```
#include <Trade/Trade.mqh>

input double Lots = 0.1;
input double RiskPercent = 2.0;

input int OrderDisPoints = 200;
input int TpPoints = 200;
input int SlPoints = 200;
input int TslPoints = 5;
input int TslTriggerPoints = 10;

input ENUM_TIMEFRAMES Timeframe = PERIOD_H1;
input int BarsN = 5;
input int ExpirationHours = 50;
input int Magic = 111;

CTrade trade;

ulong buyPos, sellPos;
int totalBars;
double sl;
double tp;
double lots;

int OnInit()
  {
  trade.SetExpertMagicNumber(Magic);
   return(INIT_SUCCEEDED);
  }

void OnDeinit(const int reason)
  {

  }

void OnTick(){

   processPos(buyPos);
   processPos(sellPos);

   int bars = iBars(_Symbol, Timeframe);
   if(totalBars != bars){
      totalBars = bars;

      if(buyPos <= 0){
         double high = findHigh();
         if(high > 0){
         executeBuy(high);
         }
      }
       if(sellPos <= 0){
       double low = findLow();
         if(low > 0){
         executeBuy(low);
```

```
       }
      }
     }
   }

void OnTradeTransaction(

   const MqlTradeTransaction&    trans,
   const MqlTradeRequest&  request,
   const MqlTradeResult&  result
){
   if(trans.type == TRADE_TRANSACTION_ORDER_ADD){
      COrderInfo order;

      if(order.Select(trans.order)){
      if(order.Magic() == Magic){
      if(order.OrderType() == ORDER_TYPE_BUY_STOP){
         buyPos = order.Ticket();

      }else if(order.OrderType() == ORDER_TYPE_SELL_STOP){
         sellPos = order.Ticket();
      }
     }
    }
   }
  }

void processPos(ulong &posTicket){
   if(posTicket <= 0) return;
   if(OrderSelect(posTicket)) return;

   CPositionInfo pos;
   if(!pos.SelectByTicket(posTicket)){
   posTicket = 0;
   return;

   }else{
      if(pos.PositionType() == POSITION_TYPE_BUY){
      double bid = SymbolInfoDouble(_Symbol,SYMBOL_BID);

      if(bid > pos.PriceOpen() + TslTriggerPoints * _Point){
      double sl = bid - TslPoints * _Point;
       sl = NormalizeDouble(sl,_Digits);

      if(sl > pos.StopLoss()){
         trade.PositionModify(pos.Ticket(),sl,pos.TakeProfit());


        }
      }
    }else if(pos.PositionType() == POSITION_TYPE_SELL){
      double ask = SymbolInfoDouble(_Symbol,SYMBOL_ASK);

      if(ask < pos.PriceOpen() - TslTriggerPoints * _Point){
```

```
          double sl = ask +TslPoints * _Point;
           sl = NormalizeDouble(sl,_Digits);

           if(sl < pos.StopLoss() || pos.StopLoss() == 0){
          trade.PositionModify(pos.Ticket(),sl,pos.TakeProfit());

          }
       }
     }
   }
}

void executeBuy(double entry){
   entry = NormalizeDouble(entry,_Digits);

   double ask = SymbolInfoDouble(_Symbol,SYMBOL_ASK);
   if(ask > entry - OrderDisPoints * _Point) return;

    tp = entry + TpPoints * _Point;
   tp = NormalizeDouble(tp,_Digits);

   sl = entry - SlPoints * _Point;
   sl = NormalizeDouble(sl,_Digits);

    lots = Lots;
   if(RiskPercent > 0) lots = calcLots(entry - sl);

   datetime expiration = iTime(_Symbol,Timeframe,0) + ExpirationHours *
PeriodSeconds(PERIOD_H1);

   trade.BuyStop(lots,entry,_Symbol,sl,tp,ORDER_TIME_SPECIFIED,expiration);

   buyPos = trade.ResultOrder();

}

void executeSell(double entry){
   entry = NormalizeDouble(entry,_Digits);

   double ask = SymbolInfoDouble(_Symbol,SYMBOL_ASK);
   if(ask > entry - OrderDisPoints * _Point) return;

   tp = entry - TpPoints * _Point;
   tp = NormalizeDouble(tp,_Digits);

   sl = entry + SlPoints * _Point;
   sl = NormalizeDouble(sl,_Digits);

    lots = Lots;
   if(RiskPercent > 0) lots = calcLots(sl - entry);

   datetime expiration = iTime(_Symbol,Timeframe,0) + ExpirationHours *
PeriodSeconds(PERIOD_H1);
```

```
   trade.SellStop(lots,entry,_Symbol,sl,tp,ORDER_TIME_SPECIFIED,expiration);

   sellPos = trade.ResultOrder();
}

double calcLots(double SlPoints){

   double risk = AccountInfoDouble(ACCOUNT_BALANCE) * RiskPercent / 100;

   double ticksize = SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
   double tickvalue = SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_VALUE);
   double lotstep = SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);

   double moneyPerlotstep = SlPoints / ticksize + tickvalue * lotstep;
   double lots = MathFloor(risk / moneyPerlotstep) * lotstep;

   lots = MathMin(lots,SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX));
   lots = MathMax(lots,SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN));

   return lots;
}

double findHigh(){
   double highestHigh = 0;
   for(int i = 0; i <200; i++){
      double high = iHigh(_Symbol,Timeframe,i);
      if(i > BarsN && iHighest(_Symbol,Timeframe,MODE_HIGH,BarsN*2+1,i-5) == i){
         if(high > highestHigh){
         return high;
         }
       }
       highestHigh = MathMax(high,highestHigh);
   }
   return -1;
}

double findLow(){
   double lowestLow = 0;
   for(int i = 0; i <200; i++){
      double low = iLow(_Symbol,Timeframe,i);
      if(i > BarsN && iLowest(_Symbol,Timeframe,MODE_HIGH,BarsN*2+1,i-5) == i){
         if(low > lowestLow){
         return low;
         }
       }
       lowestLow = MathMax(low,lowestLow);
   }
   return -1;
}
```