Introduction to the AmiBroker AddToComposite()

By Herman van den Bergen ~ Sunday, 13 October, 2002.

~ ~ ~

For convenience the AddToComposite() will from here on be referred to with “Atc”.

The Atc is an extremely powerful function and I strongly encourage you to. This application will show you the basics as well as introduce you to more sophisticated applications. For further help in the application of the Atc please post your questions to the AmiBroker user list, there are many users there willing to help you.

Boxed text contains text copied from AmiBroker Help, AmiBroker Newsletters, and various web postings. Credit was given where it was readily available but omitted in other cases for the sake of saving time.

This is not a programming tutorial, and the code segments provided may not be optimum for what is being accomplished: they are only examples or ideas to get you started with the AddToComposite.

The trading systems included or referred to should NOT be traded as they are; here again: they are only examples of how one could go about designing a system.

Thanks goes to Tomasz Janeczko, Dale Wingo and Dimitris Tsokakis for proof reading this document, for making helpful suggestions, and for providing additional information to be included.

CONTENTS

Introduction
[Next]

To be a successful trader you have to be able to recognize (or better even: anticipate) what other traders are doing. After all the Market doesn't move by itself: it is trader driven. If everybody is buying the price goes up and if everybody is selling the price goes down.

To know what traders are doing you have to deal with averages; i.e. you need an indicator to show how many people are selling or buying, what is the average price change in a sector, how many shares are traded, etc. Some indices may provide this information but they are rigid; you cannot modify the stock population they are based on. Using the AmiBroker Atc you can create almost any averaging (based on multiple securities) indicator you may need.

You can also use it to sum parameters or values from traditional indicators for individual Stocks or groups of stocks in your Watchlists, or in Sectors, Industries, specific Markets or even All Markets.

A typical application for the Atc would be to "gate" or "enable" your trading signals with a Composite Bull/Bear Trend indicator. In this case you may decide not to place Long trades in a Bear Market and/or not to place Short trades in a Bull's Market. This is sometimes referred to as "qualifying" your trades or "confirming" your trades with secondary signals or indicators.

You can exploit the correlation of individual stocks to composite indicators by designing a Trade-The-Market (TTM) trading system. Dimitris Tsokakis has made numerous posts on this topic on the Yahoo AmiBroker user list. Such systems may completely ignore individual stock price movements and trade a portfolio purely on the basis of signals derived from overall market movements. This idea is controversial however I suggest you explore the idea. Several very interesting TTM systems have been discussed on the AmiBroker mailing list.

For those who enjoy researching the Markets and look for anomalies (great pastime!) the Atc offers a variety of opportunities. This function is a great tool to sum up events, patterns, parameters, etc for export and analysis to/in Excel.

It is important to develop a general appreciation of what the Atc can do for you. The best place to start is the AmiBroker Atc help (Copied below) and the 12/2001 and 04/2000 issue of AmiBroker tips newsletter on the AmiBroker Website.

The 12/2002 newsletter was written before the afl Atc became available. It creates and explains how to create composites using J- and VB-scripts. Program examples are

discussed and listings are provided. While the script versions are more complicated they do provide a different view on the topic and this could be important in some applications.

Calculating multiple-security statistics with the AddToComposite function [Next]

The vast majority of AFL functions operate on single security prices. The are two exceptions from this rule provided by **RelStrength()** and **Foreign()** functions. These two functions allow you to use other security prices in the AFL formula. Although these functions are very useful for things like relative performance charts, they are not so useful for tasks requiring prices of all stocks (or a large number of stocks) because one would need to type several hundreds of **Foreign()** function calls to do so. Moreover this approach would require listing all the ticker names within the formula, which makes the formula tight to particular market. We obviously need completely different approach...

Just imagine if we were able to store the results of calculations performed on single security somewhere and then use those partial data to generate some multiple security indicators. You may say that one can create the exploration, then export the results to the CSV file, then load it into Excel and then perform the calculations there. It would work (in some cases) but you have to agree that the solution is not nice.

This is the area where AddToComposite function can help.

Basically the concept behind AddToComposite is that we run our formula (using Scan feature) through a group of stocks performing some calculations. We will compute some multiple security statistics and store the results in the artificial ticker created using AddToComposite function.

2.3 The solution

The key to the solution is the following algorithm:

1. Do some ordinary AFL calculations using any of available functions
2. Add the result of the calculations to one of the O, H, L, C, V, I fields of our artificial ticker (named for example "~composite")

When the above procedure is repeated over a group of symbols our composite ticker will contain the sum of results of individual symbol calculations.

