Program Trading

The FTS Interactive Trader lets you create program trading strategies, as follows:

- You create the strategy in Excel by writing a VBA macro function
- The strategy can depend on your position and current market conditions
- Your function returns a trade order
- The FTS Interactive Trader (Windows version) runs your VBA function and executes the trades as desired

This lets you monitor multiple markets and create virtually any type of strategy you like:

- You can enter bids and asks
- You can enter buy and sell orders
- You can submit multiple orders (e.g. both a bid and ask) for a security
- You can submit orders for multiple securities at the same time

The operation is very simple, and requires just a little understanding of Excel macros and VBA programming. The examples below get you started, but we describe extremely simple trading strategies so we can focus on the mechanics.

You can download an Excel workbook with the example below and from http://www.ftsmodules.com/public/modules/trader/FTSProgramTrading.xlsm. It is also available from the Help menu of the program trading interface of the FTS Trader.

Example

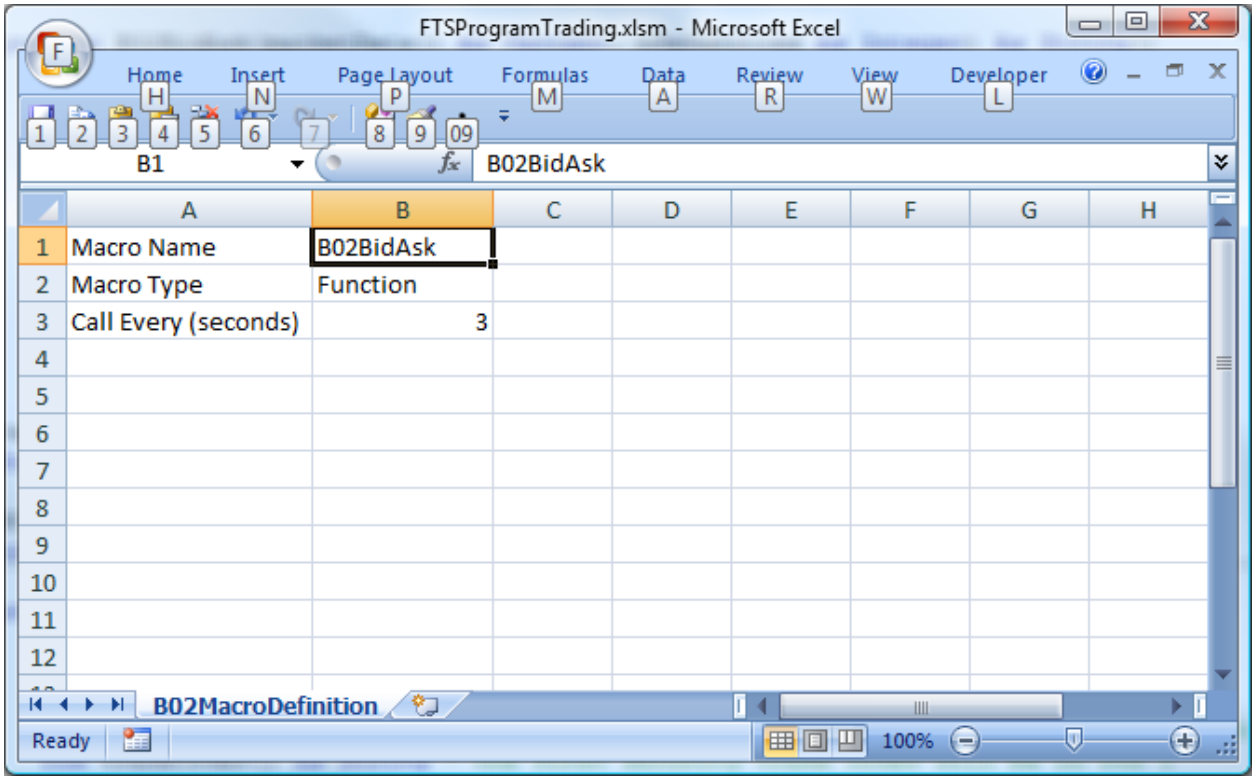
As a simple illustration, consider the following strategy in Case B02:

- In period 1, for the coupon bond:
  - if the bid exceeds 91, sell the quantity available
  - if the bid is below 89, enter a bid of 89 for 10 bonds
  - if the ask is 89 or below, buy the quantity available

