

Hello,

A month ago, an expert here wrote a custom EA for me based on a special indicator. However, as I used the EA, problems arose—both bugs and condition paradoxes. Unnecessary and untimely BUY-SELL conditions occurred. Correct and perfect orders closed unexpectedly... etc. Anyway, I have now modified this EA a bit. Some additions were made, and conditions were adjusted based on a new indicator. Adjustments were also made to AI modules. There is a proper flowchart, and the logic is good. However, now I have some issues in MetaEditor. I suspect certain codes are not working properly. Therefore, I need an expert who can quickly identify and fix these issues in the EA. I kept the explanations long to explain EA and reduce the number of questions. I will also share all the documents.

EA1: old version (to be attached, does not give any error in Meta Editor.)

EA2: New version (to be attached, gives 7 errors in Meta editor and will optimize.)

Special Indicator: no problem. Will share. "147609_SP_06"

EA Summary:

- The special indicator **Track1** determines order placement based on the upward and downward movements of **L2, L3, L4** lines, triggered by **L1**. Orders are executed at the market price (no pending orders).
- **Sto1 and Sto2** support buy/sell conditions with **K period** confirmation.
- **TP, SL, Trailing Stop, and _BE_TP** conditions work with a trailing stop linked to **Track1 LBE**.
- The EA checks the trading days of the week and adjusts parameters based on market conditions (ATR-based).

Tasks:

1. Fix **7 errors** found in MetaEditor during compilation (detailed in Note 3 below).
2. Debug the mode where EA works with **BE_Normal and _BE_TP**, ensuring indicator-based **TP and SL** operate correctly (detailed in Note 1 below).
3. Ensure that **indicator and condition queries refresh on each new bar instead of every tick** to optimize EA performance.
4. Verify the **12 BUY-SELL conditions** in EA2 and correct their logic if necessary (simple explanation provided in Note 2 below).
5. **Remove unnecessary and irrelevant code** (not by hiding with "//", but completely deleting them).
6. **Check and correct** parameters of **two Stochastic Oscillators** if there are any issues.
7. Ensure **TP, SL, Trailing Stop, BE, and RiskRewardRatio formulas** are correctly implemented and compatible with each other.
8. Verify if the following condition is working correctly, and fix it if necessary:

```
input bool    CloseOnOppositeSignal    = false;           // Close Position on Opposite  
Signal
```

9. Simplify the code where possible (at the developer's discretion).
10. Ensure **all transactions are based on points, not pips**.
11. Perform **general optimization**.

EA Flowchart and Summary

General Flowchart (Simplified)

EA Flow Chart and Summary

General Flow Chart (Simplified):

graph LR

```
A[Start (OnInit)] --> B{Load Indicators?};
B -- Yes --> C{Successful?};
B -- No --> H[Error (INIT_FAILED) and Stop];
C -- Yes --> D[Go to Tick Function (OnTick)];
C -- No --> H[Error (INIT_FAILED) and Stop];
D --> E{New Bar Formed? (IsNewBar)};
E -- Yes --> F{Open Position Exists? (HasOpenPosition)};
E -- No --> D;
F -- Yes --> G[Manage Positions (ManagePositions)];
F -- No --> I[Check Entry Signals (CheckEntrySignals)];
G --> D;
I --> J{Signal Exists? (GetEntrySignal)};
J -- Yes --> K{Buy Signal?};
J -- No --> D;
K -- Yes --> L{Buy Entry Allowed? (IsBuyEntryAllowed)};
K -- No --> M{Sell Signal?};
L -- Yes --> N[Place Buy Order (ExecuteOrder - BUY)];
L -- No --> D;
M -- Yes --> O{Sell Entry Allowed? (IsSellEntryAllowed)};
M -- No --> D;
O -- Yes --> P[Place Sell Order (ExecuteOrder - SELL)];
O -- No --> D;
N --> D;
P --> D;
H --> End[End (OnDeinit)];
D --> End;
End --> Q[EA Stopped (OnDeinit)];
```

