

```

//+-----+
//|          Nasser03.mq5 |
//|          Copyright 2014, MetaQuotes Software Corp. |
//|          http://www.mql5.com |
//+-----+

#property copyright "Copyright 2014, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

/-- input parameters
input int   StopLoss=30;   // Stop Loss
input int   TakeProfit=100; // Take Profit
input int   ADX_Period=8;  // ADX Period
input int   MA_Period=8;   // Moving Average Period
input int   EA_Magic=12345; // EA Magic Number
input double Adx_Min=22.0; // Minimum ADX Value
input double Lot=100000;   // Lots to Trade

/-- Other parameters
int adxHandle; // handle for our ADX indicator
int maHandle;  // handle for our Moving Average indicator
double plsDI[],minDI[],adxVal[]; // Dynamic arrays to hold the values of +DI, -DI and ADX values for each
bars
double maVal[]; // Dynamic array to hold the values of Moving Average for each bars
double p_close; // Variable to store the close value of a bar
int STP, TKP; // To be used for Stop Loss & Take Profit values

//+-----+
//| Expert initialization function |
//+-----+

int OnInit()

```

```

{
//---
//--- Get handle for ADX indicator
    adxHandle=iADX(NULL,0,ADX_Period);
//--- Get the handle for Moving Average indicator
    maHandle=iMA(_Symbol,_Period,MA_Period,0,MODE_EMA,PRICE_CLOSE);
//--- What if handle returns Invalid Handle
    if(adxHandle<0 || maHandle<0)
    {
        Alert("Error Creating Handles for indicators - error: ",GetLastError(),"!!!");
    }
//---
//--- Get the last price quote using the MQL5 MqlTick Structure

    return(INIT_SUCCEEDED);
}
//+-----+
//| Expert deinitialization function          |
//+-----+
void OnDeinit(const int reason)
{
//---
    IndicatorRelease(adxHandle);
    IndicatorRelease(maHandle);

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

```

```

void OnTick()
{
//---
// Do we have enough bars to work with
if(Bars(_Symbol,_Period)<60) // if total bars is less than 60 bars
{
Alert("We have less than 60 bars, EA will now exit!!");
return;
}
// We will use the static Old_Time variable to serve the bar time.
// At each OnTick execution we will check the current bar time with the saved one.
// If the bar time isn't equal to the saved time, it indicates that we have a new tick.
static datetime Old_Time;
datetime New_Time[1];
bool IsNewBar=false;

// copying the last bar time to the element New_Time[0]
int copied=CopyTime(_Symbol,_Period,0,1,New_Time);
if(copied>0) // ok, the data has been copied successfully
{
if(Old_Time!=New_Time[0]) // if old time isn't equal to new bar time
{
IsNewBar=true; // if it isn't a first call, the new bar has appeared
if(MQL5InfoInteger(MQL5_DEBUGGING)) Print("We have new bar here ",New_Time[0]," old time
was ",Old_Time);
Old_Time=New_Time[0]; // saving bar time
}
}
else

```

```

{
    Alert("Error in copying historical times data, error =",GetLastError());
    ResetLastError();
    return;
}

```

//--- EA should only check for new trade if we have a new bar

```

if(IsNewBar==false)
{
    return;
}

```

//--- Do we have enough bars to work with

```

int Mybars=Bars(_Symbol,_Period);
if(Mybars<60) // if total bars is less than 60 bars
{
    Alert("We have less than 60 bars, EA will now exit!!");
    return;
}

```

//--- Define some MQL5 Structures we will use for our trade

```

MqlTick latest_price; // To be used for getting recent/latest price quotes
MqlTradeRequest mrequest; // To be used for sending our trade requests
MqlTradeResult mresult; // To be used to get our trade results
MqlRates mrate[]; // To be used to store the prices, volumes and spread of each bar
ZeroMemory(mrequest); // Initialization of mrequest structure
ZeroMemory(mresult); // Initialization of mrequest structure
/*

```

Let's make sure our arrays values for the Rates, ADX Values and MA values

```

    is store serially similar to the timeseries array
*/
// the rates arrays
    ArraySetAsSeries(mrate,true);
// the ADX DI+values array
    ArraySetAsSeries(plsDI,true);
// the ADX DI-values array
    ArraySetAsSeries(minDI,true);
// the ADX values arrays
    ArraySetAsSeries(adxVal,true);
// the MA-8 values arrays
    ArraySetAsSeries(maVal,true);
if(!SymbolInfoTick(_Symbol,latest_price))
    {
        Alert("Error getting the latest price quote - error:",GetLastError(),"!!");
        return;
    }
//--- Get the details of the latest 3 bars
if(CopyRates(_Symbol,_Period,0,3,mrate)<0)
    {
        Alert("Error copying rates/history data - error:",GetLastError(),"!!");
        return;
    }
//--- Copy the new values of our indicators to buffers (arrays) using the handle
if(CopyBuffer(adxHandle,0,0,3,adxVal)<0 || CopyBuffer(adxHandle,1,0,3,plsDI)<0
    || CopyBuffer(adxHandle,2,0,3,minDI)<0)
    {
        Alert("Error copying ADX indicator Buffers - error:",GetLastError(),"!!");
        return;
    }

