Algorithmic Trading

The FTS Real Time System lets you create algorithmic 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 and recent prices and volumes
- It can also depend on historical data or any other data that you can access
- Your function returns a trade order
- The FTS Real Time Client (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, including:

- Contingent orders
- Basket trading
- Pairs trading and other statistical arbitrage strategies
- Dynamic trading strategies
- Optimal trade execution strategies
- Inter-market trading
- Tracking

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 the Excel workbook with the examples by clicking on:
FTSAlgoTrading.xlsm

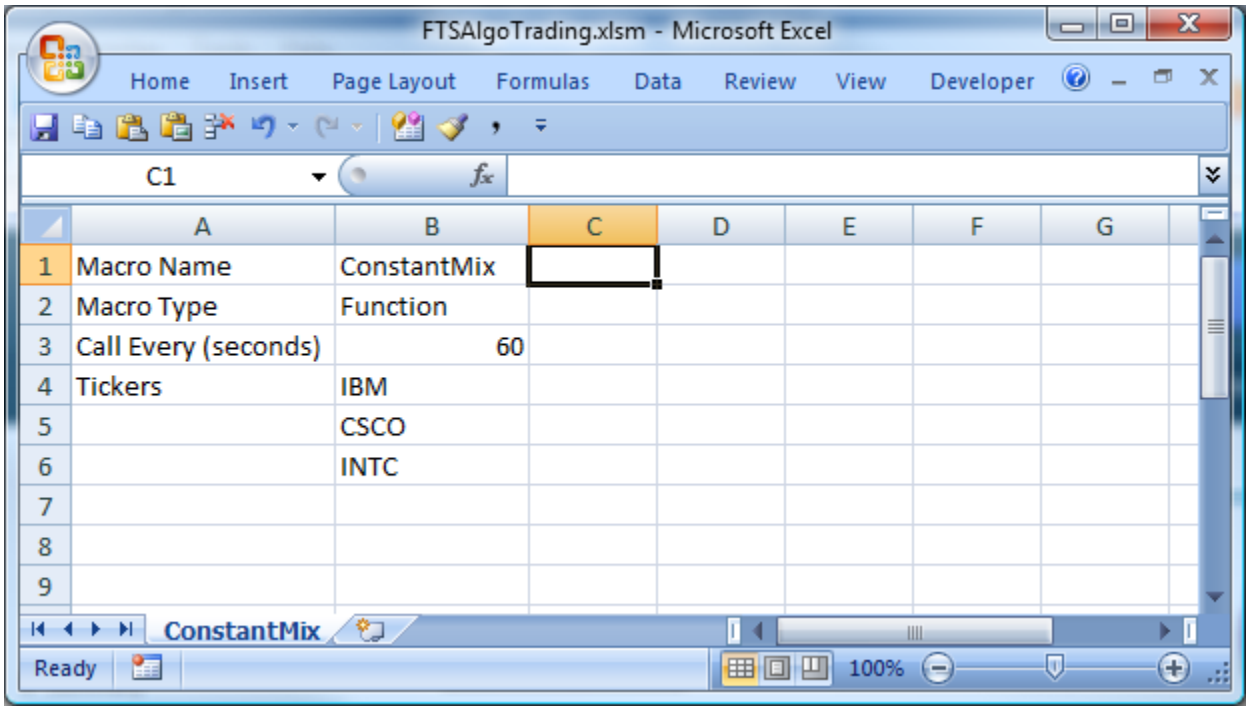
Example 1

As a simple illustration, consider the following "constant mix" strategy in the FTS 1000 Stock case:

- Monitor 3 tickers: IBM, CSCO, and INTC.
- Every 60 seconds, make sure that the dollar amount invested in each is (within rounding) equal to $10000
    - This is called a "constant mix strategy"
    - The rounding occurs because the quantity we trade is an integer, and so you may not get an exact number of shares to buy

Here is what you would do:

1. **Open Excel and specify the function and tickers 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. In example 2, we will show you how to and why you may also want to use a sub.
    - You can have multiple macros. We will illustrate this in example 2.

- The number of tickers is read from this sheet; in this case, there are 3 tickers. The order of the tickers is important. Tickers are processed in the macro in the same order as in this sheet