- if the ask is above 91, enter an ask of 91 for 10 bonds
  - Note that you could enter both a bid and an ask

Here is what you would do:

1. **Open Excel and specify the function to be called in a worksheet**



- It's self-explanatory, except for the second row: note that the macro type is a "Function."
  - An Excel macro can be a Function or a Sub. To obtain trade orders from a macro, you must write a function. Later, we discuss why you may need a sub.
  - You can have multiple macros. Here, if you had a second macro, it would be specified in column C. Each macro must obviously have a unique name.
    - Different macro's can perform different functions. For example, one macro can focus on market making, another can focus on picking out arbitrage. Putting these in different macros makes it simpler for you by separating the tasks.

2. **Write the macro**

- In the Visual Basic editor (Developer menu item), insert a module (from the Insert menu)
- A function must be specified in a precise way. It takes the form:
    - Function B02BidAsk(marketData() as Variant, nSecurities as integer) as String()
    - The marketData array is sent in to the macro. It contains information as specified below. nSecurities is the number of securities in the case
    - The function return a string array with the trade orders

- Details about all of this are given below, but here is a simple example of a trade function, it just posts a bid and ask depending on the period:

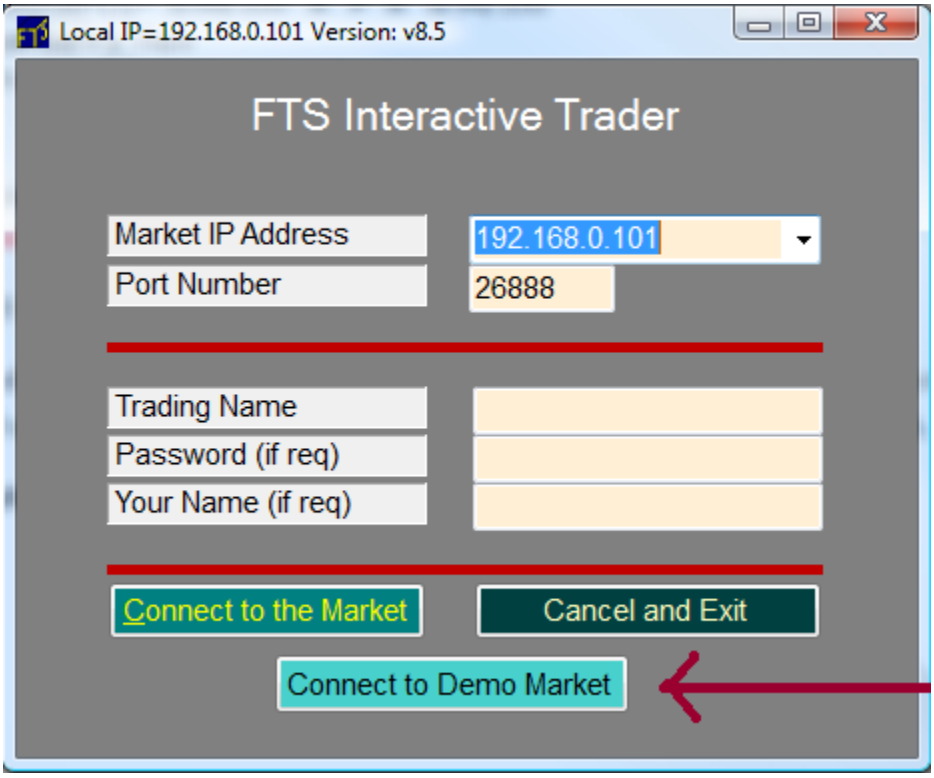
```
Function B02BidAsk(marketData() As Variant, nSecurities As Integer) As String()

  ' define the string array to hold the trade orders.
  Dim tradeOrder() As String ' the first security trade order will be in row 1
  ReDim tradeOrder(nSecurities)

 timeRow = nSecurities + 4  ‘ this is the row with period, trial, time information

  Dim period As Integer
  period = toInteger(marketData(timeRow, cPeriod)) ‘these are explained below

  If period = 1 Then
     tradeOrder(1)=”b/87/100” & “#” & “a/93/100”
  ElseIf period = 2 Then
     tradeOrder(1)=”b/88/100” & “#” & “a/94/100”
 ElseIf period = 3 Then
        tradeOrder(1)=”b/89/100” & “#” & “a/96/100”
End If

 B02BidAsk = tradeOrder 'send back the orders
End Function
```

In line 5 of the macro, we call a function called *toInteger* to calculate the period; it is defined in the workbook and converts cell values to integers.