subgraph OnInit Subprocesses

```
A --> B
C --> D
end
```

subgraph OnTick Subprocesses

```
D --> E
F --> G
I --> J
K --> L
M --> O
end
```

subgraph Position Management (ManagePositions)

```
G --> G1[Stop Loss Management (ManageStopLoss)]
G1 --> G2[Take Profit Management (ManageTakeProfit)]
G2 --> G3[Breakeven Management (ManageBreakeven)]
G3 --> G4{Trailing Stop Used?}
G4 -- Yes --> G5[Trailing Stop Management (ManageTrailingStop)]
G4 -- No --> G6{Trailing TP Used?}
G5 --> G6
G6 -- Yes --> G7[Trailing TP Management (ManageTrailingTP)]
G6 -- No --> G
G7 --> G
end
```

subgraph Entry Signal Check (CheckEntrySignals)

```
I --> J
J -- Yes --> K
K -- Yes --> L
M -- Yes --> O
end
```

subgraph Entry Permission Check (IsBuyEntryAllowed / IsSellEntryAllowed)

```
L --> L1[Trade Direction Check (TradeDirectionCheck)]
```

```

L1 --> L2[Time Filter (TimeFilter)]
L2 --> L3[Spread Filter (SpreadFilter)]
L3 --> L4[Cooldown Check (CooldownCheck)]
L4 --> L5[Maximum Position Check (MaxPositionCheck)]
L5 --> L{Buy Entry Allowed?}

O --> O1[Trade Direction Check (TradeDirectionCheck)]
O1 --> O2[Time Filter (TimeFilter)]
O2 --> O3[Spread Filter (SpreadFilter)]
O3 --> O4[Cooldown Check (CooldownCheck)]
O4 --> O5[Maximum Position Check (MaxPositionCheck)]
O5 --> O{Sell Entry Allowed?}
end

subgraph Order Placement (ExecuteOrder)
N --> N1[Calculate Lot Size (CalculateLotSize)]
N1 --> N2[Calculate Stop Loss (CalculateStopLoss)]
N2 --> N3[Calculate Take Profit (CalculateTakeProfit)]
N3 --> N4[Send Buy Order (trade.Buy)]
N4 --> N

P --> P1[Calculate Lot Size (CalculateLotSize)]
P1 --> P2[Calculate Stop Loss (CalculateStopLoss)]
P2 --> P3[Calculate Take Profit (CalculateTakeProfit)]
P3 --> P4[Send Sell Order (trade.Sell)]
P4 --> P
end

```

EA's Functionalities and Operation Order (Summary):

This EA (Expert Advisor), **Advanced_Track1_EA.mq5**, is an advanced trading robot designed for automated trading in financial markets. The basic functions and operation order of the EA are as follows:

1. Initialization (OnInit):

- Retrieves and refreshes symbol information.
- Sets time filters and parameters.
- Loads the "**147609_SP_06**" custom indicator, **Stochastic (70 and 140 periods)**, and **ATR** indicators. If the indicators cannot be loaded, the EA cannot be started.
- Configures trading settings (Magic Number, comment, slippage, fill type).
- Prints that the EA has started and the symbol and Magic Number.

2. Tick Operations (OnTick):

This function runs every time a price movement (tick) arrives.

- Refreshes symbol prices.
- Checks if the account mode is "Hedging" (the EA only works in this mode).
- **Checks if a new bar has formed (IsNewBar).** Operations are usually performed when a new bar is formed.

If a new bar has formed:

Checks if there is an open position (HasOpenPosition).

If there is an open position: Manages positions (**ManagePositions**). In this step, it checks and updates Stop Loss, Take Profit, Breakeven, and Trailing Stop/TP settings.

If there is no open position: Checks for new entry signals (**CheckEntrySignals**).

Checks whether an entry signal is received (GetEntrySignal). This function sequentially checks the active condition functions (con1-con12) to determine if a buy or sell signal is generated.