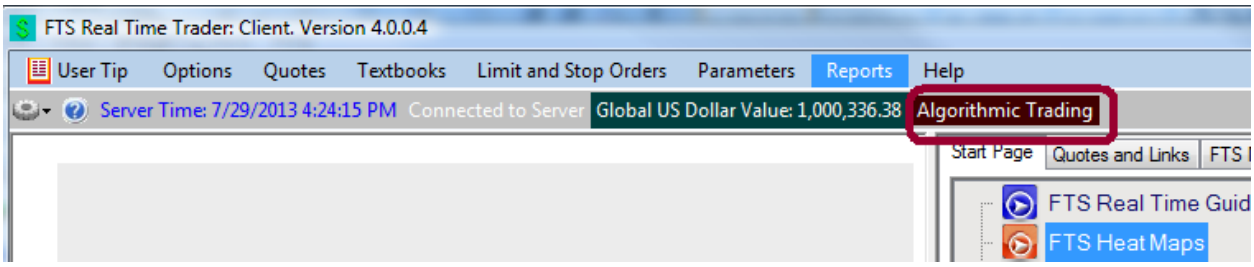
2. **Write the macro**
   - In the Visual Basic editor (Developer menu item), insert a module (from the Insert menu)
   - Enter the following into the module. You can copy and paste, but be careful because line breaks when pasting cause errors; it's better to download the workbook from our site.

```
Function ConstantMix(bids() As Single, asks() As Single, last() As Single, volume() As Single, qty() As Single, serverTime As Date) As String()

Dim n As Integer, T As Integer
n = UBound(bids, 1) ' n is the number of tickers, in the same order as in the worksheet
T = UBound(bids, 2) ' T is the number of recent historical prices

' all the array's start at 1

Dim returnString() As String 'this is what you send back to the FTS Real Time Client
ReDim returnString(n) ' one trade order for each ticker; it could be blank

Dim dollarAmount As Single
dollarAmount = 10000

Dim i As Integer, nTrade As Integer
For i = 1 To n
  If (qty(i, 1) * last(i, 1) < dollarAmount And last(i, 1) > 0) Then
    nTrade = dollarAmount / last(i, 1) - qty(i, 1)
    returnString(i) = "cashbuy/" & CStr(nTrade)
  ElseIf (qty(i, 1) * last(i, 1) >= dollarAmount) Then
    nTrade = qty(i, 1) - dollarAmount / last(i, 1)
    returnString(i) = "cashsell/" & CStr(nTrade)
  End If
Next
ConstantMix = returnString
```

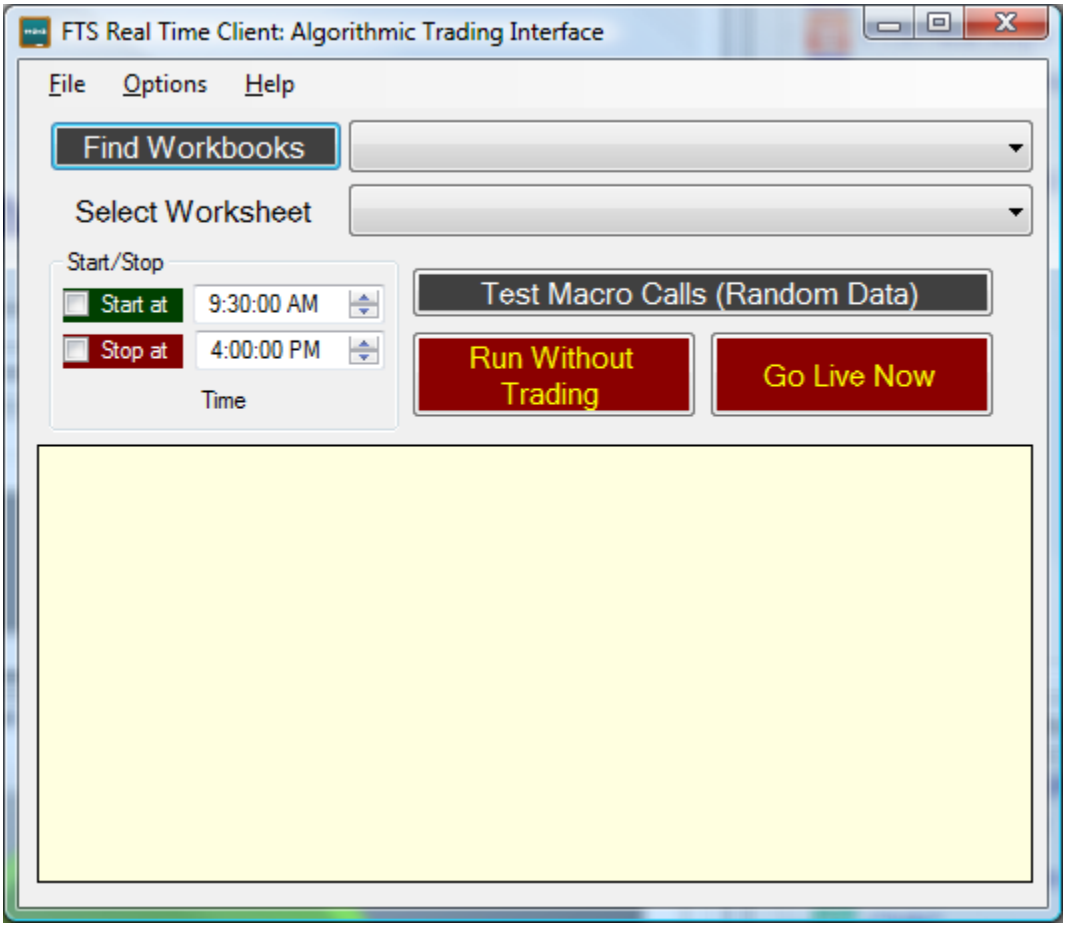
End Function

That's all it takes. Save the workbook, but leave it open in Excel. We saved ours with the name FTSAgloTrading.xlsm. The "xlsm" specifies a macro-enabled workbook.

