ShellAutoTradingRobotCode 10.7.mq4 is an outline code shell that allows programmers to create mq4 trading EA's quickly and easily, using Bob's 10.7 trading rationale. It is not an auto-generator of the type available on the internet, so non-coders look away now; this is not for you.

The key functions of the shell are:

- int init()
- int start()
- void ReadIndicatorValues()
- void LookForTradingOpportunities()
- void CountOpenTrades()
- bool LookForTradeClosure(int ticket){

int init()

I declare inputs such as TakeProfitPips and StopLossPips as integers in the extern list, so that users know they are dealing with actual pips and not the points rubbish inflicted upon us by those morons at CrapperQuotes. In any sane computing language, double x = 1 / 4 would result in x = 0.25. In the lunatic world of Crapql4, the same equation returns x = 0, because an integer division creating a result < 0 is not allowed.

To overcome this, I use doubles converters, for instance TakeProfit and StopLoss. Int init() contains the conversions, for example TakeProfit = TakeProfitPips.

Int start()

I use start() mostly for calling the key functions. You will often find you can leave it untouched.

void ReadIndicatorValues()

Here is the process I follow whenever I add an indicator to my shell (this is merely how I do it – you may have different methods):

- Declare the extern inputs somewhere within my inputs list at the top of my code.
- Add a function that calls the indicator. Mine start with 'Get' as in double GetCSS(double index, int shift).
- Add code to ReadIndicatorValues to call the indi reading function.
- Add a chart feedback display to show the my call to the indi results in a correct value.

There is a key input early in my code – EveryTickMode. If 'true', then my calls to the indi functions usually require a value at the close of the previous candle to be compared with the value *now.* When EveryTickMode is 'false', then my calls to the indi functions usually require a value at the close of the two previous candles. I save a lot of faffing around with the int shift declaration at the top of the function.

void LookForTradingOpportunities()

Basic 10.7 is this:

- Long trend trade:
  - Red at the top.
  - Yellow at the top.
  - Stoch7 crosses the 10 or 30 levels from below.
- Long Counter Trend (CT) trade:
  - Red is at the bottom.

    - Yellow is at the top.
    - Stoch7 crosses the 10 or 30 levels from below.
- Short trend trade:
    - Red at the bottom.
    - Yellow at the bottom.
    - Stoch7 crosses the 90 or 70 levels from above.
- Short Counter Trend (CT) trade:
    - Red is at the top.
    - Yellow is at the bottom..
    - Stoch7 crosses the 90 or 70 levels from above.

The code to detect all this is in the two blocks beginning with the comment "//Specific system filters.". If the requirements for a long trade are met, then SendLong or SendShort are set to 'true' as appropriate.

Your indi filters then go in the two code blocks beginning with:

```
  //Usual filters
  if (SendLong)
  {
```

and

```
  //Usual filters
  if (SendShort)
  {
```

These blocks are commented 'Usual filters' because they contain the stuff I normally include. You can see how I implement a 'return' to cancel the trade if any of the filters fail. You can adopt this method.

void CountOpenTrades()
This function iterates through the open trades on your platform and manages any that it 'owns'. The ea recognises ownership by any method you chose. For me, it is by magic number.

CountOpenTrades() calls any management functions required, along with the vital LookForTradeClosure(int ticket).
{

bool LookForTradeClosure(int ticket)
When you have a trading system that opens a trade when x conditions are met, you also have one that closes said trade when the opposite conditions occur. Here you enter the code that sets CloseThisTrade to 'true' when the opposite conditions occur – plenty of examples in the shell.

Using the shell
You are completely free to use the shell to create ea's for yourself, for distribution at SHF and amongst your friends; you do not need to ask the permission of Tommaso or myself to use it.

Bear in mind that: Bob has given of his expertise freely; Tomasso and myself have given our time freely in creating this shell; many other contributors have given their time freely in creating code the shell uses. You are **not** free to use the shell for your own commercial profit and doing so will have repercussions should I discover that you have done so. In particular, Matt Kennel's liborderreliable is used on the strict understanding that it is not to be used for commercial profit without a license from him. You can contact Matt via SHF should you need advice.

In conclusion

Let's produce some killer ea's.