

[ ]:
```
//+------------------------------------------------------------------+
// NOTIFICATION
//+------------------------------------------------------------------+

void post_notification()
{
    if (tf_trade.is_cross() == _up_ && tf_base01.is_up_down() == _up_)
    {
        SendNotification(msg_up);
    }
    else if (tf_trade.is_cross() == _down_ && tf_base01.is_up_down() == _down_)
    {
        SendNotification(msg_down);
    }
}
```

9.0.1 Giải thích:

- Chỉ notification khi có sự đồng pha tín hiệu ở khung thời gian trade và cơ sở 1

10 Hàm kéo stoploss về hòa vốn

[ ]:
```
//+------------------------------------------------------------------+
// TAKE BREAK_EVEN (SL = OPEN)
//+------------------------------------------------------------------+

int _total;
double order_open, order_sl, price_current, new_sl;
ulong order_ticket;
long _order_type;
```

```
void take_break_even()
{
    _total = PositionsTotal();
    for (int i = 0; i < _total; i++)
    {
        if (PositionGetSymbol(i) == _Symbol)
        {
            price_current = PositionGetDouble(POSITION_PRICE_CURRENT);
            order_open = PositionGetDouble(POSITION_PRICE_OPEN);
            order_sl = PositionGetDouble(POSITION_SL);
            _order_type = PositionGetInteger(POSITION_TYPE);

            if ((_order_type == POSITION_TYPE_BUY && order_open > order_sl) ||
                (_order_type == POSITION_TYPE_SELL && order_open < order_sl))
            {
                if (MathAbs(price_current - order_open) / MathAbs(order_sl -␣
 ↪order_open) >= ratio_rr)
                {
                    order_ticket = PositionGetInteger(POSITION_TICKET);

                    if (_order_type == POSITION_TYPE_BUY)
                    {
                        new_sl = order_open + sl_point_gap * _Point / 3;
                    }
                    else if (_order_type == POSITION_TYPE_SELL)
                    {
                        new_sl = order_open - sl_point_gap * _Point / 3;
                    }

                    bool m = trade.PositionModify(order_ticket, new_sl, 0);
                }
            }
        }
    }
}
```

10.0.1 Giải thích:

- Liên tục kiểm tra xem mỗi vị thế đang mở đã được kéo stoploss hòa vốn hay chưa?
  - Nếu đã kéo hòa vốn thì bỏ qua
  - Nếu chưa kéo hòa vốn thì kiểm tra khoảng cách giá đã đi xa hơn so với tỷ lệ Reward : Risk hay chưa?
  - Thỏa điều kiện thì thực thi new_sl và modify

11 Hàm thực thi trên mỗi Tick của giá

[ ]:
```
//+------------------------------------------------------------------+
// RUNNING EVERY TICK
//+------------------------------------------------------------------+

void OnTick()
{
    if (time_base01.is_new_candle())
    {
        if (alert)
        {
            if (all_signal)
            {
                tf_base01.alert_up_down();
                tf_base01.alert_cross();
            }
            else if (good_signal)
            {
                tf_base01.alert_cross();
            }
        }
        tf_base01.draw_status();
    }

    if (time_base02.is_new_candle())
    {
        if (alert)
        {
            if (all_signal)
            {
                tf_base02.alert_up_down();
                tf_base02.alert_cross();
            }
            else if (good_signal)
            {
                tf_base02.alert_cross();
            }
        }
        tf_base02.draw_status();
    }

    if (time_trade.is_new_candle())
    {
        if (alert)
        {
            if (all_signal)
```

```
            {
                tf_trade.alert_cross();
            }
            else if (good_signal)
            {
                alert_good_signal();
            }
            else if (best_signal)
            {
                alert_best_signal();
            }
        }

        if (notification)
        {
            if (good_signal || best_signal)
            {
                post_notification();
            }
        }

        tf_trade.draw_status();

        btn_dollar_sl.update_button_info();
        btn_points.update_button_info();
        btn_vol.update_button_info();
    }

    if (is_break_even)
    {
        take_break_even();
    }
}
```

11.0.1 Giải thích:

- Tương ứng với instance mối khung thời gian (trade, cơ sở 1 và 2), kiểm tra xem đã sang nến mới hay chưa?
  - Nếu vừa sang nến mới, thì tương ứng với sự cho phép alert và notification để thực thi
  - Sau đó cập nhật trạng thái và thông tin lên label và button trên chart
- Liên tục kiểm tra xem các lệnh đang mở đã được kéo sl hòa vốn hay chưa? Nếu chưa, kiểm tra đủ điều kiện và thực hiện

12 Hàm thực thi khi có sự kiện click vào button

[ ]:
```
//+------------------------------------------------------------------+
// CHECK BUY/SELL EVENT FROM BUTTON
//+------------------------------------------------------------------+

void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
{
    if (id == CHARTEVENT_OBJECT_CLICK)
    {
        if (sparam == "_buy_")
        {
            btn_buy.execution();
        }
        else if (sparam == "_sell_")
        {
            btn_sell.execution();
        }
        else if (sparam == "_close_")
        {
            btn_close.execution();
        }
    }
}
```

12.0.1 Giải thích:

- Thực thi vào lệnh hoặc đóng lệnh, khi có sự kiện click vào button tương ứng

13 Hàm phá hủy toàn bộ object

[ ]:
```
//+------------------------------------------------------------------+
// DESTROYS ALL
//+------------------------------------------------------------------+

void OnDeinit(const int reason)
{
    ObjectsDeleteAll(0);
}
```

13.0.1 Giải thích:

- Xóa bỏ toàn bộ Object trên chart khi tháo Bot công cụ ra khỏi chart

14 Hàm cài đặt ban đầu và Hạn sử dụng

[ ]:
```
//+------------------------------------------------------------------+
// SETUP & EXPIRED
//+------------------------------------------------------------------+

int OnInit()
{
    if (TimeCurrent() > D'01.07.2024')
        return (INIT_FAILED);

    tf_trade.draw_status();
    tf_base01.draw_status();
    tf_base02.draw_status();

    btn_close.create_button();
    btn_buy.create_button();
    btn_sell.create_button();

    btn_dollar_sl.create_button();
    btn_points.create_button();
    btn_vol.create_button();

    return (INIT_SUCCEEDED);
}
```

14.0.1 Giải thích:

- Kiểm tra hạn sử dụng
- Vẽ lại các label và button trên chart mới khi chuyển timeframe

[ ]:
```

```