Pflichtenheft für den Expert-Advisor „TimeTraderEA“

Sprache vom Dokument: Deutsch (German)

Ich möchte einen Expert Advisor (nachfolgend nur noch kurz „EA“) haben, der auf mehrere Benutzereingaben reagiert und nachfolgend beschriebene Funktionen fehlerfrei abarbeitet.

Grundlegend soll der EA auf Eingaben (Inputs) auf Datum und Uhrzeit reagieren und je nach Einstellung Stop-Orders in den Markt legen und verwalten.

Für ein schnelleres gemeinsames Verständnis habe ich eine *.mq5 Datei vorbereitet und mit einigen wenigen Eingaben usw. versehen, damit mein Wunsch vom Aufbau des Codes klarer wird.

1. Code, Strategietester, <input>:

1.1. Programmiersprache, Kommentare und Namensgebung im Code

Programmiert wird in MQL5, ich möchte selbstverständlich den vollen Quellcode haben.

Es sind im Code Kommentare in englischer Sprache für alle Funktionen, Schleifen, Variablen, Arrays, usw. zu schreiben, die deren Aufgabe und Bedeutung beschreiben. Zudem sind sinnvoll, verständliche Namen mit Bezug zur Funktion/Aufgabe für alles zu verwenden.

1.2. Strategietester, Live-Trading

Der EA soll sowohl im Strategietester, als auch im live-Trading identisch funktionieren.
Das bedeutet, dass Funktionen die im Strategietester nicht funktionieren wie z.B. „Sleep 1000“ verboten sind.

1.3. Include nicht erlaubt

Generell ist kein „Include“ erlaubt!
Gemeint ist z.B. diese Codezeile “#include<Trade/Trade.mqh>”

Da dieser Wunsch von mir in der Vergangenheit häufig auf Verwunderung bei Programmieranfängern gestoßen ist, liefere ich hier ein Beispiel welches performanter arbeitet und ausschließlich die ursprünglich hinterlegten Funktionen verwendet.
Kein sehr schönes Beispiel, aber ein recht deutliches, um meinen Wunsch zu verdeutlichen.

```
//+------------------------------------------------------------------+
//| Open Position: BUY
//+------------------------------------------------------------------+
void OpenBuyPosition()
{
```

```
  ulong ticket = 0;

  MqlTradeRequest request;
  MqlTradeResult  result;
  ZeroMemory(request);
  ZeroMemory(result);

  double ask = SymbolInfoDouble(_Symbol, SYMBOL_ASK);

  request.action      = TRADE_ACTION_DEAL;
  request.symbol      = _Symbol;
  request.volume      = InpLotSize;
  request.type        = ORDER_TYPE_BUY;
  request.price       = ask;
  request.type_filling = ORDER_FILLING_FOK;
  request.deviation   = 10;
  request.sl          = 0;
  request.tp          = 0;
  request.magic       = InpMagicNumber;
  request.position    = ticket;
  request.comment     = IntegerToString(InpMagicNumber);

  bool isOrderSend  = false;

  ResetLastError();
  isOrderSend = OrderSend(request, result);

  if (isOrderSend == true)
  {
    Print("BUY-Order created. Symbol: ", _Symbol, ", Magic-Number: ", InpMagicNumber);
  }
  if (isOrderSend == false)
  {
    Print("BUY-Order failed. Symbol: ", _Symbol, ", Magic-Number: ", InpMagicNumber, ". Error: ", GetLastError());
  }
}
```

1.4. Erwartete <input> Werte von dem EA

Für die bessere Übersicht habe ich eine *.mq5 Datei vorbereitet und darin grob die von mir gewünschten <input> Werte erzeugt.

Es handelt sich um eine erste Idee meiner <input> Wünsche und hat nicht den Anspruch auf Vollständigkeit.

Bei bedarf sollen/müssen Sie dort selbstverständlich erweitern.

1.5. Struct the EA / Funktionen

Ich erwarte, dass er EA Objektorientiert aufgebaut wird, genauso wie ich es mit dem < EA_STRUCT> schon vorgegeben habe. Dieses Struct darf/muss selbstverständlich angepasst/erweitert werden nach dem Funktionsumfang vom EA.

Ich halte sehr viel von der Verwendung von selbst aufgebauten, kleineren Funktionen um den Code übersichtlich zu halten.

1.6. Ausbaufähigkeit dieses EA

Dieser EA soll rein für mich als eine Art Master-EA dienen und zukünftig deutlich weiter ausgebaut werden. Deshalb soll er möglichst offen und flexibel programmiert sein.

Parallel entwickle ich selbst schon Erweiterungen, welche später zu einem EA verschmolzen werden sollen.

2. Funktionsumfang

Der EA soll unterschiedliche Handelsregeln verarbeiten, welche wie zuvor beschrieben und in der bereitgestellten *.mq5 Datei zusätzlich erklärten Art und Weise alle Funktionen fehlerfrei abarbeitet.

Der Funktionsumfang wird sowohl hier in diesem Dokument, als auch im beigefügten und rechtlich Gleichberechtigten *.mq5 gemeinsam dargestellt.

2.1. Vorbereitete ENUM

2.1.1. ENUM_Strategie

Mit diesem ENUM soll die allgemeine Strategie auswählbar sein,
ob einfach Stop-Orders in beide Richtungen,
oder ein Grid in beide Richtungen,
oder klar nur eine Richtung Buy or Sell vorgegeben wird.

2.1.2. ENUM_Trailing

Mit diesem ENUM soll das Trailing-Verhalten auswählbar sein,
ob z.B. fixe SL + TP Werte definiert vorgegeben werden,
oder ob ab dem Moment wo aus dem Deal, eine Position wurde mit fixen Distance Points der SL getrailed werden soll,
oder ob 2 stufig getrailed werden soll. Stufe 1 ab Abstand von n-Punkten im Profit soll der SL BreakEven gezogen werden und Stufe 2 ab Abstand von p-Punkten soll mit einem Abstand 2 der TSL nachgezogen werden.
Oder ab dem Moment wo aus Deal eine Position wurde, wenn der erste Tick in die Gegenrichtung der Position läuft soll diese geschlossen werden.

2.1.3. ENUM_LotSize

Hier soll die gewünschte Los-Size für die Position definiert werden, anhand fixer Wertvorgabe, oder prozentualem Wert der Equity / Ballance.

3. Strategie

Der EA soll nach den vorgegebenen <input> Werten das Datum und die Uhrzeit erkennen, um Deals zu eröffnen.

Das soll er ein paar Sekunden vor der Startzeit tun, mit SL+TP wie in den Einstellungen vorgegeben.

Wenn vorgegeben wurde, dass „OCO“ One-Cancel-the Other” aktiv ist, soll der EA ab dem Moment wo aus einer Sell-Stop Order eine Position wurde die Buy-Stop Order löschen.

Entweder läuft eine Position in den SL, in den TP oder wird durch die vorgegebene Zeit geschlossen.