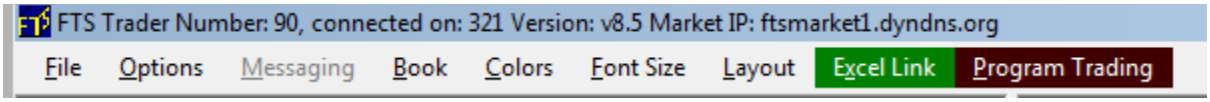
That’s all it takes. Save the workbook, but leave it open in Excel. Incidentally, you can download the workbook with these examples from http://www.ftsmodules.com.

We saved ours with the name FTSProgramTrading.xlsm. The “xlsm” specifies a macro-enabled workbook. You must make sure macros are enabled in your workbook.

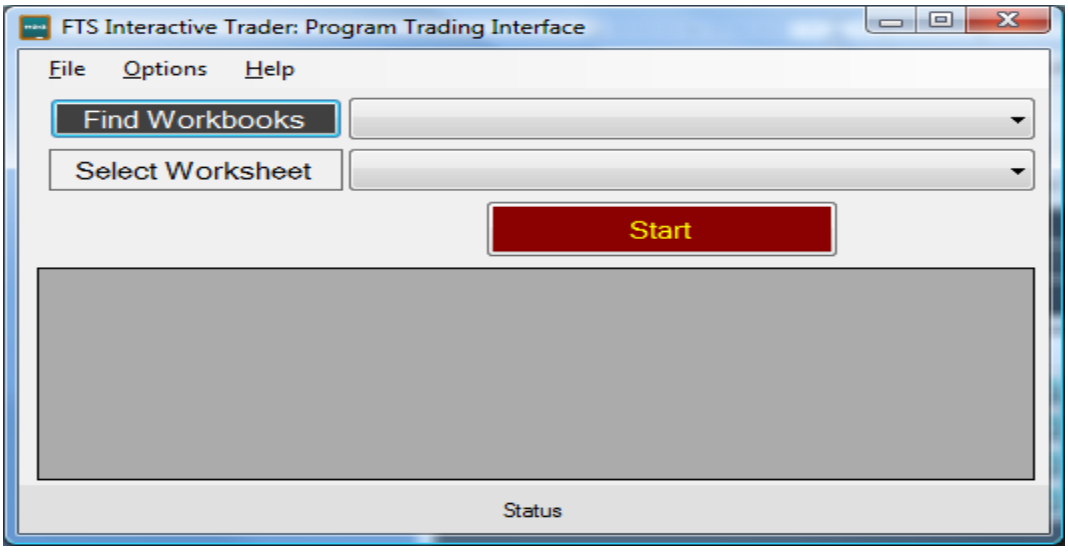
3. Launch the FTS Trader (2013 Version) and connect to the demo market.



