```mql4
#property copyright "Copyright 2021, Zyuna32246"
#property link      "https://zyunafx.com/"
#property version   "1.00"
#property strict
//---パラメーター設定
sinput int MagicNumber=2021;            //マジックナンバー
input int Slippage=10;                  //スリッページ
input double Lots=0.01;                 //ロット数
input int TakeProfit=100;               //利確幅(pips)
input int StopLoss=50;                  //損切幅(pips)
input bool TrailingSwitching=true;      //トレーリング(On=true;OFF=false)
input int TrailingPips=20;              //トレーリングストップ幅(pips)
input string comment="BB_Cross_Zn";     //コメント
//---手法選択
enum MethodChange{
         MarketFollower,    //順張り
         Contrarian,        //逆張り
         Doten              //ドテン
         };
input MethodChange MethodInput=MarketFollower;  //手法選択
//---インジケーター設定
input string BB="";                             //ボリンジャーバンド
input int BBPeriod=20;                          //期間
input int BBDeviation=2;                        //偏差
input ENUM_APPLIED_PRICE BBPrice=PRICE_CLOSE;   //価格
input string MA="";                             //移動平均線
input int MAPeriod=5;                           //期間
input ENUM_MA_METHOD MAMethod=MODE_SMA;         //方法
input ENUM_APPLIED_PRICE MAPrice=PRICE_CLOSE;   //価格
//---プログラム全体に影響する変数
double Pips;
int SLP;
//+------------------------------------------------------------------+
int OnInit(){                           //EA導入時に一度だけ動くプログラム

   Pips=AdjustPoint(_Symbol);
   SLP=(int)AdjustSlippage(_Symbol,Slippage);

   return(INIT_SUCCEEDED);
}
//+------------------------------------------------------------------+
void OnTick(){                          //ティック受信する度に動くプログラム

//トレーリングストップ
   if(TrailingSwitching){
      int sec=TimeSeconds(TimeGMT());
      if(sec==0 || sec==30){
         TrailingStop();
      }
   }

//ポジション数を確認
   int BuyPosition=Position(1);
   int SellPosition=Position(-1);

//エントリー判定
   int Entry=0;
   int Closed=0;

   //順張りロジック
   if(MethodInput==MarketFollower){
      if(BuyPosition==0){
         if(BBCross1()==1)OpenOrder(1);
      }else{
         if(BBCross2()==1)CloseOrder(1);
      }
      if(SellPosition==0){
         if(BBCross1()==-1)OpenOrder(-1);
      }else{
         if(BBCross2()==-1)CloseOrder(-1);
      }
   }else
   if(MethodInput==Contrarian){
      if(BuyPosition==0){
         if(BBCross1()==-1)OpenOrder(1);
      }else{
         if(BBCross3()==1)CloseOrder(1);
      }
      if(SellPosition==0){
         if(BBCross1()==1)OpenOrder(-1);
      }else{
         if(BBCross3()==-1)CloseOrder(-1);
      }
   }else
   if(MethodInput==Doten){
      if(BuyPosition==0){
         if(BBCross1()==-1)OpenOrder(1);
      }else{
         if(BBCross1()==1){
            CloseOrder(1);
            OpenOrder(-1);
         }
      }
      if(SellPosition==0){
         if(BBCross1()==1)OpenOrder(-1);
      }else{
         if(BBCross1()==-1){
            CloseOrder(-1);
            OpenOrder(1);
         }
```

```
      }
   }

}//OnTick
//+------------------------------------------------------------------+
void OnDeinit(const int reason){ //EA削除時に最後に動くプログラム
}
//+------------------------------------------------------------------+
                                 //オリジナル関数
//ポジション調整
double AdjustPoint(string Currency){
   int SymbolDigits=(int)MarketInfo(Currency,MODE_DIGITS);
   double CalculatedPoint=0.0;
   if(SymbolDigits==2 || SymbolDigits==3){
      CalculatedPoint=0.01;
   }else
   if(SymbolDigits==4 || SymbolDigits==5){
      CalculatedPoint=0.0001;
   }
   return(CalculatedPoint);
}

//スリッページ調整
double AdjustSlippage(string Currency,int SlippagePips){
   double CalculatedSlippage=0.0;
   int SymbolDigits=(int)MarketInfo(Currency,MODE_DIGITS);
   if(SymbolDigits==2 || SymbolDigits==3){
      CalculatedSlippage=SlippagePips;
   }else
   if(SymbolDigits==4 || SymbolDigits==5){
      CalculatedSlippage=SlippagePips*10;
   }