Step 2 described above is implemented entirely inside AddToComposite function:

SYNTAX **AddToComposite**(array, "ticker", "field", flags = 3)

RETURNS	NOTHING
FUNCTION	Parameters: array - the array of values to be added to "field" in "ticker" composite symbol "ticker" - the ticker of composite symbol. It is advised to use ~comp (tilde at the beginning) newly added composites are assigned to group 253 by default and have "use only local database" feature switched on for proper operation with external sources Possible field codes: "C" - close , "O" - open, "H" - high, "L" - low, "V" - volume, "I" - open interest, "X" - updates all OHLC fields at once. Flags - contains the sum of following values 1 - reset values at the beginning of scan (recommended) 2 - put composite ticker into group 253 and EXCLUDE all other tickers from group 253 (avoids adding composite to composite) 4 - put last scan date/time stamp into FullName field. AddToComposite function also detects the context in which it is run (it works ONLY in scan mode) and does NOT affect composite ticker when run in Indicator or Commentary mode, so it is now allowed to join scan and indicator into single formula.

AddToComposite function opens up a huge variety of interesting applications. The following examples will help you understand what you can do with AddToComposite function.

Example 1:

Let's say we want to create custom index (average of prices of multiple tickers). With AddToComposite function you can do this fairly easy:

```
/* AddToComposite statements are for Automatic Analysis -> Scan */
/* add Close price to our index OHLC fields */
AddToComposite(Close, "~MyIndex", "X" );
/* add one to open intest field (we use this field as a counter) */
AddToComposite( 1, "~MyIndex", "I" );
Buy = 0; // required by scan mode
/* this part is for Indicator Builder */
Graph0 = Foreign( "~MyIndex", "C" )/Foreign( "~MyIndex", I" );
```

You should use above the formula in Automatic Analysis -> Scan mode (over the group of stocks of your choice). This will create "~MyIndex" artificial ticker that will contain

your index.

Shortly this formula just adds Close price to OHLC fields (the "X" field stands for all OHLC) of our artificial ticker ~MyIndex. Additionally we add "1" to "I" (open interest) field - effectivelly counting the number of symbols scanned. We can use symbol count later on to divide the sum of prices by the number of symbols included (the last line of the formula above).

Example 2:

In the second example we will show how to calculate the indicator that shows the number of stocks meeting certain criterion. In this example this would be RSI less than 30 (oversold condition), but it can be anything you like.

So the first line of our formula will be:

```
values = rsi() < 30;
```

This will store "true" in the values array for all date points when RSI is less than 30. Then we add regular AddToComposite part:

```
buy = 0; // do not generate signals
AddToComposite( values, "~MyComposite", "V" );
```

If we run the formula using "Scan" function of Automatic Analysis window the result would be an artificial stock "~MyComposite" filled with quotations. The Volume field of those quotes will contain the number of stocks meeting our criterion (RSI<30) in the population of scanned stocks.

You can easily see the chart of this new "indicator" using the following custom formula in Indicator Builder:

```
graph0 = foreign("~MyComposite", "V");
```

High values of this "indicator" show that most of the stocks in the analysed group are oversold. This usually happens before a great rise of the whole market. We just created market-wide oversold detector!

Example 3:

In the third example I will show you how to use the same technique to count the number of open positions of your trading system. This is useful if you want to know how big account would you need to trade your system following all the trades. Our formula will be very similar to the one before.

First we should have our original trading system formula:

```
/* Your original formula here */
/* In this example this is simple macd/signal crossover system)
buy = cross( macd(), signal() );
sell = cross( signal(), macd() );

/* the following line uses Flip function to get "1" after the buy signal and
reset it back to "0" after sell appears. */
in_trade = flip( buy, sell );
AddToComposite( in_trade, "~OpenPosCount", "V" );
```

We use "~OpenPosCount" artificial ticker to store the results. Again we should run just Scan of the formula and the "~OpenPosCount" ticker would become available.

Use

```
graph0 = foreign( "~OpenPosCount", "V");
```

in Indicator Builder after running the back-test to see the chart of the number of open positions of your system.

2.4 Notes

Please note that to update any composite ticker (for example after adding/editing quotes) you should run "Scan" again.

The idea was originally presented in the 12/2001 issue of AmiBroker tips newsletter. Special thanks to Mr. Dimitris Tsokakis for very constructive discussions that allowed creation and enhancements of this idea.

AmiBroker 4.16 AddToComposite additional flag definitions [Next]

In AFL you can use the “+” or the “|” (preferred) to combine flags. You can simple add the integers or you can spell-out the full name of the Flag. You should not use an assignment (Flag = 1) as an argument, this syntax is only used to indicate that this argument is optional and has a “default” (you can omit the argument). For example:

AddToComposite(array, "ticker", "field", 7)

Is the same as:

AddToComposite(array, "ticker", "field")

.