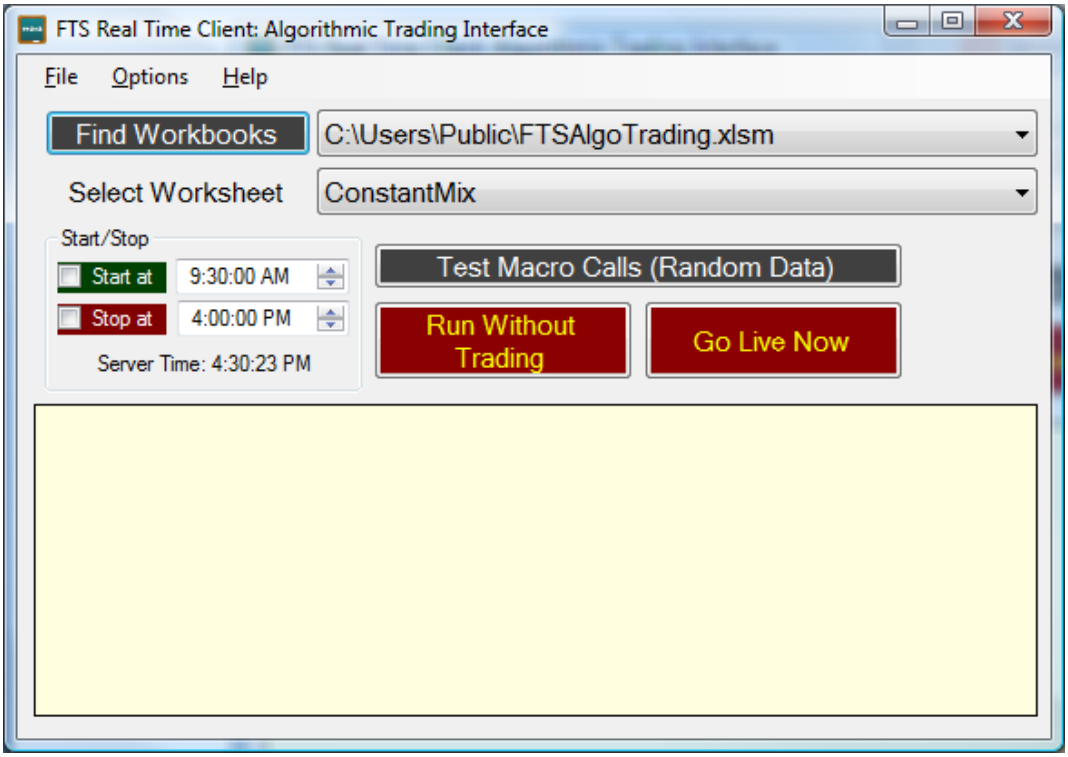
3. Launch the FTS Real Time Client, and log in after selecting the FTS 1000 Stock Case (you must have a trading account set up for this case). At the top, you will see the Algorithmic Trading button:



Click the Algorithmic Trading button to get:



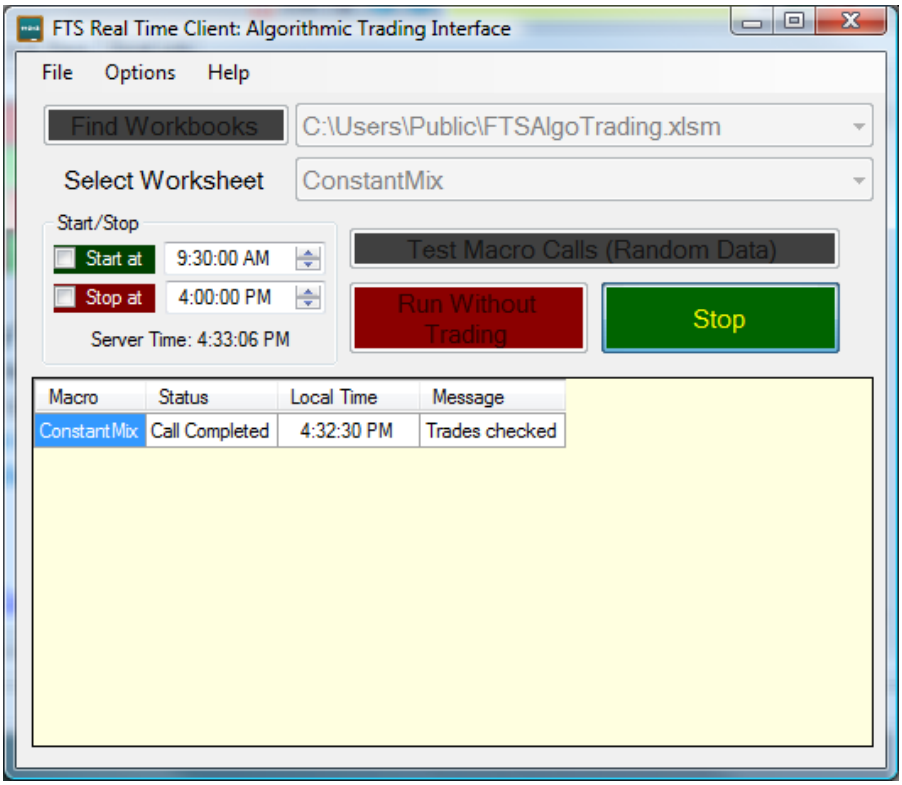
Click **Find Workbooks**. The workbooks that are open in Excel will be listed. Select the workbook and in the next dropdown, select the worksheet with your macro definition:



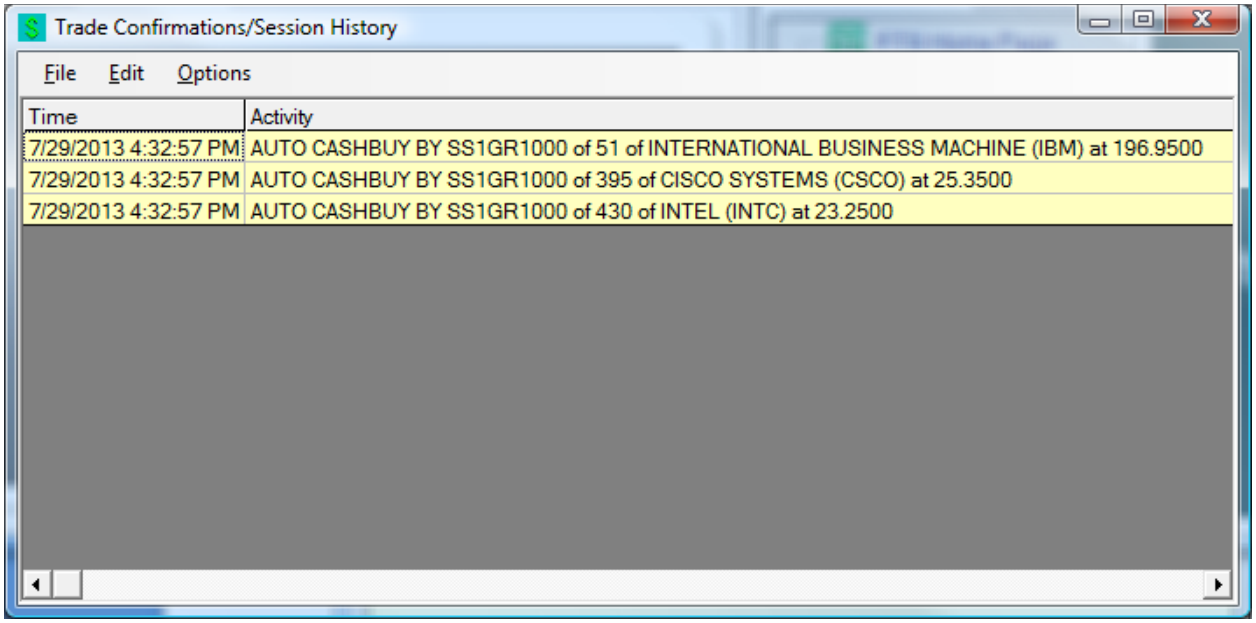
Notes:

- The worksheet is read in when you select the worksheet. This defined the number of tickers, the order of the tickers, the macro names, and so on.
- If you make changes to the worksheet *after* you have selected the worksheet, you have to click **Find Workbooks** again and re-select the worksheet.

Click **Go Live Now.** The macro will be implemented. Here is what happened to us:



The macro was run at 4:32:30 PM, and we received the following trade confirmations:



Our position display shows us:

Stocks	Position	Last	Last Value
CISCO SYSTEMS (CSCO)	395	25.33	10,005.35
INTEL (INTC)	430	23.24	9,993.20
INTERNATIONAL BUSINESS MACHINE (IBM)	51	196.21	10,006.71

You can see that we have invested approximately $10000 in each stock. To get exactly $10000 in INTC, you would have to buy 430.2926 shares, we bought 430 shares.