   return(CalculatedSlippage);
}

//ポジション数を取得
int Position(int PositionDirection){
   int res=0;
   for(int i=0;i<OrdersTotal();i++){
      if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==true){
         if(OrderSymbol()==_Symbol && OrderMagicNumber()==MagicNumber){
            if(PositionDirection==1){
               if(OrderType()==OP_BUY)res++;
            }else
            if(PositionDirection==-1){
               if(OrderType()==OP_SELL)res++;
            }
         }
      }
   }
   return(res);
}

//ポジションエントリー関数
void OpenOrder(int EntryPosition){
   int res=0;
   double TP=TakeProfit*Pips;
   double SL=StopLoss*Pips;
   if(EntryPosition==1){
      res=OrderSend(_Symbol,OP_BUY,Lots,Ask,SLP,Ask-SL,Ask+TP,comment,MagicNumber,0,clrBlue);
   }else
   if(EntryPosition==-1){
      res=OrderSend(_Symbol,OP_SELL,Lots,Bid,SLP,Bid+SL,Bid-TP,comment,MagicNumber,0,clrRed);
   }
}

//ポジションクローズ関数
void CloseOrder(int ClosePosition){
   int res=0;
   for(int i=OrdersTotal()-1;i>=0;i--){
      if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==true){
         if(OrderSymbol()==_Symbol && OrderMagicNumber()==MagicNumber){
            if(OrderType()==OP_BUY && ClosePosition==1){
               res=OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),SLP,clrMagenta);
            }else
            if(OrderType()==OP_SELL && ClosePosition==-1){
               res=OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),SLP,clrMagenta);
            }
         }
      }
   }
}

//トレーリングストップ
void TrailingStop(){
   double TSP;
   int res=0;
   for(int i=OrdersTotal()-1;i>=0;i--){
      if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==true){
         if(OrderSymbol()==_Symbol && OrderMagicNumber()==MagicNumber){
            if(OrderType()==OP_BUY){
               TSP=Ask-TrailingPips*Pips;
               if(OrderOpenPrice()<TSP && OrderStopLoss()<TSP){
                  res=OrderModify(OrderTicket(),OrderOpenPrice(),TSP,0,0,clrYellow);
               }
            }else
            if(OrderType()==OP_SELL){
               TSP=Bid+TrailingPips*Pips;
               if(OrderOpenPrice()>TSP && OrderStopLoss()>TSP){
                  res=OrderModify(OrderTicket(),OrderOpenPrice(),TSP,0,0,clrYellow);
```

```
                }
            }
        }
    }
  }
}

//ボリンジャーバンドクロス（順）
int BBCross1(){
   int judge=0;
   double MA1=iMA(_Symbol,0,MAPeriod,0,MAMethod,MAPrice,1);
   double MA2=iMA(_Symbol,0,MAPeriod,0,MAMethod,MAPrice,2);
   double BU1=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_UPPER,1);
   double BU2=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_UPPER,2);
   double BL1=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_LOWER,1);
   double BL2=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_LOWER,2);

   if(MA2<=BU2 && BU1<MA1)judge=1;
   if(MA2>=BL2 && BL1>MA1)judge=-1;

   return(judge);
}

//ボリンジャーバンドクロス（逆）
int BBCross2(){
   int judge=0;
   double MA1=iMA(_Symbol,0,MAPeriod,0,MAMethod,MAPrice,1);
   double MA2=iMA(_Symbol,0,MAPeriod,0,MAMethod,MAPrice,2);
   double BU1=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_UPPER,1);
   double BU2=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_UPPER,2);
   double BL1=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_LOWER,1);
   double BL2=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_LOWER,2);

   if(MA2>=BU2 && BU1>MA1)judge=1;
   if(MA2<=BL2 && BL1<MA1)judge=-1;

   return(judge);
}

//ボリンジャーバンドクロス（中）
int BBCross3(){
   int judge=0;
   double MA1=iMA(_Symbol,0,MAPeriod,0,MAMethod,MAPrice,1);
   double MA2=iMA(_Symbol,0,MAPeriod,0,MAMethod,MAPrice,2);
   double BM1=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_MAIN,1);
   double BM2=iBands(_Symbol,0,BBPeriod,BBDeviation,0,BBPrice,MODE_MAIN,2);

   if(MA2<=BM2 && BM1<MA1)judge=1;
   if(MA2>=BM2 && BM1>MA1)judge=-1;

   return(judge);
}
```