AddToComposite has 2 new flags that allow running AddToComposite in backtest/optimization mode and exploration mode

```
atcFlagEnableInBacktest = 8
atcFlagEnableInExplore = 16
```

the following constants were added for readability of formulas using AddToComposite:

```
atcFlagResetValues = 1
atcFlagCompositeGroup = 2
atcFlagTimeStamp = 4
atcFlagEnableInBacktest = 8
atcFlagEnableInExplore = 16

atcFlagDefaults = atcFlagResetValues | atcFlagCompositeGroup | atcFlagTimeStamp;
```

Naming Composites [Next]

Composite names should be less than 15 char and start with an tilde (~). In the 12/2001 newsletter names for composites start with an underscore (_) however this can result in count errors.

You should use tilde ~ as a first character of your composites. Using underscore (_) is not recommended. Why? Symbols beginning with underscore are put at the start of the symbol list this causes that during first scan the first symbol is counted twice.
This happens only in the first run. If you don't delete composites physically but simply re-run scans they are updated correctly as the second and further scans do not face the problem of re-indexing. Anyway using tilde at start is better because results are the same no matter if you delete the files or not.

Composite names are limited to 15 chars but the TIME STAMPS are not stored in ticker symbol field but in "Full Name" field. [Tomasz Janeczko]

Deleting Composites [Next]

You can delete composite files from the Workspace, just click-select (highlight) the composite to delete and Right-Click -> Delete. However you can only remove one composite at a time this way and it is a bit slow, execution- as well as user-wise.

Caution is advised since deleted stocks or indices can be a hassle to replace. To remove many composites it is better to click Symbols -> Organize Assignments -> Groups -> Group 253 (or your custom assigned name) and use one of two methods:
1) To delete a sequential block of composite files you can use Shift-Select to select a block of stocks, and then click Delete to delete them all.

2) To delete randomly listed composite files you hold down the CTRL key while click-selecting the composites you want to delete. When all have been selected (Highlighted) you click the Delete button to delete them all.

Sometimes, when we experiment with code we may mess up things and we may end up with composite files that do not show in the composite directory (group 253) but that still remain in the All group and there appears no way to Delete these files. You can correct this problem by Deleting these composite files from the subfolder in your data directory named "_" (underscore), then delete broker.master file in your data directory, and then re-load amibroker. Always make backups before performing making major changes.

While somewhat off-topic I might as well caution you also that you should never remove stocks from one of your Watchlists by Deleting them from your Workspace: this will remove them permanently from your database! Only use Symbol -> Organize Assignments or Symbol -> Erase WatchList(s) to modify your Watchlist contents.

Remember that if your current symbol is a Composite your Atc will not function.

Making the Atc work in all AA Functions [Next]

The previous topic explained the different flags for the Atc and unless you select the proper flags it will not work. You can make the Atc work in several or all AA operations by combining the appropriate flags, Flag=0 for Scan, Flag = 8 for the Backtester and Optimizer and a Flag of 16 for the Explorer:

```
AddToComposite( Data, "~MyComposite", "X", 1 + 2 + 8 + 16 );
```

Note that this code will Reset the Composite at each iteration when Optimizing. More on the Reset later.

Enabling and Disabling the Atc [Next]

You may want to software-control whether data is added to the composite, in other words, the data to be summed must meet certain criteria, which are defined by you. The result of your equation must be a binary, True (non-zero) or False (zero), expression.

As AddToComposite just ADDs to composite it is relatively easy to disable adding

by simply ADDING ZERO:

```
AddToComposite( IIF( Enable, Array, 0 ), "~composite", "C");
```

[Tomasz Janeczko]

Atc Resets [Next]

The Atc has a Reset Flag, if this Flag is set to True (non-zero) the bar-data in a pre-existing composite array will be Reset to Empty (–1e10) at the begin of each new Exploration, Backtest or Scan. An Optimization consists of many Backtest so, unless you take proper precautions, any existing composite file will be Reset at the begin of each new iteration.

Question: If I do not use Flag = 1, does this mean the values won't get reset? If they do not get reset, then what happens the next time I run a scan on AddtoComposite? Are the new values added to the values from the previous scan?

Answer: If you use flag = 0 values won't reset. new values are added to the values from the previous scan. [Tomasz Janeczko]

Where do the Composites go? [Next]

You can click Symbol -> Categories -> Groups -> Select Group 253 -> Edit Name -> Type your name for your Composite Group (I renamed it to “Composites”).