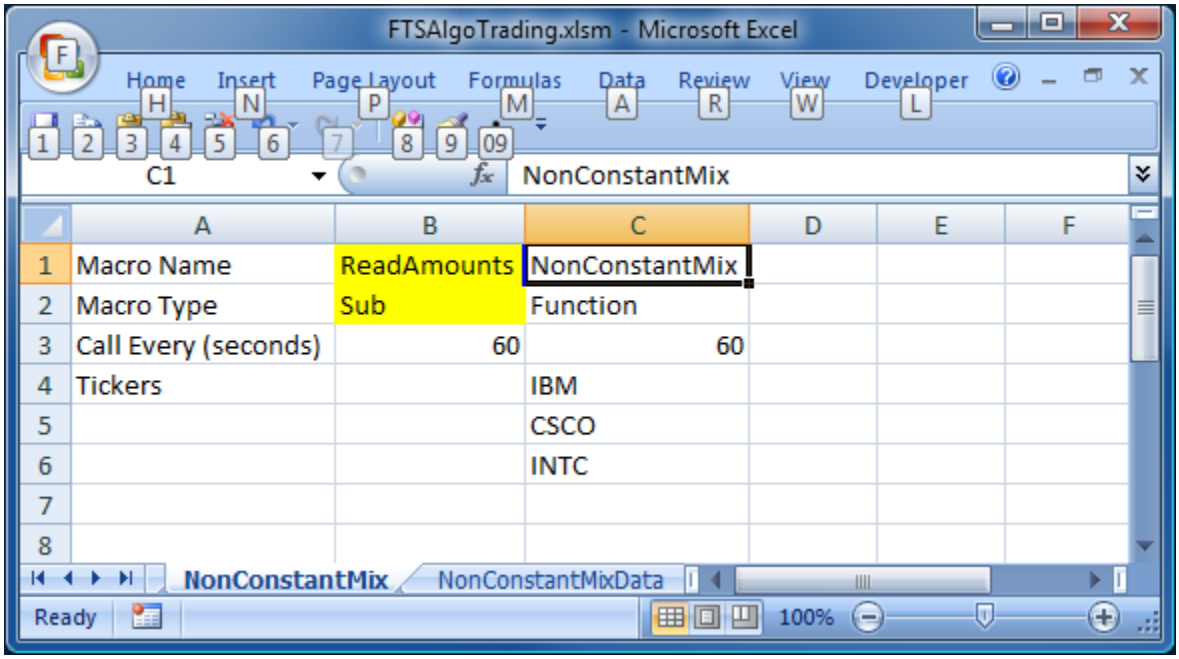
Now, every 60 seconds, the FTS Real Time Client would run the macro again and trade if necessary as calculated by the trading algorithm.

That's it.

Example 2:

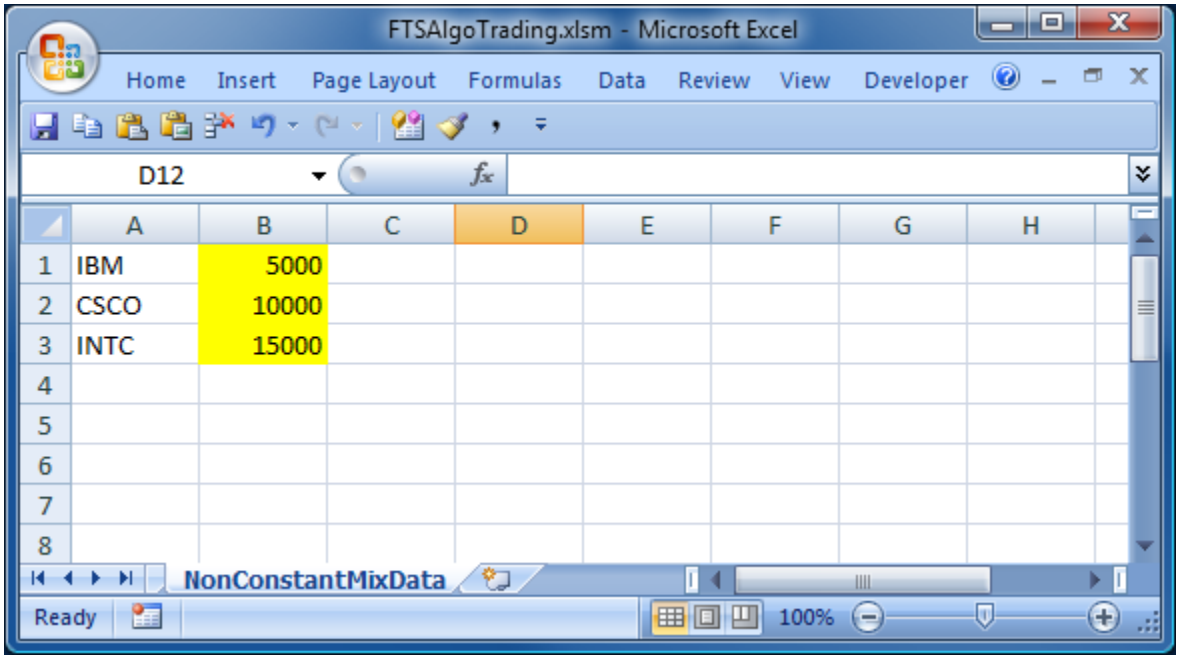
Let's modify the example a little bit: suppose that we don't necessarily want an equal amount invested in each, but instead, we want different amounts, and we want to specify the amounts in a worksheet. *This will illustrate how you use data stored a worksheet in your trading algorithm.* While our example is simple, the method illustrates you can use any data, such as historical data, in your strategy.