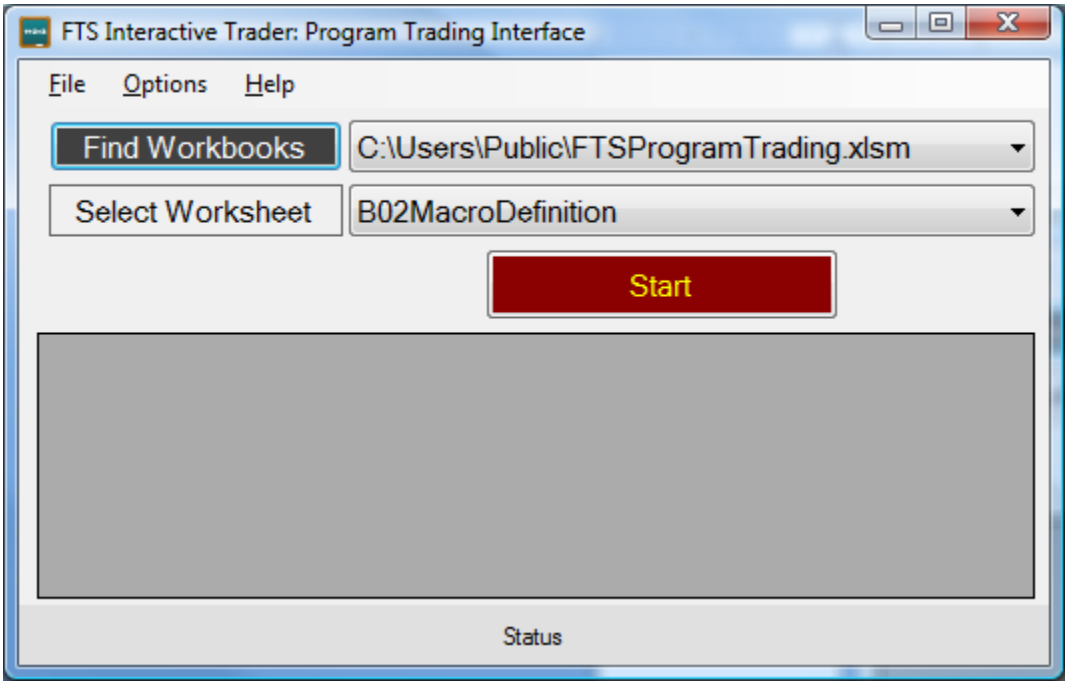
In the main trading window, click Program Trading:



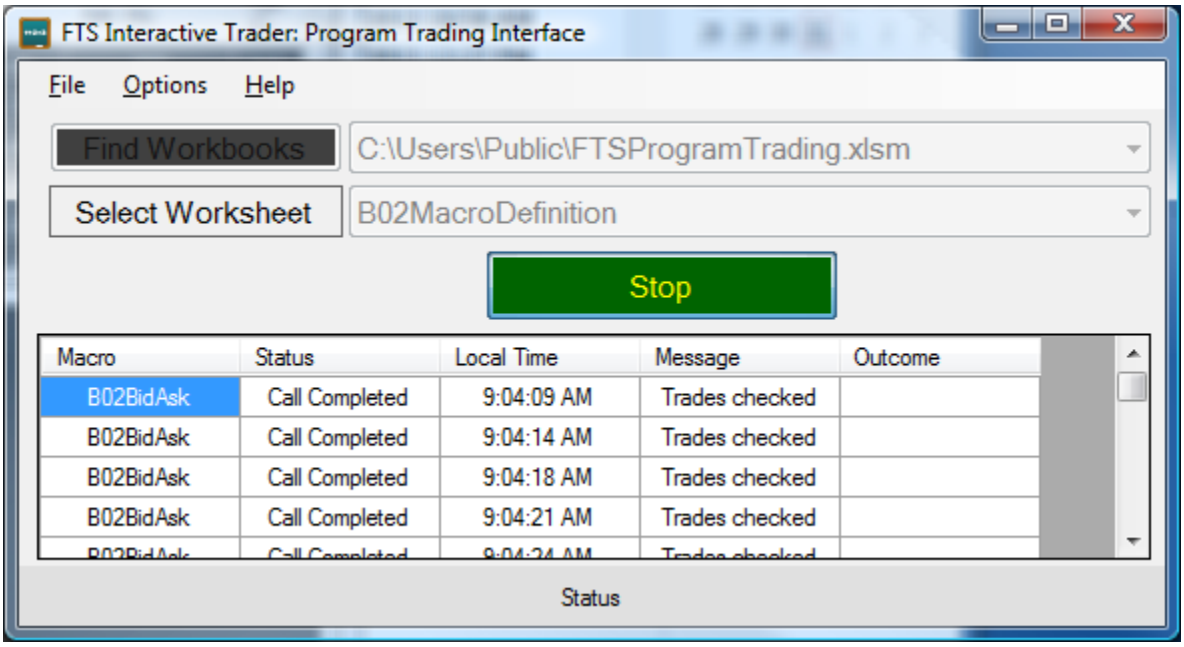
You will see:



Click Find Workbooks and then select the worksheet with the macro definitions:



Click Start, and the macro will run every three seconds. After a few seconds, you will see something like:



That's it.

In the sample workbook, we have a slightly more complex example for case B02.