Question: " Somehow I have Composite ticker in both Group 253 and Market 253.
I don't know what the impact is.... any comments / experiences? "

Answer: Yes it is normal. By default all newly created composites are put into group 253 and market 253 for easy identification and to be able to filter them out from further scans (not to include previous composite value in a composite itself) [Tomasz Janeczko]

Which Field to use in your Atc? [Next]

The OHLC fields have certain mathematical relationships that restricts their use and prohibits you from putting random numbers in all fields. Due to inherent error checking of the arrays by AmiBroker the results may be unpredictable if you ignore this. The easiest way is to use the “X” field designator; it fills all four price fields (OHLC) with the same values and no conflicts will arise:

AddToComposite(Data, “~MyComp”,”X”;

You can also use the “X” designator to read back data from composites created this way:

Plot(Foreign(“~MyComp”,”X”),”MyComp”,1,1);

Volume and open interest fields are "independent". Although you can put your composites in any of OHLCVI fields the close field is related to high/low fields, and if you store the values only in close field without changing H/L and try to display candlesticks out of it - the results will be strange. [Tomasz Janeczko]

Copying Composite Files [Next]

You can substitute you own Composite names for the ~MACDBULL and ~MACDBEAR names used in the example below and make copies of your composites as follows.

Question: While we're at it ... is there any way to make the composite 'save' and immediately make it available in *all* of the AmiBroker databases? Answer: As composite is simple data file you can just copy the file. (please make sure that composite ticker symbol exists in all target directories). AmiBroker allows to define "special" symbols that are ALWAYS stored in local database. This setting is available from Symbol -> Information "Use only local database for this symbol". So you have to do it in slightly reversed order: 1) Create symbols ~MACDBULL and ~MACDBEAR in AmiFast directory and mark "Use only local database for this symbol" 2) Create ~MACDBULL and ~MACDBEAR in source database (AddToComposite) 3) Copy the files ~MACDBULL and ~MACDBEAR over to database AmiFast. (AmiFast is a database which has FastTrack data plug-in and Local Storage disabled.) 4) Select ~MACDBULL. This procedure is independent of plugin type. [Tomasz Janeczko]

Cloning a Security [Next]

Sometimes it is handy to clone a security so that you can make changes to it for testing purposes. You cannot make changes to external data base securities but composites are saved in local memory and can be modified and saved with altered values.

```
AddToComposite(O,"~"+ Name(),"O");
AddToComposite(H,"~"+ Name(),"H");
AddToComposite(L,"~"+ Name(),"L");
AddToComposite(C,"~"+ Name(),"C");
AddToComposite(V,"~"+ Name(),"V");
Buy=Sell=Short=Cover=0;
Filter=1;
```

Caution: You should Scan this formula on a single stock or you will clone all the stocks in your group. One application would be to create a dummy stock for testing purposes, its price arrays could be made up from interesting price periods from a variety of different stocks or artificial data.

Dynamic naming of Composites [Next]

You can convert numbers to strings using the WriteVal() function so you can embed all kinds of numbers into you composite name. Let us assume you wanted to save the full Equity array for each iteration of your optimization, you could use this type of formula:

```
ID = Optimize("ID",1,1,5,1);
AddToComposite(Close, "~Composite-"+WriteVal(ID,1.0), "X", 2 + 8 + (ID==1));
```

This Optimization would result in five composite files being added to your workspace. You can also include Security names by using Name() or any other function that returns a string. Be careful when running an optimization using many iterations! You don't want to end up with 1000 composite files!

Note that if the Reset flag (1) were set all the time the composite would Reset at each new iteration, using (ID==1) causes it to Reset only on the first pass when ID=1.

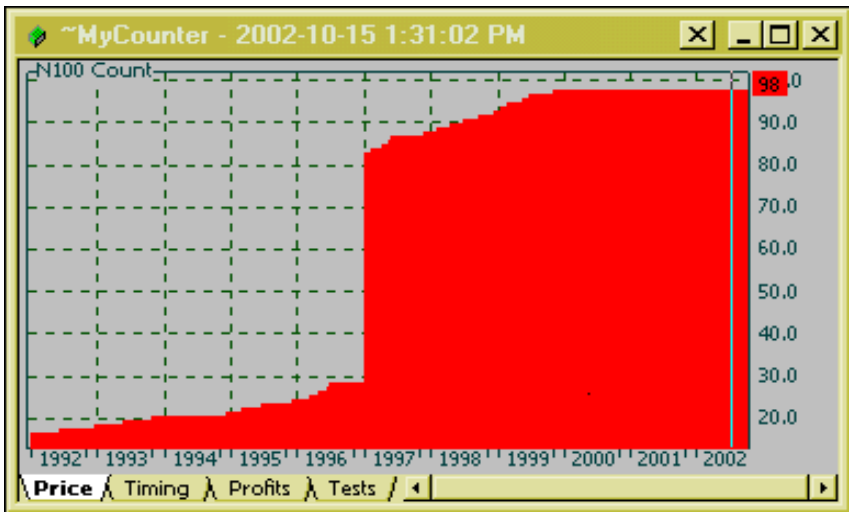
Counting the number of stocks traded each day [Next]