Create a new sheet, let's call it NonConstantMix, it has 2 macros:



The yellow parts show you the new information: we have a sub called ReadAmounts. This will also be called every 60 seconds, but it will be called ***before*** the function NonConstantMix. In

the sub, we will read the level of each investment; that information is in the sheet NonConstantMixData:



The levels are specified: 5000 in IBM, 10000 in CSCO, 15000 in INTC. *Note that this data could be anything that you want.*

Insert a module and enter (copy/paste) the following:

```
Option Explicit
Dim dollarAmount(3) As Single

Sub ReadAmounts()
dollarAmount(1) = Worksheets("NonConstantMixData").Cells(1, 2).Value ' cell B1
dollarAmount(2) = Worksheets("NonConstantMixData").Cells(2, 2).Value ' cell B2
dollarAmount(3) = Worksheets("NonConstantMixData").Cells(3, 2).Value ' cell B3
Worksheets("NonConstantMixData").Cells(4, 2).Value = Now
End Sub

Function NonConstantMix(bids() As Single, asks() As Single, last() As Single, volume() As
Single, qty() As Single, serverTime As Date) As String()
Dim n As Integer, T As Integer
n = UBound(bids, 1) ' n is the number of tickers
T = UBound(bids, 2) ' T is the number of recent historical prices, takes at 15 second intervals for
the past 30 minutes or since you logged in

' all the array's start at 1
```

```
Dim returnString() As String
ReDim returnString(n) 'this is where trade orders get specified

Dim i As Integer, nTrade As Integer
For i = 1 To n
   If (qty(i, 1) * last(i, 1) < dollarAmount(i) And last(i, 1) > 0) Then
      nTrade = dollarAmount(i) / last(i, 1) - qty(i, 1)
      returnString(i) = "cashbuy/" & CStr(nTrade)
   ElseIf (qty(i, 1) * last(i, 1) >= dollarAmount(i) And last(i, 1) > 0) Then
      nTrade = qty(i, 1) - dollarAmount(i) / last(i, 1)
      returnString(i) = "cashsell/" & CStr(nTrade)
   End If
Next
returnString(0) = "strategy checked"
NonConstantMix = returnString
End Function
```

Here is what will happen:

- Every 60 seconds, **Sub ReadAmounts** will be called
- Immediately after that, **Function NonconstantMix** will be called
    - Any resulting trades will be transacted

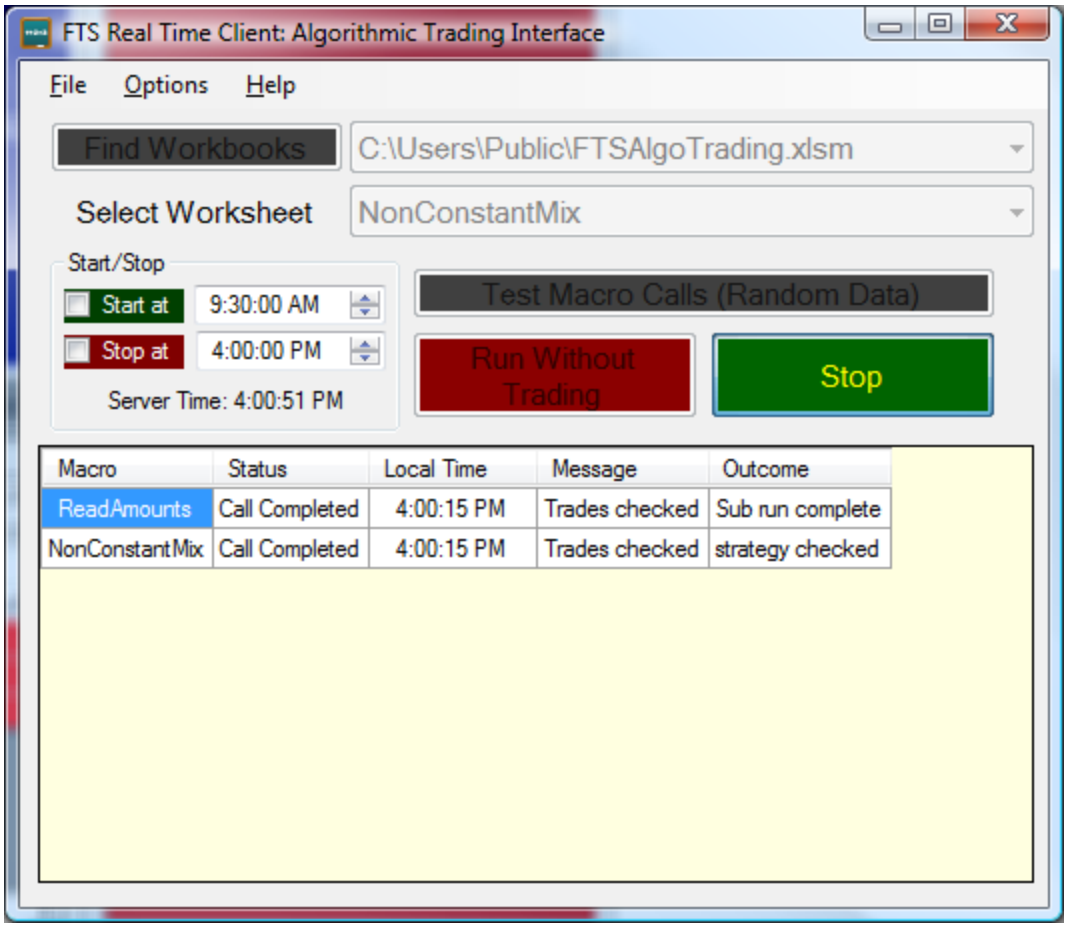
In the module, we have declared a global variable called dollarAmount(3). **Sub ReadAmounts** reads in these numbers from the worksheet *NonConstantMixData*. It also writes out the time at which the sub was called.

