

Introduction

This tutorial will help traders program a complete high-frequency trading market making algorithm for the ALGO2 case.

Tutorial

Begin by linking your spreadsheet to basic RIT datafields via the RTD link function. You should be familiar with the function of the RTD links from previous spreadsheets. It is helpful to have fields of data set up such as the following:

Broker, Period, Cash, Buying Power, Net Liquidation Value, Risk Free Rate, Time Remaining, Time in a Year, and Profit.

We have set them up in the following spreadsheet. (Please note that the column B is live linked Data, and Column C describes the equations used in column B to link the data. i.e., your Column C should be completely blank.

Code		Controls		
A1		Case Data		
	A	B	C	D
1	Case Data			
2	Broker	[error]	<---=RTD("rit", "NAME")	
3	Period	[error]	<---=RTD("rit", "CP")	
4	Cash	[error]	<---=RTD("rit", "CASH")	
5	Buying Power	[error]	<---=RTD("rit", "BP")	
6	Net Liq. Value	[error]	<---=RTD("rit", "NLV")	
7	Risk Free Rate	[error]	<---=RTD("rit", "RF")	
8	Time Remaining	[error]	<---=RTD("rit", "TR")	
9	Time in a Year	[error]	<---=RTD("rit", "TY")	
10	Profit	#VALUE!	<---=B6-500000	
11				

If a case is actively running, you should see values in the fields. If no case is running, it should show "[error]". If it shows #N/A then you have either typed the equation incorrectly, or do not have the RTD data patch (provided by Microsoft) installed properly.

Next, add the following hard-coded parameters to your spreadsheet: The Column B section of the parameters (rows 13-16) are all numbers input by the user.

B15			fx	35
	A	B	C	D
1	Case Data			
2	Broker	[error]	<---=RTD("rit", "NAME")	
3	Period	[error]	<---=RTD("rit", "CP")	
4	Cash	[error]	<---=RTD("rit", "CASH")	
5	Buying Power	[error]	<---=RTD("rit", "BP")	
6	Net Liq. Value	[error]	<---=RTD("rit", "NLV")	
7	Risk Free Rate	[error]	<---=RTD("rit", "RF")	
8	Time Remaining	[error]	<---=RTD("rit", "TR")	
9	Time in a Year	[error]	<---=RTD("rit", "TY")	
10	Profit	#VALUE!	<---=B6-500000	
11				
12	Parameters			
13	Spread	0.02		
14	TriggerStart	290		
15	TriggerStop	35		
16	Shares	10000		
17				

These parameters will govern how the algorithm will act, they define:

When the algorithm starts to trade, when the algorithm stops trading, how much of a spread to submit when submitting buy/sell orders, and how many shares to buy/sell when submitting an order.

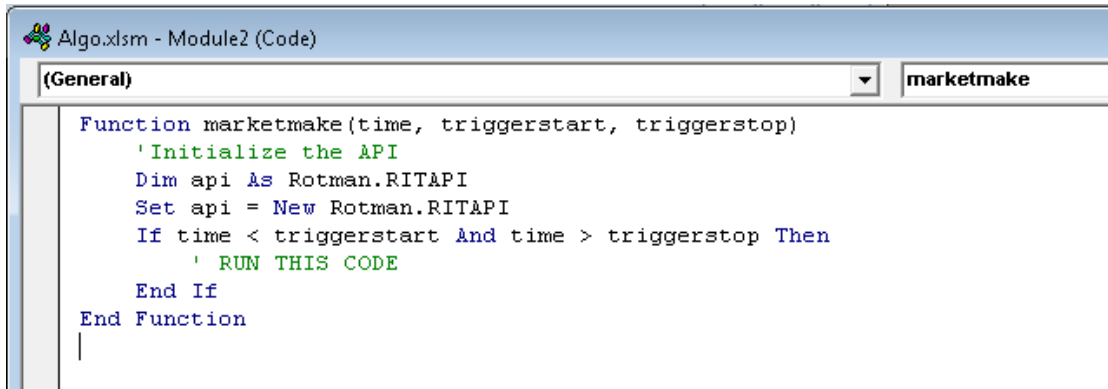
Lastly, we need to link the market data:

G22				fx						
	A	B	C	D	E	F	G	H	I	J
1	Case Data									
2	Broker	[error]	<---=RTD("rit", "NAME")		Market Data					
3	Period	[error]	<---=RTD("rit", "CP")		Stock	Algo	<-- Algo			
4	Cash	[error]	<---=RTD("rit", "CASH")		Position	[error]	<-- =RTD("rit", "ALGO POSN")			
5	Buying Power	[error]	<---=RTD("rit", "BP")		Bid	[error]	<-- =RTD("rit", "ALGO BID")			
6	Net Liq. Value	[error]	<---=RTD("rit", "NLV")		Ask	[error]	<-- =RTD("rit", "ALGO ASK")			
7	Risk Free Rate	[error]	<---=RTD("rit", "RF")		Midmarket	#DIV/0!	<-- =ROUND(AVERAGE(F5,F6),2)			
8	Time Remaining	[error]	<---=RTD("rit", "TR")							
9	Time in a Year	[error]	<---=RTD("rit", "TY")							
10	Profit	#VALUE!	<---=B6-500000			0				
11										
12	Parameters									
13	Spread	0.02								
14	TriggerStart	290								
15	TriggerStop	35								
16	Shares	10000								
17										

Now that we have all of that data setup in Excel, we can start to program our algorithm.

The Algorithm

We begin our algorithm by going into the VBA editor and adding a new module. In the module, we declare a function called “marketmake” with parameters “time, triggerstart, and triggerstop”. The function initializes the API and then compares the current time to the start/end times and runs specific code “if” the time is between the start/stop times.

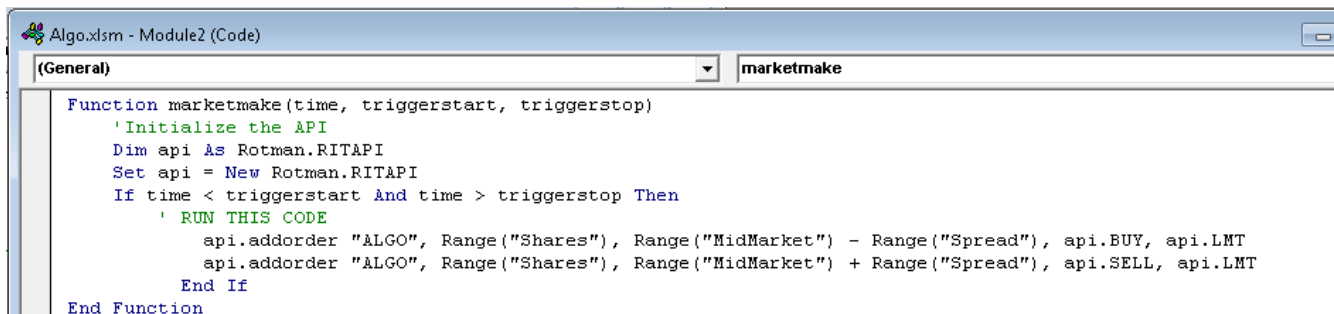


```
Algo.xlsm - Module2 (Code)
(General) marketmake

Function marketmake(time, triggerstart, triggerstop)
    ' Initialize the API
    Dim api As Rotman.RITAPI
    Set api = New Rotman.RITAPI
    If time < triggerstart And time > triggerstop Then
        ' RUN THIS CODE
    End If
End Function
```

Next, we need to instruct the function as to what to do when it reaches the “RUN THIS CODE” portion. We want it to submit two orders, a bid at a price of <Midmarket> – <spread> for <volume>, and an offer at a price of <midmarket> + <spread> for <volume>. <sections> denote links to the spreadsheet variables.