This is a very frequent application for the Atc. Whenever we sum stock parameters and we want to calculate the average for each bars we need to know how many stock were summed. To count stocks you Scan the Market or Watchlist of your choice with this line of code:

```
AddToComposite(1,"~MyCounter","X");
```

```
Plot(Foreign("~MyCounter","C"),"MyCounter",4,2);
Buy=Sell=Short=Cover=0;
Filter=1;
```

If you plot this with you will see a chart depicting the number of stocks traded on each trading day:



Stock Count of my N100 Watchlist, it only contained 98 stocks.

Calculating Advance/Decline lines [Next]

You can count the number of stocks for which the price increased, decreased or did not change, in your portfolio of WatchList by counting the bars for which the daily rate-of-change of price was greater, less than, or equal to zero:

```
AddToComposite(ROC(C,1)>0,"~UpBars","X");
AddToComposite(ROC(C,1)<0,"~DnBars","X");
AddToComposite(ROC(C,1)==0,"~NcBars","X");
```

Using such composites you can create you own variation of the AmiBroker AD-Line (see AFL Function), which is defined as:

```
Difference = ( AdvIssues() - DecIssues() )/ ( UncIssues() + 1 );
DiffSqrt = IIF( Difference > 0, sqrt( Difference ), - sqrt( - Difference ) );
ADLine = Cum( DiffSqrt );
```

When possible you should use built-in composites for A/D lines, see:
http://www.amibroker.com/newsletter/04-2000.html or use
vendor-provided special tickers from advancing/declining issues.

Counting binary Events [Next]

If the Event or Condition is a simple binary value, like a Buy or Sell signal or the number of stops that were executed, you can count them as follows:

```
AddToComposite(Event,”~EventCounter”,”X”);
AddToComposite(Sell==2,”~MxLsStopCount”,”X”);
```

Note:

```
// You can find out on which bars stops were
triggered
// now sell array contains
// 1 - for normal exit
// 2 - for max. loss stop
// 3 - for profit target
// 4 - for trailing stop
```

Of course there is nothing wrong using more complex conditions to count the occurrence of certain patterns (the condition used here does nothing meaningful):

```
Condition = ((C>O) and (ROC(C,1) > 0.5)) or (O > ref(O,-1)*1.07);
AddToComposite(IIF(Condition,1,0),”~CondTrueCnt”,”X”);
```

Qualifying Counts [Next]

You can also qualify Events with secondary conditions, right in the Atc, for example when the volume is above a certain level or Equity increased more than 10% in the last 3 days:

```
AddToComposite(IIf(Buy AND (V > 1000000),1,0),"~MyCounter","X");
AddToComposite(Sell * (ROC(Equity(),3)>10),"~NiceEquity","X");
```

Don’t forget trade delays when you work with Buy and Sell signals. Note that here again I take advantage of the fact that a Boolean expression returns a 0 or 1, so we can multiply our remaining expression with that to simplify code.

You may think you can use the AA Filter setting but the Filter works only in Scan and Explorations.

Coping with Volume overflow [Next]

Question: I am stumped and need help. Hopefully some one can steer me in the right

direction. I have run a composite on a universe of some 1300 stocks with the following code.....(snip snip: code and explanation that the Volume composite behaves erratic).

Answer: Divide volume by 1000 or more to avoid overflow of integer volume field
AddToComposite(V/1000,"~Universe" ,"v");

Volume field is stored as an 32 bit integer number which has the range of - 2^31 to 2^31 –1 [Tomasz Janeczko]

Full Sector, Industry or Market Composites [Next]

With a little experimenting you will be able to adapt the following example to Industries and Markets as well.

Question: I am building a single scan to build 12 indexes (1 each for each of the 12 sectors). Before I type the whole thing, times 12, could I please have input to my method?

