

ซอสโค้ดระบบซื้อขายอัตโนมัติหลายสกุลเงิน

ในส่วนนี้จะเป็นส่วนหนึ่งของซอสโค้ดของโปรแกรมที่ใช้ในโปรแกรม Metatrader 5 ของกลุ่มระบบซื้อขายอัตโนมัติหลายสกุลเงิน

```
#property copyright "Copyright 2020, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.01" // check PositionsTotal()==0 before open order
#property version   "1.02" // close buy at target
#property version   "1.03" // input lot p1-p2-p3
#property version   "1.04" // edit close order by target
#property version   "1 . 0 5 " // add funtion open_order and close_order with
POSITION_TICKET
#property version   "1.06" // add StopLoss
#property version   "1.07" // add funtion LineNotify
#property version   "1.08" // add status ea and remove Stoploss

#include <Trade\Trade.mqh>
#include <Math\Stat\Math.mqh> // class for libraries calculations mathematics.
CTrade          trade ;

input string Step1 = "==== Currency pairs Setting ====="; // • Step 1
int MG_B          = 1 ; //Magic Number
input string P1    = "EURUSDm" ; //Symbol P1
input double Lot_P1 = 0.01 ; //Lot P1
input string P2    = "GBPUSDm" ; //Symbol P2
input double Lot_P2 = 0.02 ; //Lot P2
input string P3    = "USDCHFm" ; //Symbol P3
input double Lot_P3 = 0.03 ; //Lot P3

input string Step4 = "==== Indicator Bollinger bands Setting ====="; //•
Step 2
input double BB_Period = 50 ; //BB Period
input double STD       = 2.0 ; //BB STD
input double TP_Target = 15.0; // TP_Target(USD)
string version = "1.08";
```

```

input string Step5 = "==== Line Notify Setting ====="; //• Step 3
input bool Use_LineNotify = false; //• Use LineNotify
string message = "", endl = "\n";
string message_2 = "";
input string token = "gAQrlXr9ifJ1jHPdjk2djZ17GWt9lfqENQLAeXzB7nl"; // Token
input string api_url = "https://notify-api.line.me/api/notify"; // URL API

```

```

// status EA Send to LineNotify
string status_1 = "BUY";
string status_2 = "SELL";
string status_3 = "CLOSE";
string status_4 = "EA Runing";
string status_5 = "EA Stop";

```

```

//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//---
    Call_LineNotify(status_4);

//---
    return(INIT_SUCCEEDED);
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//---
    Call_LineNotify(status_5);
}

//+-----+
//| Expert tick function |

```

```

//+-----+
int MG_S = MG_B + 987 ;
void OnTick() {
    MqlTradeRequest  myRequest ;
    MqlTradeResult   myResult ;
    MqlTick          Tick ;
    //-----| BUY
    if(EntrySignal() == "Buy" && (PositionsTotal()==0)) {

        BUY();

    }//end if

    //-----| SELL
    if(EntrySignal() == "Sell" && (PositionsTotal()==0)) {
        SELL();
    }//end if

    //-----| CLOSE
    if (AccountInfoDouble(ACCOUNT_PROFIT)>=+MathAbs(TP_Target))
    { Call_LineNotify(status_3);
      for(int i=PositionsTotal()-1;i>=0;i--)
      {
          if(PositionSelectByTicket(PositionGetTicket(i)))
          {
              trade.PositionClose(PositionGetInteger(POSITION_TICKET));
              message_2 + = " Close TP Target by Ticket =
"+IntegerToString(PositionGetTicket(i)) + endl;
          }
      }

    }
}
//end function
//+-----+
// funtion  OpenOrder Position BUY

