

```

//変数の宣言
extern int Adjust_Stop = 2;
extern int Stop_Loss = 3;
extern int Take_Profit = 3;

int Ticket_L = 0;
int Ticket_S = 0;

double Pips = 0;

//関数の定義
double AdjustPoint(string Currency)
{
    int Symbol_Digits = (int)MarketInfo(Currency,MODE_DIGITS);

    if(Symbol_Digits == 2 || Symbol_Digits == 3)
    {
        double Calculated_Point = 0.01;
    }
    else if(Symbol_Digits == 4 || Symbol_Digits == 5)
    {
        Calculated_Point = 0.0001;
    }

    return(Calculated_Point);
}

int init()
{
    Pips = AdjustPoint(Symbol());

    return(0);
}

int start()
{
    //保有ポジションの決済処理を記述

    //エントリー処理
    //買いエントリー
    if(買いエントリー条件)
    {
        Ticket_L = OrderSend(Symbol(),OP_BUY,...,0,0,...);

        if(Ticket_L > 0)
        {
            if(OrderSelect(Ticket_L,SELECT_BY_TICKET) == true)
            {
                //ストップレベルの計算
                double Stop_Level = MarketInfo(Symbol(),MODE_STOPLEVEL) * Point; ←ア

                RefreshRates();

                double Upper_Stop_Level = Ask + Stop_Level; ←イ
                double Lower_Stop_Level = Bid - Stop_Level;

                double Minimum_Stop = Adjust_Stop * Pips; ←ウ

                //損切り値と利益確定値の計算
                if(Stop_Loss > 0)
                {
                    double Buy_Stop_Loss = OrderOpenPrice() - (Stop_Loss * Pips); ←ア
                }
                if(Take_Profit > 0)
                {
                    double Buy_Take_Profit = OrderOpenPrice() + (Take_Profit * Pips); ←イ
                }

                //損切り値と利益確定値の検証
                if(Buy_Stop_Loss > 0 && Buy_Stop_Loss > Lower_Stop_Level)
                {
                    Buy_Stop_Loss = Lower_Stop_Level - Minimum_Stop; ←ア

                    Print("Stop-Loss has been automatically adjusted.");
                }
                if(Buy_Take_Profit > 0 && Buy_Take_Profit < Upper_Stop_Level)
                {
                    Buy_Take_Profit = Upper_Stop_Level + Minimum_Stop; ←イ

                    Print("Take-Profit has been automatically adjusted.");
                }

                //注文の変更
                if(Buy_Stop_Loss > 0 || Buy_Take_Profit > 0)
                {
                    bool Modified = OrderModify(OrderTicket(),OrderOpenPrice(),
                                                Buy_Stop_Loss,Buy_Take_Profit,0,clrNONE);
                }
            }
        }
    }

    //売りエントリー
    if(売りエントリー条件)
    {
        Ticket_S = OrderSend(Symbol(),OP_SELL,...,0,0,...);

        if(Ticket_S > 0)
        {
            if(OrderSelect(Ticket_S,SELECT_BY_TICKET) == true)
            {
                //ストップレベルの計算
                Stop_Level = MarketInfo(Symbol(),MODE_STOPLEVEL) * Point;

                RefreshRates();

                Upper_Stop_Level = Ask + Stop_Level;
                Lower_Stop_Level = Bid - Stop_Level;

                Minimum_Stop = Adjust_Stop * Pips;

                //損切り値と利益確定値の計算
                if(Stop_Loss > 0)
                {
                    double Sell_Stop_Loss = OrderOpenPrice() + (Stop_Loss * Pips);
                }
                if(Take_Profit > 0)
                {
                    double Sell_Take_Profit = OrderOpenPrice() - (Take_Profit * Pips);
                }

                //損切り値と利益確定値の検証
                if(Sell_Stop_Loss > 0 && Sell_Stop_Loss < Upper_Stop_Level)
                {
                    Sell_Stop_Loss = Upper_Stop_Level + Minimum_Stop;

                    Print("Stop-Loss has been automatically adjusted.");
                }
                if(Sell_Take_Profit > 0 && Sell_Take_Profit > Lower_Stop_Level)
                {
                    Sell_Take_Profit = Lower_Stop_Level - Minimum_Stop;

                    Print("Take-Profit has been automatically adjusted.");
                }

                //注文の変更
                if(Sell_Stop_Loss > 0 || Sell_Take_Profit > 0)
                {
                    Modified = OrderModify(OrderTicket(),OrderOpenPrice(),
                                                Sell_Stop_Loss,Sell_Take_Profit,0,clrNONE);
                }
            }
        }
    }

    return(0);
}

```