Answer: You can also write universal formula:

```
sym = "~" + SectorID(1);
AddToComposite( L, sym,"L");
AddToComposite(  O, sym,"O");
AddToComposite( H, sym,"H");
AddToComposite( C, sym,"C");
AddToComposite( V, sym, "V");
Buy=Sell=Short=Cover=0;
Filter=1;
```

that calculates individual composites for each sector automatically. [Tomasz Janeczko]

Creating WatchList Composites [Next]

The following formulae make it easy to create a composite for each of your WatchLists. Like in some other examples this method takes advantage of the fact that a False Boolean expression equates to Zero and that all values multiplied by zero will be equal to zero. This fact can be used to simplify many formulae that otherwise would require an IIF() statement.

```
Var = Close;
AddToComposite( InWatchList( 0 )  *  Var, "~Watch1Comp", "X" );
```

```
AddToComposite( InWatchList( 1 )  *  Var, "~Watch2Comp", "X" );
AddToComposite( InWatchList( 3 )  *  Var, "~Watch3Comp", "X" );
Buy=Sell=Short=Cover=0;
Filter=1;
[Waz]
```

Creating Equity Composites when Backtesting [Next]

To create a simple composite Equity array representing the sum of all equities in your backtest, you can use:

```
AddToComposite( Equity(),"~SumEq", "X", 11);
```

Creating individual Equity composites for each stock optimized[Next]

To create individual Equity arrays for each stock tested in an Optimization run you can use this code:

```
SN = Optimize("StockNum",1,1,5,1);
TP = Optimize("TestParameter",1,1,2,1);
AddToComposite( Equity(),"~CompEq"+ WriteVal(Status("StockNum"),1.0) + "-
" + WriteVal(TP,1.0), "X", 11);
```

Be careful: you will get one separate composite ticker for each Optimization iteration you run!

Creating an Equity chart with OHLC price fields [Next]

One could develop a trading system to “trade the Equity”, do a search on the AB user list for more info on this. When doing this you may want to use indicators or formulae that require OHLC prices (like the SAR) and create an Equity that has OHLC prices:

```
AddToComposite(E1,"~EqOHLC","C",1+2+8+16);
AddToComposite(E1*H/C,"~EqOHLC","H",1+2+8+16);
AddToComposite(E1*L/C,"~EqOHLC","L",1+2+8+16);
AddToComposite(E1*O/C,"~EqOHLC","O",1+2+8+16);
Plot(Foreign("~EqOHLC","C"),"~EqOHLC",1,64);
```

For actual use you may want to give a little more thought to what exactly are proper Highs and Lows for the Equity curve – I am sure this could be debated at some length.

Calculate an average Market price [Next]

This example averages the Closing price for a group of stocks but you can substitute Open, High or Low, if you like.

```
//Scan #1
AddToComposite(1,"~Count","X");
//Scan #2
n = Foreign("~Count","C");
AddToComposite(Close/n,"~AveClose","X");

//Plotting
AveClose = Foreign("~AveClose","C");
Plot(AveClose,"AveClose",2,1);
Plot(Close,"Close",1,1);
Buy=Sell=Short=Cover=0;
Filter=1;
```

The above example is structured to alert you to a potential problem. When using multiple Atc calls as shown above you may wonder why you cannot put them all one after another in the same program. For it to function properly you must perform separate Scans if you use Foreign() to provide data to be used in your next line of code.

> You can use Foreign in the next line of code but it represents composite updated only up to current ticker. This is by design.
> [Tomasz Janeczko]

A better way of coding this would be:

```
//Scan
AddToComposite(1,"~Count","X");
AddToComposite(Close,"~SumClose","X");
Buy=Sell=Short=Cover=0;
Filter=1;

//Plotting
AveClose = Foreign("~SumClose","C")/Foreign("~Count","C");
Plot(AveClose,"AveClose",2,1);
Filter = 1;
```

Using the Optimizer as a Task Scheduler [Next]

There may be occasions where you are required to perform two or more successive Scans to accomplish a task. We touched on this earlier. Another example would be were you are using the results of an earlier Atc to calculate the data for the next Atc. Using the recently introduced Atc flags you can do many Scans in one operation by using the Optimizer as a Backtester (because you are not optimizing now) and simple Task Scheduler:

```
Task = Optimize("Task",1,1,3,1);
AddToComposite(IIf(Task==1,1,0),"~Count","X", 2+10+(Task==1));
AddToComposite(IIf(Task==2,Close,0),"~SumClose","X", 2+10+(Task==2));
AveClose = Foreign("~SumClose","C")/Foreign("~Count","C");
AddToComposite(IIf(Task==3,AveClose,0), "~AveClose", "X",
2+10+(Task==3));
```

You have to take care of two things: Control what and when data is added and when the composite is reset. You can extend this idea to your trading systems to do away with many different Scans and, if you structure you program correctly, you can now update your composite as part of your backtest. This way you do not need to keep track of whether your composites are up-to-date.

Using the Atc to create Static storage space [Next]

To create adaptive optimizations you will need to store data from one iteration to the next. AmiBroker doesn't have this facility however if you really need it you can use the Atc to simulate static data storage for single integers and/or floating point numbers and arrays on disk, for recall in later iterations.