```

```

void BUY(){
trade.Buy( Lot_P1,
           P1,
           NULL, // execution price
           NULL, // stop loss price
           NULL, // take profit price
           NULL  // comment
           );
trade.Buy( Lot_P2,
           P2,
           NULL, // execution price
           NULL, // stop loss price
           NULL, // take profit price
           NULL  // comment
           );
trade.Buy( Lot_P3,
           P3,
           NULL, // execution price
           NULL, // stop loss price
           NULL, // take profit price
           NULL  // comment
           );

message_2 += "Buy :";
message_2 += " Lot="+DoubleToString(Lot_P1,2);
message_2 += " Symbol="+P1;
message_2 += endl;

message_2 += "Buy :";
message_2 += " Lot="+DoubleToString(Lot_P2,2);
message_2 += " Symbol="+P2;
message_2 += endl;

message_2 += "Buy :";
message_2 += " Lot="+DoubleToString(Lot_P3,2);
message_2 += " Symbol="+P3;

```

```

message_2 += endl;

Call_LineNotify(status_1);

}

//funtion  OpenOrder Position SELL
void SELL(){
    trade.Sell( Lot_P1,
                P1,
                NULL, // execution price
                NULL, // stop loss price
                NULL, // take profit price
                NULL // comment
            );
    trade.Sell( Lot_P2,
                P2,
                NULL, // execution price
                NULL, // stop loss price
                NULL, // take profit price
                NULL // comment
            );
    trade.Sell( Lot_P3,
                P3,
                NULL, // execution price
                NULL, // stop loss price
                NULL, // take profit price
                NULL // comment
            );
    message_2 += "Sell :";
    message_2 += " Lot="+DoubleToString(Lot_P1,2);
    message_2 += " Symbol="+P1;
    message_2 += endl;

    message_2 += "Sell :";
    message_2 += " Lot="+DoubleToString(Lot_P2,2);

```

```

message_2 += " Symbol="+P2;
message_2 += endl;

message_2 += "Sell .:";
message_2 += " Lot="+DoubleToString(Lot_P3,2);
message_2 += " Symbol="+P3;
message_2 += endl;

Call_LineNotify(status_2);

}

//funtion CloseOrder

void CloseAll_MG( int MG ) {

} //end function

string EntrySignal(){
    string Output = "" ;
    double C1 = 0, C2 = 0, C3 = 0, Price = 0 ;
    int i = 0 ;
    double Sum = 0, AVG = 0, UBand = 0, LBand = 0, SD = 0 ;
    for( i = 0 ; i < BB_Period ; i++ ) {
        C1 = iClose( P1, PERIOD_CURRENT, i ) ; C2 = iClose( P2, PERIOD_CURRENT, i ) ;
        C3 = iClose( P3, PERIOD_CURRENT, i ) ;
        Price = C1 * C2 / C3 ;
        Sum += Price ;
    } //end for
    AVG = Sum / BB_Period ;

    double hold = 0 ;
    for( i = 0 ; i < BB_Period ; i++ ) {
        C1 = iClose( P1, PERIOD_CURRENT, i ) ; C2 = iClose( P2, PERIOD_CURRENT, i ) ;
        C3 = iClose( P3, PERIOD_CURRENT, i ) ;
        Price = C1 * C2 / C3 ;
    }
}

```

```

        hold += MathPow( ( Price - AVG ), 2 );
    }//end for
    SD = MathSqrt( hold / ( BB_Period - 1 ) );
    UBand = AVG + SD * STD ;
    LBand = AVG - SD * STD ;

    i = 1 ;
    C1 = iClose( P1, PERIOD_CURRENT, i );    C2 = iClose( P2, PERIOD_CURRENT, i );
    C3 = iClose( P3, PERIOD_CURRENT, i );
    double PriceNext = C1 * C2 / C3 ;
    double BBW = ( ( PriceNext - LBand ) / ( UBand - LBand ) ) * 100 ;

    if( BBW > 100 )    Output = "Sell" ;
    else if( BBW < 0 )    Output = "Buy" ;

    Comment(
        "\n Order Buy  = " + DoubleToString( OrdersTotalMG( MG_B ), 0 )
        + "\n Order Sell = " + DoubleToString( OrdersTotalMG( MG_S ), 0 )
        + "\n\nPrice = " + DoubleToString( Price, (int)Digits() )
        + "\n\n-----\nMA = " + DoubleToString( AVG, 5 )
        + "\nUBand = " + DoubleToString( UBand, 5 )
        + "\nLBand = " + DoubleToString( LBand, 5 )
        + "\n\nBBW = " + DoubleToString( BBW, 2 ) + " %"
    );

    return Output ;
} //end function

string ExitSignal(){
    string Output = "" ;

    double C1 = 0 ;
    double C2 = 0 ;
    double C3 = 0 ;