If a buy signal exists: Checks if buy entry is allowed (**IsBuyEntryAllowed**). The permission check evaluates factors such as trade direction, time filter, spread filter, cooldown period, and maximum number of positions. If permission is granted, it places a buy order (**ExecuteOrder - BUY**).

If a sell signal exists: Checks if sell entry is allowed (**IsSellEntryAllowed**). The permission check and order placement process are similar to the buy signal (**ExecuteOrder - SELL**).

3. Position Management (ManagePositions): Used for managing open positions.

- Loops through open positions.
- For each position:

Manages Stop Loss (ManageStopLoss). Checks the current stop loss level according to the stop loss type (fixed points or ATR multiplier) and updates it if necessary.

Manages Take Profit (ManageTakeProfit). Checks the current take profit level according to the take profit type (fixed points or risk/reward ratio) and updates it if necessary.

Manages Breakeven (ManageBreakeven). Checks breakeven settings (trigger point, profit point, ATR multiplier, TP percentage) and moves the stop loss to the breakeven level or profit level. In Breakeven TP percentage mode, it can also trigger trailing stop/TP management after breakeven.

Manages Trailing Stop (ManageTrailingStop). If Trailing Stop is active and not in Breakeven TP percentage mode, it moves the stop loss level up or down according to the Trailing Stop type (fixed points, ATR, or LBE). The LBE Trailing Stop option uses a special LBE indicator value for trailing stop.

Manages Trailing TakeProfit (ManageTrailingTP). If Trailing TP is active and in Breakeven TP percentage mode, it moves the Take Profit level up or down.

4. Entry Signal Check (CheckEntrySignals and GetEntrySignal): Evaluates entry signals for new positions.

- **GetEntrySignal:** Sequentially checks the entry conditions (con1 - con12 functions) between EnableCondition_1 and EnableCondition_12. If an active condition function generates a buy or sell signal, it returns this signal. If no condition generates a signal, it returns no signal (-1).
- **CheckEntrySignals:** Evaluates the signal received from the GetEntrySignal function. If a buy signal is received and the "Close Position on Opposite Signal" setting is active, it closes open sell positions. Then, it checks if buy entry is allowed and places a buy order if permission is granted. A similar process is followed for sell signals.

5. Entry Permission Checks (IsBuyEntryAllowed, IsSellEntryAllowed and related filter functions): Checks various filters before entering a new order.

- **Trade Direction Check (TradeDirectionCheck):** Checks whether the EA should trade in buy only, sell only, or both directions.
- **Time Filter (TimeFilter and DayTimeFilter):** Checks whether the EA should trade within the specified time intervals and days. Daily and weekly time filters can be set separately.
- **Spread Filter (SpreadFilter):** Checks if the current spread value is lower than the specified maximum spread value.
- **Cooldown Check (CooldownCheck):** Prevents excessively frequent trading by checking if a certain number of bars have passed since the last trade and if the price has moved a certain number of points.
- **Maximum Position Check (MaxPositionCheck):** Checks if the number of open positions exceeds the specified maximum number of positions.

6. **Order Placement (ExecuteOrder and Lot/SL/TP Calculation Functions):** Sending orders to the market and calculating lot size, stop loss, and take profit levels.
- **CalculateLotSize:** Calculates the lot size according to the lot size type (fixed lot, risk percentage, dynamic ATR). The calculation considers parameters such as account balance, risk percentage, ATR value, and maximum lot size.
 - **CalculateStopLoss:** Calculates the Stop Loss level according to the stop loss type (fixed points or ATR multiplier).
 - **CalculateTakeProfit:** Calculates the Take Profit level according to the take profit type (fixed points or risk/reward ratio). When using risk/reward ratio, it determines the take profit distance using the stop loss distance.
 - **ExecuteOrder:** Sends a buy or sell order to the market using the calculated lot, stop loss, and take profit values. Records the transaction time and price. Prints an error message if there is an order placement error.
7. **Indicator Value Retrieval Functions (stoch, atr, GetLBEValue, _CopyBuffer):** Helper functions used to retrieve indicator values.
- **stoch:** Retrieves values from the Stochastic indicator (70 or 140 period).
 - **atr:** Retrieves values from the ATR indicator.
 - **GetLBEValue:** Retrieves the LBE (Last Bar Estimation) value from the special "147609_SP_06" indicator (default buffer 6).
 - **_CopyBuffer:** A general helper function to copy buffer data from indicators.
8. **Helper Functions:** Various general-purpose helper functions (time conversion, enum conversion, fill type control, position counting, price normalization, logging, error logging, etc.).