To save single variables you first need to put your values in an array. You could use the first five bars of an array named Buffer to save variables Data1…Data5:

```
Buffer =
IIF(Cum(1) == 1, Data1,
IIF(Cum(1) == 2, Data2,
IIF(Cum(1) == 3, Data3,
IIF(Cum(1) == 4, Data4,
IIF(Cum(1) == 5, Data5,0))))));
```

The buffer length will be that of the current stock array. You could even make the Atc save/sum single number results for individual optimizations and store those in single bars of the composite price array for review or later use. This would allow you to count unique events over a set of Optimizations, like relative stock performance in terms of equity. I haven't really explored all this but there are lots of new opportunities now we can use the Atc in all AA operations.

To save an array you simply replace Buffer with the array you want to make static (save). Now all that needs to be done is store your Buffer on disk using the Atc function:

```
Enable = 8; // Set to 8 when you want to save data, to 0 to disable the Atc
AddToComposite(iif(Enable,Buffer,0), “~MyStaticBuffer”,”X”, 3+Enable);
```

You should call this function only once or it will reset on each iteration.

To recall the variables you use the Foreign():

```
Buffer = Foreign("~MyStaticBuffer","C");
Data1 = ValueWhen((Cum(1)==1) AND Enable, Buffer);
  …
Data5 = ValueWhen((Cum(1)==5) AND Enable, Buffer);
Plot(Data5,"Data5",4,1);
```

Adding Columns to your Optimizer Reports [Next]

OK, I cannot give you more columns for your report BUT: you can simulate adding columns to your Optimizer Reports by storing custom data in a composite. After the backtest you can plot the composite or show the contents with an Exploration. Until Tomasz adds custom columns to the Reports here is a work-around you can use when you *really* need that extra data.

```
LB = Optimize("LB",8,7,9,1);
StochRsi=EMA((RSI(8)-LLV(RSI(8),LB))/(HHV(RSI(8),LB)-
LLV(RSI(8),LB)),3)*100;
Buy=Cross(17,StochRsi);
Sell=Cross(StochRsi,83);
Short=Cross(StochRsi,83);
Cover=Cross(17,StochRsi);
E = Equity(1);
Plot(E,"E", 1,1);
AddToComposite( Equity(),"~CompEq LB="+ WriteVal(LB,1.0), "X", 11);
AddColumn(Foreign("~CompEq LB=7","X"),"LB=7");
AddColumn(Foreign("~CompEq LB=8","X"),"LB=8");
AddColumn(Foreign("~CompEq LB=9","X"),"LB=9");
Filter=1;
/*
```

For simplicity this example saves the Equity for each value of LB (LookBack) test, but the idea is of course to save a custom parameter that is not so readily available. Note that the Optimization parameter is inserted in the Composite name for easy identification.

You can display the added information with an Exploration, set Range -> n last quotations = 1 to obtain similar column length as obtained with the optimization.

You can use this method to gather any type of optimization data you want, list it with an Exploration and copy/export to Excel for further analysis or 3D charting. Who said AmiBroker Optimizations were limited?

Caution again: you will add as many Composite files to your Workspace as there are iterations in your Optimization.

Using the Atc for Signal-Averaging (Unfinished) [Next]

Signal-averaging is a technique to display the average behavior of a signal around an Event. In our case the most likely events we would like to Signal-Average are trading signals and chart patterns. I will use the Buy signal for the event but you can substitute any simple or complex condition for the event.

You can use the optimizer to step through n Buy signals and average 10 bars before and after the signal. Events (Buys) are numbered and whenever an Event number matches the Optimizer iteration number the data is shifted using the ref() function to add it with the ref offset at the Display location of the array.

This is ***UNFINISHED*** code! For you to complete **J**

```
// Trading system
StochRsi=EMA((RSI(8)-LLV(RSI(8),8))/(HHV(RSI(8),8)-
LLV(RSI(8),8)),3)*100;
Buy=Cross(17,StochRsi);Sell=Cross(StochRsi,83);
Short=Cross(StochRsi,83);Cover=Cross(17,StochRsi);
E = Equity(1);

// Signal Averaging
Event = Ref(Buy,-1);  // Buy Entry price is the Event to analyze
Para  = ROC(Open,1);  // We will Signal-Average the Open price
MxSam = 300;          // We will Sample 300 Entries
Width = 10;           // We will Display 10 bars, 5 before and 5 after the Event
DCtr = LastValue(Cum(1))-Width/2;
SamNum = Optimize("SamNum",1,1,MxSam,1); //We collect MxSam Samples
EvNum = Cum(Event);                      //We count the Events
Hit = (SamNum==EvNum);
HitLoc = ValueWhen(Hit,Cum(1));
OffSet = DCtr - HitLoc;    //Calc offset from Display to the Event
```

```
AddToComposite(Ref(Para,-OffSet), "~MySigAve", "X", 2 + 8 +
(SamNum==1));
Co = IIf(Cum(1)>DCtr,5,4);
SigAve = Foreign("~MySigAve","C");
Plot(IIf(Cum(1)<(DCtr-Width/2),-1e10, SigAve), "MySigAve",Co,1+4);
Plot(IIf(DCtr == Cum(1), SigAve * 1.1,-1e10),"",6,2); //Ctr Marker
GraphXSpace = 5;
/*
```