This can be accomplished as follows:



```
Algo.xlsm - Module2 (Code)
(General) marketmake

Function marketmake(time, triggerstart, triggerstop)
    ' Initialize the API
    Dim api As Rotman.RITAPI
    Set api = New Rotman.RITAPI
    If time < triggerstart And time > triggerstop Then
        ' RUN THIS CODE
        api.addorder "ALGO", Range("Shares"), Range("MidMarket") - Range("Spread"), api.BUY, api.LMT
        api.addorder "ALGO", Range("Shares"), Range("MidMarket") + Range("Spread"), api.SELL, api.LMT
    End If
End Function
```

As you can see, the spreadsheet refers to ranges instead of cell locations, for example instead of saying Cells(16,2) for volume, it says Ranges(“Shares”). This is because in the Excel table we have labelled the different cells with range names. This can be accomplished by going back to the spreadsheet, clicking on each of the specific cells, and filling out the range name in the top left corner. The following is the example for labelling the volume. We suggest you label the Midmarket price, as well as the spread.

	Shares			10000				
	A	B	C	D	E	F	G	H
1	Case Data							
2	Broker	[error]	<---=RTD("rit", "NAME")		Market Data			
3	Period	[error]	<---=RTD("rit", "CP")		Stock	Algo	<-- Algo	
4	Cash	[error]	<---=RTD("rit", "CASH")		Position	[error]	<-- =RTD("rit", "ALGO	
5	Buying Power	[error]	<---=RTD("rit", "BP")		Bid	[error]	<-- =RTD("rit", "ALGO	
6	Net Liq. Value	[error]	<---=RTD("rit", "NLV")		Ask	[error]	<-- =RTD("rit", "ALGO	
7	Risk Free Rate	[error]	<---=RTD("rit", "RF")		Midmarket	#DIV/0!	<-- =ROUND(AVERAG	
8	Time Remaining	[error]	<---=RTD("rit", "TR")					
9	Time in a Year	[error]	<---=RTD("rit", "TY")					
10	Profit	#VALUE!	<---=B6-500000			0		
11								
12	Parameters							
13	Spread	0.02						
14	TriggerStart	290						
15	TriggerStop	35						
16	Shares	10000						
17								
18								

Note the upper-left yellow section. It used to say "B16", and it has been relabelled to "Shares". We can now refer to that cell in VBA as "Shares" anytime we want. Reminder: Re-label the other sections Midmarket and Spread.

The algorithm currently checks to see if the time is appropriate, if it is, it automatically submits a limit buy order and a limit sell order. There are a couple of problems with this algorithm. First of all, it will submit an infinite number of orders (keep submitting bids/offers over and over again) as long as the time is between start/stop time. Second, RIT only accepts orders at a certain speed (once every 0.25 seconds), and this algorithm is submitting them nearly instantaneously. To fix that, we alter the code as follows:

```

Algo.xlsm - Module2 (Code)
(General) marketmake

Function marketmake(time, triggerstart, triggerstop)
    'Initialize the API
    Dim api As Rotman.RITAPI
    Set api = New Rotman.RITAPI
    If time < triggerstart And time > triggerstop Then
        ' RUN THIS CODE
        ' The following loop submits the Buy section of bracket
        Status = False
        Do While Status = False
            Status = api.addorder("ALGO", Range("Shares"), Range("MidMarket") - Range("Spread"), api.BUY, api.LMT)
        Loop
        ' The following loop submits the Sell section of bracket
        Status = False
        Do While Status = False
            Status = api.addorder("ALGO", Range("Shares"), Range("MidMarket") + Range("Spread"), api.SELL, api.LMT)
        Loop
    End If
End Function

```

The RIT software returns a string called "TRUE" or "FALSE" when you submit a trade request. If the order is successful, it returns "TRUE", otherwise it returns "FALSE". By setting the "Status" variable as false, and then running the addorder command, you can track when it properly submits the order. The program doesn't exit the loop (and keeps trying to submit the order) until it is successful.

*Note: this could potentially crash your program if it's never able to submit the command! (It gets stuck in the loop forever- consider ways to fix this potential issue.