```

```

double Price = 0 ;

int i = 0 ;
double Sum = 0 ;
double AVG = 0 ;
double UBand = 0 ;
double LBand = 0 ;
double SD    = 0 ;

for( i = 0 ; i < BB_Period ; i++ ) {
    C1 = iClose( P1, PERIOD_CURRENT, i ) ;
    C2 = iClose( P2, PERIOD_CURRENT, i ) ;
    C3 = iClose( P3, PERIOD_CURRENT, i ) ;
    Price = C1 * C2 / C3 ;
    Sum += Price ;
} //end for
AVG = Sum / BB_Period ;

double hold = 0 ;
for( i = 0 ; i < BB_Period ; i++ ) {
    C1 = iClose( P1, PERIOD_CURRENT, i ) ;
    C2 = iClose( P2, PERIOD_CURRENT, i ) ;
    C3 = iClose( P3, PERIOD_CURRENT, i ) ;
    Price = C1 * C2 / C3 ;
    hold += MathPow( ( Price - AVG ), 2 ) ;
} //end for
SD = MathSqrt( hold / ( BB_Period - 1 ) ) ;
UBand = AVG + SD * STD ;
LBand = AVG - SD * STD ;

//-----|
i = 0 ;
C1 = iClose( P1, PERIOD_CURRENT, i ) ;
C2 = iClose( P2, PERIOD_CURRENT, i ) ;
C3 = iClose( P3, PERIOD_CURRENT, i ) ;
Price = C1 * C2 / C3 ;

```



```

//double BBW = ( ( UBand - LBand ) / AVG ) * 100 ;
double BBW = ( ( Price - LBand ) / ( UBand - LBand ) ) * 100 ;

if( OrdersTotalMG( MG_B ) > 0 && BBW >= 50 ) {
    Output = "ExitBuy" ;
} //end if

if( OrdersTotalMG( MG_S ) > 0 && BBW <= 50 ) {
    Output = "ExitSell" ;
} //end if

return Output ;
} //end function

int OrdersTotalMG( int MG ) {
    int Output = 0 ;
    ulong t = 0 ;

    for( int i = 0 ; i < (int)PositionsTotal() ; i++ ) {
        t = PositionGetTicket( i ) ;
        if( PositionSelectByTicket( t ) ) {
            if( PositionGetInteger( POSITION_MAGIC ) == MG ) {
                Output++ ;
            } //end if
        } //end if
    } //end for
    return Output ;
} //end function

//function LineNotify
void LineNotify(string Message)
{
    string headers;
    char post[], result[];

    headers="Authorization: Bearer "+token+"\r\n";

```

```

headers+="Content-Type: application/x-www-form-urlencoded\r\n";

ArrayResize(post,StringToCharArray("message="+Message,post,0 ,WHOLE_ARRAY,CP_UTF8)-1);
int res = WebRequest("POST", "https://notify-api.line.me/api/notify", headers, 10000,
post, result, headers);
Print("Status code: " , res, " , error: " , GetLastError());
Print("Server response: " , CharArrayToString(result));
}

//function Call LineNotify
void Call_LineNotify(string _status)
{
    if(!Use_LineNotify) return;

    message = "";
    message += "สถานะ: "+_status+ ".";
    message += " \n แจ้งเตือนรายละเอียดดังนี้\n";
    message += "AccountNumber : "+AccountInfoString(ACCOUNT_NAME)+endl;
    message += "Balance : "+DoubleToString(AccountInfoDouble(ACCOUNT_BALANCE),2)+endl;
    message += "Equity : "+DoubleToString(AccountInfoDouble(ACCOUNT_EQUITY),2)+endl;
    message += "Profit : "+DoubleToString(AccountInfoDouble(ACCOUNT_PROFIT),2)+endl;
    message += message_2 +endl;
    message_2 = "";
    LineNotify(message);
}

```