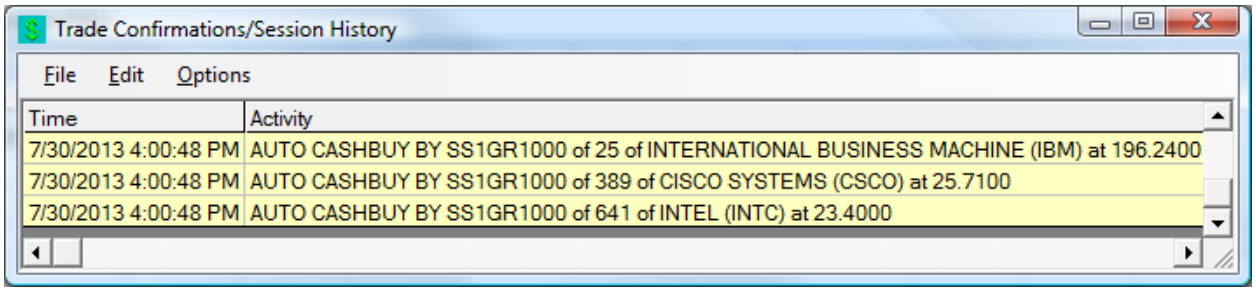
- Note: you can change these levels at any time by typing in new numbers; since the amounts are read in every 60 seconds, the next run of the macro will be based on the new amounts.
- You could also have external data, like historical data or news or whatever you need and read it in. You could even have a web query that is run that download data from web sites and read in that data.
- This note illustrates how you change parameters in a worksheet.

Function NonConstantMix is similar to the previous one, so we won't describe it further. Note, however, the line:
returnString(0) = "strategy checked"
near the bottom. Whatever you put into returnString(0) is displayed on the screen after the macro is run.

Details about how you must define the function that create sthe trade orders are in the appendix. You must follow the exact call sequence.

Here is what happened when we ran it:

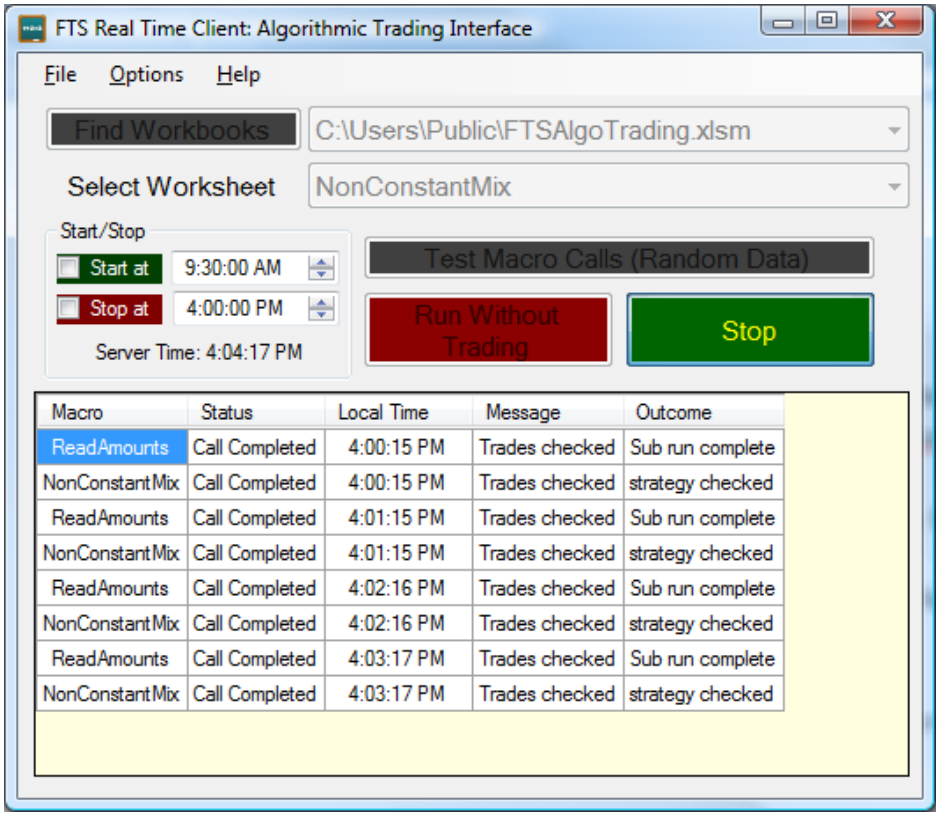




And our position was displayed as:

Currency	Amount Available	Borrowed	Credit Left
US Dollar	970,072.90	0.00	100,000.00
Stocks	**Position**	**Last**	**Last Value**
CISCO SYSTEMS (CSCO)	389	25.72	10,003.13
INTEL (INTC)	641	23.41	15,002.61
INTERNATIONAL BUSINESS MACHINE (IBM)	25	196.12	4,903.00

After a few minutes:



Extensions

Now, you know how to do 3 things:

1. How to create trade orders by creating a *function* called by the FTS Real Time Client with the specified arguments
2. How to use data stored in the workbook by using a *sub* and storing the data in global variables that can then be used by the function.
3. How to run multiple sub's and functions in sequence

You can now build any strategy you like:

- **Recent market data**: this is easy. The trade function can depend on the bid, ask, and last prices over the past 30 minutes of the tickers you specify.
- **Historical data**: you simply bring it into a worksheet, read it in to a global variable, and calculate what you need in a sub. Then, use the calculated results in the trade function.
- **Non-price data:** you can bring data on company fundamentals, even textual data that you interpret, such as news.
- **Other Excel add-ins**: since you have full access to VBA and the workbook, you can use any add-in. Just make sure that the add-in finishes its calculations before you call the trade function, i.e. give it enough time. You can call the add-in from the worksheet or, if permitted, from VBA.

