GENERAL

mt 4 expert advisor programming tutorial

Published: September 3, 2026 | Index ID: #-

MT4 Expert Advisor Programming Tutorial: From Zero to Automated Trading

MetaTrader 4 (MT4) has become the go to platform for retail traders worldwide, thanks to its robust charting tools, fast execution, and the ability to run **Expert Advisors** (EAs). An EA is essentially a piece of code that can analyze market conditions, place orders, and manage trades automatically—allowing traders to harness the power of **algorithmic trading** without constantly monitoring the screen.

Whether you're a seasoned programmer or a trader who just wants to automate a simple strategy, this tutorial will walk you through every step of building, testing, and deploying an MT4 EA. By the end, you'll understand the core concepts of **MQL4**(MetaQuotes Language 4), how to structure an EA, and how to fine tune it for optimal performance.

Why Automate with MT4?

- Emotionless Execution – EAs follow rules strictly, eliminating the emotional biases that often lead to poor decisions.
- Speed – Automated systems can react to market events in milliseconds, far faster than a human trader.
- 24/7 Monitoring – Forex markets are open 24 hours. An EA can keep trading even when you're sleeping.
- Backtesting & Optimization – MT4's built in tools let you test an EA against historical data to validate its strategy before risking real money.

These advantages make mastering EA programming a valuable skill for anyone serious about **Forex** or **binary options** trading.

Getting Started: The Development Environment

1. Install MetaTrader 4

Download the free MT4 client from your broker's website or the official MetaTrader page. During installation, make sure to select a folder where you can easily locate the MQL4 directory.

2. Open the MetaEditor

Within MT4, press F4 or click the "MetaEditor" icon. MetaEditor is the IDE (Integrated Development Environment) where you'll write, compile, and debug your EA code.

3. Familiarize with the File Structure

- Experts – Folder for all EAs.
- Indicators – Custom indicators.
- Scripts – One time execution programs.

For this tutorial, we'll focus on the Experts folder.

Understanding MQL4 Basics

What is MQL4?

MQL4 is a C like programming language designed specifically for trading on MT4. It allows you to:

- Define custom indicators.
- Create scripts for one time tasks.
- Build Expert Advisors that can trade automatically.

Key Concepts

1. Functions – Reusable blocks of code. Examples: OnInit(), OnTick(), OnTimer().
2. Events – Triggers that call functions. The most important is OnTick(), which runs every time a new price tick is received.
3. Variables & Data Types – int, double, string, bool, datetime, etc.
4. Arrays – Collections of values. Useful for storing historical price data.
5. Objects – Predefined structures like OrderSend() for trading.

Creating Your First Expert Advisor

Step 1: Start a New EA

In MetaEditor, select **File !' New !' Expert Advisor (template)**. Name it SimpleTrendEA and click **Finish**. The IDE will generate a skeleton code with comments.

Step 2: Define Global Variables

These variables hold the parameters and state of your EA. They're declared outside any function so they're accessible throughout the code.

```
int      MagicNumber = 123456;   // Unique ID for orders
double   LotSize     = 0.01;     // Default trade size
int      StopLoss    = 50;       // Stop loss in pips
int      TakeProfit  = 100;      // Take profit in pips
```

Step 3: Initialize Settings

The OnInit() function runs once when the EA is attached to a chart. Here you can set up indicators or parameters.

```
int OnInit()
{
  Print("SimpleTrendEA initialized");
  return(INIT_SUCCEEDED);
}
```

Step 4: The Core Logic – OnTick()

Every time a new tick arrives, OnTick() executes. Inside this function, we'll:

1. Check if there are open trades.
2. Determine market trend using a moving average.
3. Open a new order if conditions are met.

```
void OnTick()
{
  // 1. If we already have an open trade, do nothing
  if(OrdersTotal() > 0) return;

  // 2. Calculate trend using a 50  period SMA
  double maCurrent = iMA(NULL, 0, 50, 0, MODE_SMA, PRICE_CLOSE, 0);
  double maPrevious = iMA(NULL, 0, 50, 0, MODE_SMA, PRICE_CLOSE, 1);

  // 3. Simple trend logic: if current price above MA, buy; if below, sell
  if(Ask > maCurrent)
  {
    // Place a BUY order
    int ticket = OrderSend(Symbol(), OP_BUY, LotSize, Ask, 2,
                           Ask - StopLoss * Point,
                           Ask + TakeProfit * Point,
                           "SimpleTrendEA", MagicNumber, 0, clrGreen);
    if(ticket > 0) Print("Buy order placed, ticket #", ticket);
  }
  else if(Bid  0) Print("Sell order placed, ticket #", ticket);
  }
}
```

Step 5: Compile and Test

Click the **Compile** button. If there are no errors, attach the EA to a chart (drag from the Navigator or double click). Watch the **Experts** tab for log messages. You can also enable **AutoTrading** to allow the EA to execute trades.

Deepening Your Knowledge: Advanced Features

1. Risk Management

Instead of a fixed LotSize, calculate it based on account equity and risk per trade.

```
double CalculateLotSize()
{
```

```
  double riskPerTrade = 0.02; // 2% of equity
  double accountEquity = AccountEquity();
  double lotSize = (accountEquity * riskPerTrade) / (StopLoss * Point * 100000);
  // Round to nearest 0.01
  lotSize = NormalizeDouble(lotSize, 2);
  return(lotSize);
}
```

2. Multiple Time Frame Analysis

Incorporate higher time frame signals to filter trades.

```
double GetHigherTFSignal()
{
  // Example: 1 hour MA cross
  double ma1 = iMA(NULL, PERIOD_H1, 50, 0, MODE_SMA, PRICE_CLOSE, 0);
  double ma2 = iMA(NULL, PERIOD_H1, 200, 0, MODE_SMA, PRICE_CLOSE, 0);
  if(ma1 > ma2) return(1); // Bullish
  if(ma1
```

3. Order Management

Track open orders, adjust stops, or close trades based on trailing stops.

```
void ManageOrders()
{
  for(int i = OrdersTotal() - 1; i >= 0; i--)
  {
    if(OrderSelect(i, SELECT_BY_POS, MODE_TRADES))
    {
      if(OrderMagicNumber() == MagicNumber)
      {
        // Example: trailing stop
        double trail = 30 * Point;
        if(OrderType() == OP_BUY && Bid - OrderOpenPrice() > trail)
        {
          double newStop = Bid - trail;
          OrderModify(OrderTicket(), OrderOpenPrice(), newStop, OrderTakeProfit(), 0, clrYellow);
        }
```

```
      }
    }
  }
}
```

4. Logging and Debugging

Use Print() statements to output variable values. For more advanced logging, write to a file:

```
void LogTrade(string message)
{
  string filename = "C:\\MQL4\\Logs\\TradeLog.txt";
  FileOpen(filename, FILE_WRITE | FILE_TXT);
  FileWrite(filename, TimeToString(TimeCurrent()), message);
  FileClose(filename);
}
```

Backtesting and Optimization

Before risking real money, you should test your EA against historical data.

How to Run a Backtest

1. Open the Strategy Tester (Ctrl+R).
2. Select your EA from the dropdown.
3. Choose the symbol, timeframe, and date range.
4. Set the initial deposit and leverage.
5. Click Start.

Review the **Results** tab for profit, drawdown, and trade statistics.

Optimization Tips

- Use Optimization mode to automatically adjust parameters (e.g., moving average periods, stop loss, take profit).
- Set realistic ranges to avoid overfitting.
- Check the Graph tab to visualize equity curves.
- Validate the optimized parameters on a different data set.

Deploying on a Live Account

1. Verify Trading Permissions

Ensure your broker allows automated trading. Some brokers disable **AutoTrading** on demo accounts.

2. Start with a Small Position

Begin with a single trade to confirm the EA behaves as expected.

3. Monitor Performance

Keep an eye on the **Terminal** > **Experts** tab. Look for any error messages or unusual behavior.

4. Implement Fail Safe Mechanisms

- Limit the number of concurrent trades.
- Use a stop loss at the account level to protect against catastrophic loss.

Tags: #expert #advisor #programming #tutorial