Extensions

- Using a Sub
    - An excel function cannot read data in a worksheet. So if your trading strategy depends on data that is stored in the spreadsheet, you will have to run a Sub to read the data and store it in a global variable in your module.
- Using the trading history
    - The trading history is available via the Excel link. You can use a Sub to read it in and use it to create a history dependent strategy.

Summary and Caveats

The example shows you the basic steps: how to call functions, how to read data, and so on. Our strategies were simple so we could focus on showing you the basics, but you can probably figure out that you can create fairly complex strategies.

Things to keep in mind

- The point of the system is to help you understand how to develop and implement computerized trading strategies.
- This lets you go beyond manual trade entry to a world of automated trading, so you can explore the world of quantitative strategies that play such an important role in today's markets
- Remember that in the FTS Interactive Markets, others will react to your actions. It is important to keep this in mind and think about how to adjust your strategy if things do not go as planned.
- Do not "block" the execution of the macro, with things like stop statements. The Trader will not be able to continue until the macro has completed.
- Trap errors using "On Error" statements. Otherwise, the macro will stop at the point of the error and not be able to continue.
- You should know about "time leakage." Suppose you want the macro called every 5 seconds. But your macro takes 1 second to complete its task. The five seconds countdown will start after the macro completes its task, so the next call will actually be 6 seconds after the first time it was called.
- You have to be connected to the market for the program trading to work
    - If you lose the connection (in which case your strategy will stop) if there is an internet hiccup, if your computer (specially a laptop) goes to sleep, you should reconnect and then restart the strategy.
- You should test your strategy otherwise you could lose a bundle due to a coding error

Appendix: The structure of the MACRO and the marketData array

Function MACRONAME(marketData() As Variant, nSecurities As Integer) As String()

- marketData() is an array with the following:
  - **ROWS**
    - Row 0 is just headers, like “Bid”, “Ask”
      - It has exactly the same information as in the Excel link
    - Row 1 has data on Security 1
    - Row 2 has data on Security 2…up to security nSecurities
    - Row nSecurities +1 has data on Cash
    - Row nSecurities+2 is blank
    - Row nSecurities+3 has headers, like “Time,”Trial” etc.
    - Row nSecurities +4 has the time remaining, trial, period, etc.
  - **COLUMNS:** Put these at the top of your module so you can access the data easily
    - Public Const cName As Integer = 0
    - Public Const cBid As Integer = 1
    - Public Const cBidQ As Integer = 2
    - Public Const cAsk As Integer = 3
    - Public Const cAskQ As Integer = 4
    - Public Const cUnits As Integer = 5
    - Public Const cPayoffs As Integer = 6
    - Public Const cLast As Integer = 7
    - Public Const cLastQ As Integer = 8
    - Public Const cVWAP As Integer = 9
    - Public Const cFutOb As Integer = 10
    - Public Const cBuyVWAP As Integer = 11
    - Public Const cSellVWAP As Integer = 12
    - Public Const cMarginCash As Integer = 13
    - Public Const cMarginQty As Integer = 14
    - Public Const cMarginLoan As Integer = 15
    - Public Const cLastInfo As Integer = 16
    - Public Const cTotVol As Integer = 17
    - ‘column for the Cash row
    - Public Const cCash As Integer = 1
    - 'columns for the time/period/trial information
    - Public Const cTimeLeft As Integer = 0
    - Public Const cTotalTime As Integer = 1
    - Public Const cTrial As Integer = 2

        - Public Const cPeriod As Integer = 3
        - Public Const cRiskFree As Integer = 4

Your macro **must** define:

- Dim tradeOrder () As String
- ReDim tradeOrder (nSecurities)
- Each element of tradeOrder has the format:
    - action/price/qty
        - action is one of: b a y s for bid ask buy sell
        - Price and qty are numbers
        - This is for entering bids and asks and market limit orders
    - Or action/qty
        - Action is one of y s for buy and sell
        - Qty is a number
        - This is for entering market orders
- You can enter multiple orders for a security. The orders must be concatenated and separated by #
    - For example, “b/10/99#a/12/75” will submit a bid and an ask
- Any order that is valid will be executed.
- This allows you to trade baskets of securities.
- tradeOrder(0) is an output from the function that is displayed on the screen.