We need to add a pause to our application to make sure that it isn't running too quickly and getting confused. We can do that by adding the following sections of code to our program:

```

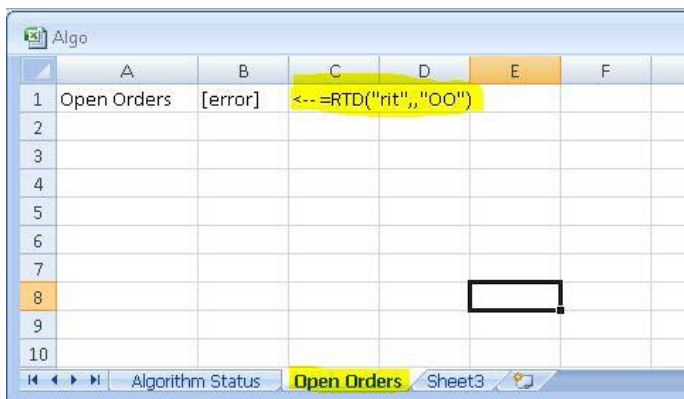
Private Declare Sub AppSleep Lib "kernel32" Alias "Sleep" (ByVal dwMilliseconds As Long)

Public Sub PauseApp(PauseInSeconds As Long)
    Call AppSleep(PauseInSeconds * 1000)
End Sub

Function marketmake(time, triggerstart, triggerstop)
    'Initialize the API
    Dim api As Rotman.RITAPI
    Set api = New Rotman.RITAPI
    'Run the algorithm during certain case time
    
```

*Note: there are better ways to handle this, but this is the easiest way for illustrative purposes.

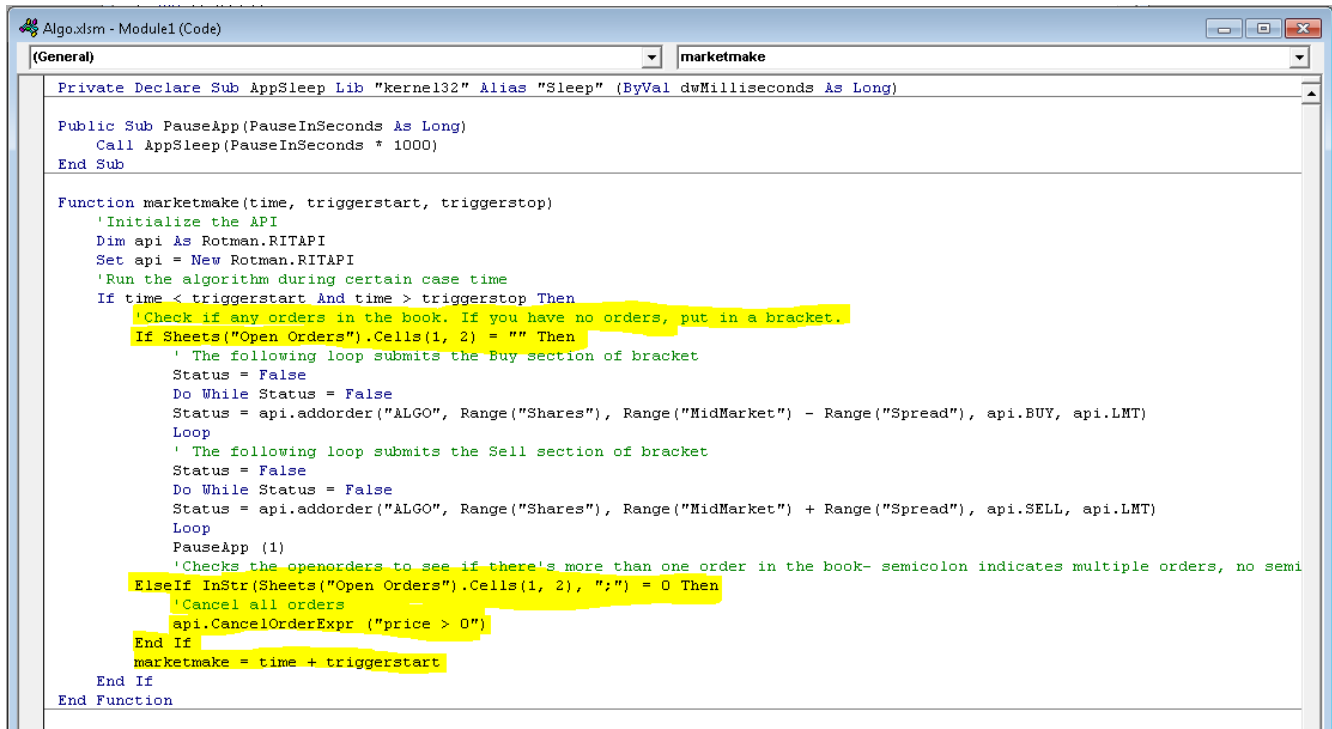
Our code still submits orders constantly, but realistically, we only want it to submit orders when we have no open orders in the book. To fix this issue, we add another sheet to our Excel book labelled "Open Orders" and we add an RTD link to it:



This allows us to track when we have orders in the book. It shows our pending orders are displayed in this box with comma's separating the fields, and semicolons separating each order. [Order1;Order2;Order3]. We can use the semicolon to count the number of orders in the book, if

there's nothing in this box, we know there's no orders, if there's more than 1 semicolon in this box, we know there's more than 2 orders.

Once again we alter our trading code, so that orders are only submitted when no orders are in the book, and we also add code to cancel our existing orders whenever there's only 1 order remaining. Our resulting program submits two orders when we have no orders, it cancels all orders if we only have 1 order, and otherwise it does nothing. This means we always have 2 orders in the book (a bid and an ask).



```
Algo.xlsm - Module1 (Code)
(General) marketmake

Private Declare Sub AppSleep Lib "kernel32" Alias "Sleep" (ByVal dwMilliseconds As Long)

Public Sub PauseApp(PauseInSeconds As Long)
    Call AppSleep(PauseInSeconds * 1000)
End Sub

Function marketmake(time, triggerstart, triggerstop)
    'Initialize the API
    Dim api As Rotman.RITAPI
    Set api = New Rotman.RITAPI
    'Run the algorithm during certain case time
    If time < triggerstart And time > triggerstop Then
        'Check if any orders in the book. If you have no orders, put in a bracket.
        If Sheets("Open Orders").Cells(1, 2) = "" Then
            ' The following loop submits the Buy section of bracket
            Status = False
            Do While Status = False
                Status = api.addorder("ALGO", Range("Shares"), Range("MidMarket") - Range("Spread"), api.BUY, api.LMT)
            Loop
            ' The following loop submits the Sell section of bracket
            Status = False
            Do While Status = False
                Status = api.addorder("ALGO", Range("Shares"), Range("MidMarket") + Range("Spread"), api.SELL, api.LMT)
            Loop
            PauseApp (1)
            'Checks the openorders to see if there's more than one order in the book- semicolon indicates multiple orders, no semi
            ElseIf InStr(Sheets("Open Orders").Cells(1, 2), ";") = 0 Then
                'Cancel all orders
                api.CancelOrderExpr ("price > 0")
            End If
            marketmake = time + triggerstart
        End If
    End Function
```

Our final code is presented above.

In order to run the code, we need to call the function from the spreadsheet. This is accomplished by inputting into one of the cells =marketmake(a,b,c) and referencing the cells to the appropriate fields.

An example is provided below (note: the function can be called from any cell, but the variables must be linked correctly to the time, triggerstart, and triggerstop).

	A	B	C	D	E	F	G	H	I	J
1	Case Data									
2	Broker	[error]	<---=RTD("rit", "NAME")		Market Data					
3	Period	[error]	<---=RTD("rit", "CP")		Stock	Algo	<-- Algo			
4	Cash	[error]	<---=RTD("rit", "CASH")		Position	[error]	<--=RTD("rit", "ALGO POSN")			
5	Buying Power	[error]	<---=RTD("rit", "BP")		Bid	[error]	<--=RTD("rit", "ALGO BID")			
6	Net Liq. Value	[error]	<---=RTD("rit", "NLV")		Ask	[error]	<--=RTD("rit", "ALGO ASK")			
7	Risk Free Rate	[error]	<---=RTD("rit", "RF")		Midmarket	#DIV/0!	<--=ROUND(AVERAGE(F5,F6),2)			
8	Time Remaining	[error]	<---=RTD("rit", "TR")							
9	Time in a Year	[error]	<---=RTD("rit", "TY")							
10	Profit	#VALUE!	<---=B6-500000				0 <---=marketmake(B8,B14,B15)			
11										
12	Parameters									
13	Spread	0.02								
14	TriggerStart	290								
15	TriggerStop	35								
16	Shares	10000								
17										

Your algorithm will now trade whenever the case is running (and the time is between 290 and 35!)