Boost Your EA Trading: Add a Custom Heads-Up Display (HUD)

Tired of digging through logs or switching windows to check your Expert Advisor's status? This guide shows you the **easiest way** to add a real-time Heads-Up Display (HUD) directly onto your MetaTrader chart using the simple Comment() function.

Why you need this guide:

- **Monitor Everything:** See account balance, equity, margin, P/L, open trades, and market data (spread, prices) at a glance.
- **Debug Faster:** Instantly check your EA's state, permissions (is trading *really* enabled?), and key metrics.
- **Track Performance:** Includes code snippets for counting Buy/Sell trades and even measuring **OrderSend() communication times** (Avg/Min/Max) to spot potential delays!
- **Get Started Quickly:** Provides clear, step-by-step MQL4/MQL5 instructions and warns about common pitfalls (like the MQL4 Comment limit).
- **Includes an AI Prompt:** Jumpstart your coding with a ready-made prompt to generate the basic HUD structure (see below!).

Stop guessing and start *seeing* what your EA is doing in real-time.

AI Prompt for HUD Code Generation

(Provide this prompt to an AI assistant like ChatGPT, Claude, or Gemini to help generate the basic code structure. Remember to specify MQL4 or MQL5!)

// Start AI Prompt Paste

Generate MQL[4/5] code for an Expert Advisor's OnTick() function to display a Heads-Up Display (HUD) using the Comment() function.

The HUD should be built as a single string variable and displayed using Comment() at the end of OnTick().

Include the following information in the HUD string, clearly labeled and separated by newlines ("\n"):

1. **EA Info:**
   * EA Name (Hardcoded string: "My Trading EA")
   * Magic Number (Assume a global variable named 'MagicNumber' exists)
   * Symbol (Current chart symbol)
   * Timeframe (Current chart timeframe)
2. **Time & Market:**
   * Server Time (Formatted as<y_bin_46>.MM.DD HH:MM:SS)
   * Current Bid Price
   * Current Ask Price
   * Current Spread (in points)
3. **Account Info:**
   * Account Balance
   * Account Equity
   * Free Margin
   * Margin Level (%)
4. **Trade Summary:**
   * Total Open Buy Positions (Count for current symbol/magic number)
   * Total Open Sell Positions (Count for current symbol/magic

number)
    * Total Open Positions (Sum of Buy and Sell)
5. **Trading Permissions:**
    * Text indicating if AutoTrading is "ENABLED" or "DISABLED" based on checking BOTH terminal trade allowance AND expert advisor trading allowance. Include reasons like "[Terminal Disabled]" or "[Experts Disabled]" if applicable.
6. **Order Send Timings:** (Provide placeholders/structure even if full implementation is complex)
    * Average OrderSend Time (ms) - Assume global variables `g_orderSendTotalTime_ms` or `_us`, `g_orderSendCount` exist.
    * Min OrderSend Time (ms) - Assume global variable `g_orderSendMinTime_ms` or `_us` exists.
    * Max OrderSend Time (ms) - Assume global variable `g_orderSendMaxTime_ms` or `_us` exists.

Ensure the code uses correct MQL[4/5] functions for retrieving data (e.g., Symbol(), Period(), TimeCurrent(), SymbolInfoDouble(), AccountInfoDouble(), OrdersTotal()/PositionsTotal(), OrderSelect()/PositionSelectByTicket, OrderType()/PositionGetInteger(POSITION_TYPE), IsTradeAllowed()/TerminalInfoInteger(TERMINAL_TRADE_ALLOWED), IsExpertEnabled()/AccountInfoInteger(ACCOUNT_TRADE_EXPERT)).

Format numbers appropriately (e.g., prices to the correct number of digits, percentages with one decimal place).

Provide the code within the OnTick() function structure. Add comments explaining each section of the HUD string construction.

// End AI Prompt Paste

Code Snippet Examples (MQL4/MQL5)

Here are brief examples showing how to get the data for each section and add it to your `hud_string`. Remember to initialize `hud_string = "";` at the start of `OnTick()`.

1. EA Info

```
// Assuming 'MagicNumber' is a global variable (e.g., input int MagicNumber = 12345;)
hud_string += "--- EA Info ---\n";
hud_string += "Name: My Trading EA\n";
hud_string += "Magic: " + IntegerToString(MagicNumber) + "\n";
// MQL5: hud_string += "Symbol: " + _Symbol + "\n";
// MQL4: hud_string += "Symbol: " + Symbol() + "\n";
// MQL5: hud_string += "TF: " + EnumToString(_Period) + "\n";
// MQL4: hud_string += "TF: " + IntegerToString(Period()) + "\n"; // Period() returns minutes
```

2. Time & Market

```
hud_string += "--- Market ---\n";
hud_string += "Time: " + TimeToString(TimeCurrent(), TIME_DATE|TIME_SECONDS) + "\n";

// MQL5 specific price/spread retrieval
// MqlTick latest_tick;
// SymbolInfoTick(_Symbol, latest_tick);
// hud_string += "Bid: " + DoubleToString(latest_tick.bid, _Digits) + "\n";
// hud_string += "Ask: " + DoubleToString(latest_tick.ask, _Digits) + "\n";
// long spread_points = SymbolInfoInteger(_Symbol, SYMBOL_SPREAD);
// hud_string += "Spread: " + IntegerToString(spread_points) + " points\n";
```

```
// MQL4 specific price/spread retrieval
// RefreshRates(); // Good practice before accessing Bid/Ask/MarketInfo
// hud_string += "Bid: " + DoubleToString(Bid, Digits) + "\n";
// hud_string += "Ask: " + DoubleToString(Ask, Digits) + "\n";
// int spread_points = MarketInfo(Symbol(), MODE_SPREAD);
// hud_string += "Spread: " + IntegerToString(spread_points) + " points\n";
```