- **Other programming languages:** you can create a DLL (dynamic link library) and call functions from the DLL in VBA. If you have large and complex calculations, these can be much faster than VBA.

The two examples show 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.

Example strategies

- **Contingent orders**: an order is executed only if one or more conditions are satisfied. Ultimately, any trading strategy results in a contingent order!
    - For example, you are interested in buying stock 1 or stock 2 but not both. So you specify a limit price on each in your function, and issue a buy order for the first stock when the limit price is reached.
- **Rebalancing**: like our constant mix example. Another example is re-allocating money across two different stock markets based on exchange rate movements.
- **Dynamic trading strategies and hedging**: buy or sell options or futures depending on stock price movements.
- **Basket trading**: buy a group of securities all at once when some conditions are met.
- **…….**

Summary and Caveats

- The point of the system is to help you understand how to develop and implement algorithmic trading strategies.
- This lets you go far beyond simple limit and stop orders and manual trade entry to the world of automated trading, so you can explore the world of quantitative strategies that play such an important role in today's markets
- This is an educational system, not designed as a high frequency trading system where you conduct a large number of trades every day. The idea is to focus on thinking about and implementing and testing strategies.
- You have to leave the real time client running and connected to our server for your algorithm to work
- If you start your algorithm, you should check periodically that you are connected to the server.
    - You can 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, or, on very rare occasions, when we have to restart our server during the trading day.
- You should test your strategy otherwise you could lose a bundle due to a coding error

Appendix: The structure of the trade function

Function trade(bids() As Single, asks() As Single, last() As Single, volume() As Single, qty() As Single, serverTime As Date) As String()

- ' n is the number of tickers
- ' T is the recent historical prices, takes at 15 second intervals for the past 30 minutes or since you logged in
- ' all the array's start at 1

- ' bids, asks, last, volume have the following structure:
- ' bids(1,1) = current bid on ticker 1
- ' bids(1,2) = the 15-second ago bid on ticker 1
- ' bids(1,3) = the 30-second ago bid on ticker 2....
- ' bids(1,T) = the oldest available bid on ticker 1
- ' bids(2,1) = current bid on ticker 2
- ' and so on

- ' The order of the tickers is as specified in the spreadsheet.

- 'qty is an n x 6 array. It has the following structure:
- ' note that every ticker has two things associated with it: a position array and the cash of the currency it is traded in.
- ' Further, a position could be a cash purchase, a margin trade, or a short position.
- ' To give you maximum flexibiity and to also handle currency trades easily, we send the quantity as follows:
- ' for anything other than an exchange rate: for the i'th element,
- ' q(i,1) = cash purchases; q(i,2)=margin purchases; q(i,3)=short position; q(i,4)=cash holding of the currency that the ticker trades in
- ' Example: you did a cash purchase of 10 shares of IBM and a margin purchase of 7 shares of IBM and you have $100 in cash: q(i,1)=10 q(i,2)=7 q(i,3)=0 q(i,4)=100
- ' Example: you short sold 23 shares of IBM and you have $700 in cash: q(i,1)=0 q(i,2)=0 q(i,3)=23 q(i,4)=700
- ' q(i,5) = VWAP of your trades
- ' q(i,6) = transaction cost per trade
- ' for an exchange rate:

- '	q(i,1) = position in the NUMERATOR CURRENCY, q(i,2) = position in the DENOMINATOR currency
- '	Example: The ticker BGBPUSD for the British pound is quoted as US$ per pound. The NUMERATOR is US$ the DENOMINATOR is Pound
- '	Example: the ticker BUSDJPY for the Japanese Yen is quoted as Yen per US$. The NUMERATOR is Yen the DENOMINATOR is US$.
- '	q(i,3) is the VWAP for the numerator currency
- '	q(i,4) is the VWAP of the denominator currency
- '	q(i,5) is not used
- '	q(i,6) is the transaction cost per trade
-
- Dim returnString() As String
- ReDim returnString(n)
- ' each element of returnString has the format: action/qty, going from 1 to n
- ' action is one of: cashbuy cashsell marginbuy marginsell shortsale shortcover
- ' qty is a number.
- ' Example: returnString(i)="cashbuy/10" means exactly what it says: buy 10 shares for cash.
- ' any order that is valid will be executed. So if you have cashbuy orders for multiple securities, they will all be executed at once.
- ' This allows you to trade baskets of securities.
- returnString(0) is an output from the function that is displayed on the screen.