Missing Parts and Suggestions (Summary):

- **Lack of Visualization:** The EA does not visually display entry signals or indicator values on the chart. **Suggestion:** Enhance understandability and traceability of the EA by drawing entry signals, indicator lines, or other important information on the chart.
- **No Backtesting and Optimization Functions:** The EA does not include specific functions for backtesting or parameter optimization. **Suggestion:** Add backtesting and optimization functions to evaluate the strategy's performance and optimize parameters.
- **Weak Advanced Error Management:** Basic error management exists (indicator loading, order placement errors), but more comprehensive error management can be added. **Suggestion:** Increase the robustness of the EA by adding more detailed error logging, automatic correction mechanisms in case of errors, or more informative error messages for the user.
- **No News Filter:** A news filter is not included for situations where high-impact news can increase market volatility and negatively affect strategy performance. **Suggestion:** Add a news filter to stop trading or reduce position risk during high-impact news times.
- **No Multi-Timeframe Analysis:** The EA only operates on the current chart timeframe. **Suggestion:** Add multi-timeframe analysis (e.g., trend confirmation in higher timeframes) for stronger signals and better filtering.
- **Performance Monitoring Can Be Improved:** Basic logging is present, but more comprehensive performance monitoring can be added. **Suggestion:** Add functions to calculate and log, or even graphically display, important performance metrics such as win rate, profit/loss ratio, maximum drawdown.
- **Complexity and Testability of Signal Conditions:** There are 12 different entry conditions (con1-con12), and they are complex. **Suggestion:** Making the conditions more modular and easily testable can simplify the process of adding new conditions and modifying existing ones. Perhaps conditions can be broken down into simpler building blocks, offering the possibility to use them in different combinations.

Note 1:

Integrate the function described below into the EA. Modify, structure, and code it in the most logical way.

_BE_TP// BE TP %

If this part is selected:

```
input string group_track1_lbe      = "=== Track1 LBE Settings ===";
input int    LBE_Inpmalen          = 120;                          // LBE MA Length
input ENUM_MA_METHOD LBE_Inpmamethod = MODE_EMA;                  // LBE MA Method
input int    LBE_Inplen            = 30;                          // LBE Averaging Length
input int    LBE_Inplen11          = 60;                          // LBE Prediction Line 1 Length
input int    LBE_Shift              = -60;                        // LBE Shift
```

As the **Track1 LBE line moves upward** with the given input parameters (if **BEMode=_BE_TP** is selected):

Input Parameters (Example):

```
Use_BE=true;
BEMode=_BE_TP;
BETPStart=50; // 50%
Use_TrailingStop=true;

ATRPeriod=14;
TrailingKCoef=1.6;
Use_TrailingTP=true;
TrailingTPKCoef=0.8;
TakeProfit1=1000; // Initial Take Profit value
```

Scenario:

- A **BUY** position is opened at **0.70000** on the **AUD/USD** pair, and the initial **Take Profit (TP)** is set to **1000 points (0.71000)**.
- The price starts rising.
- When the price reaches **50% of the initial TP target** (i.e., **50 points or 0.70500**), the **BETPStart value (50%)** is triggered.
- The **TP Percentage Breakeven** feature is activated. The stop loss level is moved **1 point** above the entry price (**0.70001**).
- After that, both **Trailing Stoploss (ATR-based)** and **Trailing TakeProfit (ATR-based)** features become active.