3. Account Info

```
hud_string += "--- Account ---\n";
// MQL4 & MQL5 (AccountInfo functions are similar)
hud_string += "Balance: " +
DoubleToString(AccountInfoDouble(ACCOUNT_BALANCE), 2) + "\n";
hud_string += "Equity: " +
DoubleToString(AccountInfoDouble(ACCOUNT_EQUITY), 2) + "\n";
hud_string += "Free Margin: " +
DoubleToString(AccountInfoDouble(ACCOUNT_MARGIN_FREE), 2) + "\n";
hud_string += "Margin Level: " +
DoubleToString(AccountInfoDouble(ACCOUNT_MARGIN_LEVEL), 1) + "%\n";
```

4. Trade Summary (Conceptual - requires looping)

```
// Inside your trade loop (MQL5 example):
// if(PositionGetSymbol(i) == _Symbol && PositionGetInteger(POSITION_MAGIC)
== MagicNumber) {
//    ENUM_POSITION_TYPE type =
(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
//    if (type == POSITION_TYPE_BUY) totalBuyTrades++;
//    else if (type == POSITION_TYPE_SELL) totalSellTrades++;
// }
// Inside your trade loop (MQL4 example):
// if(OrderSelect(i, SELECT_BY_POS, MODE_TRADES)) {
//    if(OrderSymbol() == Symbol() && OrderMagicNumber() == MagicNumber) {
//       if (OrderType() == OP_BUY) totalBuyTrades++;
```

```
//      else if (OrderType() == OP_SELL) totalSellTrades++;
//    }
// }

// After the loop:
hud_string += "--- Trades ---\n";
hud_string += "Buys: " + IntegerToString(totalBuyTrades) + "\n";
hud_string += "Sells: " + IntegerToString(totalSellTrades) + "\n";
hud_string += "Total: " + IntegerToString(totalBuyTrades + totalSellTrades) +
"\n";
```

5. Trading Permissions

```
// MQL5 Example:
// bool terminalTradeAllowed =
TerminalInfoInteger(TERMINAL_TRADE_ALLOWED);
// bool expertTradeAllowed = AccountInfoInteger(ACCOUNT_TRADE_EXPERT);

// MQL4 Example:
// bool terminalTradeAllowed = IsTradeAllowed();
// bool expertTradeAllowed = IsExpertEnabled();

// Combine results:
string tradeStatus = (terminalTradeAllowed && expertTradeAllowed) ?
"ENABLED" : "DISABLED";
string reason = "";
if (!terminalTradeAllowed) reason += "[Terminal Disabled]";
if (!expertTradeAllowed) reason += "[Experts Disabled]";

hud_string += "--- Permissions ---\n";
hud_string += "AutoTrading: " + tradeStatus + " " + reason + "\n";
```

6. Order Send Timings (Display part)

```
// Assuming global variables g_... exist and are updated elsewhere
// double avgTime_ms = (g_orderSendCount > 0) ?
(double)(g_orderSendTotalTime_us / 1000) / g_orderSendCount : 0; // MQL5
(us to ms)
// double avgTime_ms = (g_orderSendCount > 0) ?
(double)g_orderSendTotalTime_ms / g_orderSendCount : 0; // MQL4 (ms)
// ulong minTime_ms = ... // Calculate from g_orderSendMinTime_us or _ms,
handle initial large value
// ulong maxTime_ms = ... // Calculate from g_orderSendMaxTime_us or _ms

hud_string += "--- Order Send Time (ms) ---\n";
hud_string += "Avg: " + DoubleToString(avgTime_ms, 1) + "\n";
hud_string += "Min: " + IntegerToString((long)minTime_ms) + "\n";
hud_string += "Max: " + IntegerToString((long)maxTime_ms) + "\n";
hud_string += "Count: " + IntegerToString(g_orderSendCount) + "\n";
```

Note on Variable Naming (Scope Prefixes vs. Hungarian Notation)

You might notice variables like `g_orderSendCount` in the timing examples. The `g_` prefix is a common convention in MQL (and other C-like languages) to indicate **variable scope**, specifically that the variable is **Global**. It tells a programmer reading the code that this variable exists outside the current function and retains its value between function calls (like `OnTick`). Other common scope prefixes include `s_` for static variables (local to a file or function but persistent) or sometimes `p_` or `a_` for function parameters/arguments.

This is different from **Hungarian Notation**, an older convention where prefixes indicate the **data type** or intended *use* of a variable (e.g., `iCount` for an integer count, `sName` for a string name, `bEnabled` for a boolean flag, `hwndWindow` for a handle to a window). While you might see Hungarian notation in older code or other languages,

using prefixes for *scope* (`g_`, `s_`) is generally more common and often considered more useful in modern MQL development for quickly understanding a variable's lifetime and accessibility. The `g_` in the examples signifies "global scope".

Provided by EgoNoBueno at Forex Factory 4/21/25.