```

```

}
if(CopyBuffer(maHandle,0,0,3,maVal)<0)
{
    Alert("Error copying Moving Average indicator buffer - error:",GetLastError());
    return;
}
//--- we have no errors, so continue
//--- Do we have positions opened already?
bool Buy_opened=false; // variable to hold the result of Buy opened position
bool Sell_opened=false; // variable to hold the result of Sell opened position

if (PositionSelect(_Symbol) ==true) // we have an opened position
{
    if (PositionGetInteger(POSITION_TYPE) == POSITION_TYPE_BUY)
    {
        Buy_opened = true; //It is a Buy
    }
    else if(PositionGetInteger(POSITION_TYPE) == POSITION_TYPE_SELL)
    {
        Sell_opened = true; // It is a Sell
    }
}

// Copy the bar close price for the previous bar prior to the current bar, that is Bar 1

p_close=mrRate[1].close; // bar 1 close price

/*
1. Check for a long/Buy Setup : MA-8 increasing upwards,
previous price close above it, ADX > 22, +DI > -DI

```

```

*/
//--- Declare bool type variables to hold our Buy Conditions
bool Buy_Condition_1 = (maVal[0]>maVal[1]) && (maVal[1]>maVal[2]); // MA-8 Increasing upwards
bool Buy_Condition_2 = (p_close > maVal[1]); // previous price closed above MA-8
bool Buy_Condition_3 = (adxVal[0]>Adx_Min); // Current ADX value greater than minimum value
(22)
bool Buy_Condition_4 = (plsDI[0]>minDI[0]); // +DI greater than -DI

//--- Putting all together
if(Buy_Condition_1 && Buy_Condition_2)
{
if(Buy_Condition_3 && Buy_Condition_4)
{
// any opened Buy position?
if (Buy_opened)
{
Alert("We already have a Buy Position!!!");
return; // Don't open a new Buy Position
}
mrequest.action = TRADE_ACTION_DEAL; // immediate order execution
mrequest.price = NormalizeDouble(latest_price.ask,_Digits); // latest ask price
mrequest.sl = NormalizeDouble(latest_price.ask - STP*_Point,_Digits); // Stop Loss
mrequest.tp = NormalizeDouble(latest_price.ask + TKP*_Point,_Digits); // Take Profit
mrequest.symbol = _Symbol; // currency pair
mrequest.volume = Lot; // number of lots to trade
mrequest.magic = EA_Magic; // Order Magic Number
mrequest.type = ORDER_TYPE_BUY; // Buy Order
mrequest.type_filling = ORDER_FILLING_IOC; // Order execution type
mrequest.deviation=100; // Deviation from current price

```

```

    }
//--- send order
    if(OrderSend(mrequest,mresult)!=true)
    {
        Alert("The Buy order request could not be completed -error:",GetLastError());
        ResetLastError();
        return;
    }

// get the result code
    if(mresult.retcode==10009 || mresult.retcode==10008) //Request is completed or order placed
    {
        Alert("A Buy order has been successfully placed with Ticket#:",mresult.order,"!!");
    }
    else
    {
        Alert("The Buy order request could not be completed -error:",GetLastError());
        ResetLastError();
        return;
    }
}

/*
2. Check for a Short/Sell Setup : MA-8 decreasing downwards,
previous price close below it, ADX > 22, -DI > +DI
*/

//--- Declare bool type variables to hold our Sell Conditions
    bool Sell_Condition_1 = (maVal[0]<maVal[1]) && (maVal[1]<maVal[2]); // MA-8 decreasing
downwards
    bool Sell_Condition_2 = (p_close <maVal[1]); // Previous price closed below MA-8

```

```

    bool Sell_Condition_3 = (adxVal[0]>Adx_Min);           // Current ADX value greater than
minimum (22)
    bool Sell_Condition_4 = (plsDI[0]<minDI[0]);         // -DI greater than +DI

//--- Putting all together
if(Sell_Condition_1 && Sell_Condition_2)
{
    if(Sell_Condition_3 && Sell_Condition_4)
    {
        // any opened Sell position?
        if (Sell_opened)
        {
            Alert("We already have a Sell position!!!");
            return; // Don't open a new Sell Position
        }
        mrequest.action = TRADE_ACTION_DEAL;           // immediate order execution
        mrequest.price = NormalizeDouble(latest_price.bid,_Digits); // latest Bid price
        mrequest.sl = NormalizeDouble(latest_price.bid + STP*_Point,_Digits); // Stop Loss
        mrequest.tp = NormalizeDouble(latest_price.bid - TKP*_Point,_Digits); // Take Profit
        mrequest.symbol = _Symbol;                    // currency pair
        mrequest.volume = Lot;                         // number of lots to trade
        mrequest.magic = EA_Magic;                     // Order Magic Number
        mrequest.type= ORDER_TYPE_SELL;                // Sell Order
        mrequest.type_filling = ORDER_FILLING_IOC;     // Order execution type
        mrequest.deviation=100;
    } // Deviation from current price

//--- send order
    if(OrderSend(mrequest,mresult)!=true)

```

```
{  
    Alert("The Sell order request could not be completed -error:",GetLastError());  
    ResetLastError();  
    return;  
}
```

```
if(mresult.retcode==10009 || mresult.retcode==10008) //Request is completed or order placed
```

```
{  
    Alert("A Sell order has been successfully placed with Ticket#:",mresult.order,"!!");  
}
```

```
else
```

```
{  
    Alert("The Sell order request could not be completed -error:",GetLastError());  
    ResetLastError();  
    return;  
}  
}
```

```
}
```

```
//+-----+
```