Trailing Stoploss (ATR-Based):

- The EA calculates the ATR (Average True Range) indicator value using **ATRPeriod (14)**.
- **Stop loss** level moves **upwards** whenever the price increases, adding **ATR * TrailingKCoef (1.6)** points to the existing stop loss level.

Trailing TakeProfit (ATR-Based):

- Similarly, **Take Profit** level is calculated using **ATR * TrailingTPKCoef (0.8)**.
- **Take Profit** moves **upwards** whenever the price increases, adding **ATR * 0.8 points** to the existing take profit level.

As the price continues to rise, both **stop loss** and **take profit** are automatically adjusted upwards based on **ATR and Track1 LBE movements**.

- If the price drops and reaches the **stop loss**, the position **closes in profit**.
- If the price continues to rise and hits the **take profit** level, the position **closes with a higher profit**.

Let's integrate and adjust these codes accordingly.

Example:

```
BUY position opened: 1.20000
TP: 800
SL: 100
BETPStart: 50; // 50% * TP distance - triggering (so here 400)
Trailing Coeff: 1.6
TrailingTPKCoef: 0.8
ATR: 14
```

- **SL is moved to 1.20001.**
- Then:
 - **Condition:** If the **Track1 LBE line is moving upward**, the function continues to operate as follows.
 - **On each new bar:**
 - **SL moves up by $ATR * TrailingKCoef (1.6)$.**
 - **TP moves up by $ATR * TrailingTPKCoef (0.8)$.**

1. This code is already working in EA1.
2. It is partially available in EA2. It will be fully integrated and made functional.

Note 2: Trading Conditions

a) BUY Signal:

There are **12 BUY conditions** as explained below:

```
1-(L4 < L3 < L1 < L2) && L4=B && L3=B && L2=R && Sto(70) < 50 && L1TB
2-(L4 < L1 < L2 < L3) && L4=B && L2=R && L3=B && Sto(70) < 50 && L1TB
3-(L4 < L3 < L1 < L2) && L4=B && L3=R && L2=R && Sto(70) < 20 && Sto(140) < 50 &&
L1TB
4-(L4 < L1 < L2 < L3) && L4=B && L3=R && L2=R && Sto(70) < 20 && Sto(140) < 50 &&
L1TB
5-(L1 < L4 < L2 < L3) && L4=B && L2=R && L3=B && Sto(70) < 50 && L1TB
6-(L1 < L4 < L2 < L3) && L4=B && L2=R && L3=R && Sto(70) < 20 && Sto(140) < 50 &&
L1TB
7-(L1 < L2 < L3 < L4) && L2=R && L3=B && L4=B && Sto(70) < 50 && L1TB
8-(L1 < L2 < L3 < L4) && L2=R && L3=R && L4=B && Sto(70) < 20 && Sto(140) < 20 &&
L1TB
9-(L3 < L1 < L2 < L4) && L3=B && L2=R && L4=B && Sto(70) < 50 && L1TB
10-(L3 < L1 < L2 < L4) && L3=R && L2=R && L4=B && Sto(70) < 20 && L1TB
11-(L3 < L4 < L1 < L2) && L3=B && L4=B && L2=R && Sto(70) < 20 && Sto(140) < 50 &&
L1TB
12-(L3 < L4 < L1 < L2) && L3=R && L4=B && L2=R && Sto(70) < 20 && Sto(140) < 20 &&
L1TB
```

b) SELL Signal:

As explained under below, there are 12 SELL conditions.

```
1-(L4 > L3 > L1 > L2) && L4=R && L3=R && L2=B && Sto(70) > 50 && L1TR
2-(L4 > L1 > L2 > L3) && L4=R && L2=B && L3=R && Sto(70) > 50 && L1TR
3-(L4 > L3 > L1 > L2) && L4=R && L3=B && L2=B && Sto(70) > 80 && Sto(140) > 50 &&
L1TR
4-(L4 > L1 > L2 > L3) && L4=R && L3=B && L2=B && Sto(70) > 80 && Sto(140) > 50 &&
L1TR
5-(L1 > L4 > L2 > L3) && L4=R && L2=B && L3=R && Sto(70) > 50 && L1TR
6-(L1 > L4 > L2 > L3) && L4=R && L2=B && L3=B && Sto(70) > 80 && Sto(140) > 50 &&
L1TR
```

```

7-(L1 > L2 > L3 > L4) && L2=B && L3=R && L4=R && Sto(70) > 50 && L1TR
8-(L1 > L2 > L3 > L4) && L2=B && L3=B && L4=R && Sto(70) > 80 && Sto(140) > 80 &&
L1TR
9-(L3 > L1 > L2 > L4) && L3=R && L2=B && L4=R && Sto(70) > 50 && L1TR
10(L3 > L1 > L2 > L4) && L3=B && L2=B && L4=R && Sto(70) > 80 && L1TR
11-(L3 > L4 > L1 > L2) && L3=R && L4=R && L2=B && Sto(70) > 80 && Sto(140) > 50 &&
L1TR
12-(L3 > L4 > L1 > L2) && L3=B && L4=R && L2=B && Sto(70) > 80 && Sto(140) > 80 &&
L1TR

```

c) Other definition details:

1-) * L1, L2, L3, L4 is the indicators Track1, if they are equal B then its mean is the line is increasing. So while coding use that;
1 previous bar (i-1) < current bar (i).
And reverse of it; if they are equal R then its mean is the line is decreasing. So while coding use that; 1 previous bar (i-1) > current bar (i).

2-) * L1TB Means L1 Line turns from decreasing to increasing. 1 bar before it was decreasing, now turned increasing. Use in coding that, for Track1 Line:

3 previous bar (i-3) > 2 previous bar (i-2) < 1 previous bar (i-1) < current bar (i).

At the same time this condition is Triggering condition.

** L1TR Means L1 Line turns from increasing to decreasing. 1 Bar before it was increasing, now turned decreasing. Use in coding that, for Track1 Line:

3 previous bar (i-3) < 2 previous bar (i-2) > 1 previous bar (i-1) > current bar (i).

At the same time this condition is Triggering condition.

3-) L4=R An example. Actually it is valid all other Lines. (L1, L2, L3). L4 at this time is decreasing, so while coding use that;

1 previous bar (i-1) > current bar (i).

Reverse of it L4=B, Actually it is valid of all other Lines. (L1, L2, L3). It is increasing, So while coding use that;

1 previous bar (i-1) < current bar (i).

4-) Sto1 (70) > 80 means is Standart Stochastic Oscillator (70) greater than 80 value. It is valid for other stochastic functions. If there are other definitions about Stochastic Oscillator same rules are valid. Sto1 (70) (default) actually variation

Note 3: Meta Editor Errors:

Trade.mqh Object.mqh StdLibErr.mqh OrderInfo.mqh HistoryOrderInfo.mqh PositionInfo.mqh

DealInfo.mqh AccountInfo.mqh SymbolInfo.mqh 'SetComment' - undeclared identifier 284 11

'TradeComment' - some operator expected

284 22

'Close' - undeclared identifier

596 71

'),' - expression expected

596 77

possible loss of data due to type conversion from 'double' to 'int'

653 34

'SymbolInfoInteger' - no one of the overloads can be applied to the function call

657 77

could be one of 2 function(s)

657 77

built-in: long SymbolInfoInteger(const string,ENUM_SYMBOL_INFO_INTEGER)

657 77

built-in: bool SymbolInfoInteger(const string,ENUM_SYMBOL_INFO_INTEGER,long&)

657 77

)' - expression expected

796 103

'SymbolInfoInteger' - no one of the overloads can be applied to the function call

813 48

could be one of 2 function(s)

813 48

built-in: long SymbolInfoInteger(const string,ENUM_SYMBOL_INFO_INTEGER)

813 48

built-in: bool SymbolInfoInteger(const string,ENUM_SYMBOL_INFO_INTEGER,long&)

813 48

7 errors, 1 warnings 8 2