The easiest way to work on this is using the button sequence Optimize -> Backtest -> Equity. The ultimate application would be to Signal-Average n signals before the current day and analyzing the results and adjust you formula to compensate for any change in timing.

Calculating Multiple Security Indicators [Next]

Many of the standard indicators can be applied to multiple securities. Good examples would be the RSI, CCI and CMO:

```
AddToComposite(1,"~Count","V");
AddToComposite(RSI(),"~SumRSI","X");
AddToComposite(CCI(),"~SumCCI","X");
AddToComposite(CMO(),"~SumCMO","X");
```

After summing you can calculate the Average values by dividing by

```
N = Foreign("~Count","v");
```

When you are summing parameters across a universe of stocks you can only sum normalized or oscillator values that swing over a constant range regardless of the absolute price values. Normally you should not sum absolute values.

Such composites can be very useful in determining the overall market direction. A simple examples is shown below:

```
/*MACD BULL-BEAR by Dimitris Tsokakis*/
ob=Signal()<MACD();
os=Signal()>=MACD();
values5 = os>0;
values6 = ob>0;
AddToComposite(Values5,"~macdbear","V");
AddToComposite(Values6,"~macdbull","V");
Buy=0;
```

Stochastic Divergences, PDI, NDI [Next]

Stochastic Divergences, PDI, NDI. By Dimitris Tsokakis.
Definitions.
1. Positive Stochastic Divergence Issues (PDI) is the population of stocks that presented a positive stochastic divergence, calculated on daily basis.
2. Negative Stochastic Divergence Issues (NDI) is the population of stocks that presented a negative stochastic divergence, calculated on daily basis.

Formula:

```
/*STOCHDIV*/
ST33=StochD(14);
TR1=LLVBars(ST33,4);
TR2=IIf(ST33<30 AND TR1>0 AND Ref(TR1,-1)==0,Ref(ST33,-1),0);
TRC=IIf(TR2>0,C,0);
vs=ValueWhen(tr2, Ref(st33,-1), 1);
dvs=vs-Ref(vs,-1);
vc=ValueWhen(trc, LLV(C,3), 1);
dvc=vc-Ref(vc,-1);
diver=IIf(dvs>0 AND dvc<0,30,0);
DAS=BarsSince(Ref(TR2,-1)>0);
dd=IIf(DAS<20 AND C>=Ref(C,-1),DIVER,0);

HTR1=HHVBars(ST33,4);
HTR2=IIf(ST33>70 AND HTR1>0 AND Ref(HTR1,-1)==0,Ref(ST33,-1),0);
HTRC=IIf(HTR2>0,C,0);
Hvs=ValueWhen(Htr2, Ref(st33,-1), 1);
Hdvs=Hvs-Ref(Hvs,-1);
Hvc=ValueWhen(Htrc, HHV(H,3), 1);
Hdvc=Hvc-Ref(Hvc,-1);
Hdiver=IIf(Hdvs<0 AND Hdvc>0,90,0);
HDAS=BarsSince(Ref(HTR2,-1)>0);
hdd=IIf(HDAS<20 AND C<Ref(C,-1),HDIVER,0);
values3 = dd>0;
values4=hdd>0;
AddToComposite(Values3,"~posstochdiv","V");
AddToComposite(Values4,"~negstochdiv","V");
Buy=0;

/*To graph PDI and NDI, paste AFL code in Indicator Builder*/
graph0= foreign("~negstochdiv","V");
graph0color=5;
graph1 = foreign("~posstochdiv","V");
```

```
graph1color=6;
TITLE=writeval(LASTVALUE( foreign("~posstochdiv","V")))+
" ISSUES WITH BULLISH STOCHASTIC DIVERGENCE "+
" AND "+writeval(LASTVALUE( foreign("~negstochdiv","V")))+
" ISSUES WITH BEARISH STOCHASTIC DIVERGENCE ";
```

36 Trading Systems Based on the RSIt [Next]

This system is rather long and you should pick up the code by following the URL.

36 TRADING SYSTEMS BASED ON THE RSIt By Dimitris Tsokakis.
Mean indicators may be considered as an average of simple indicators, calculated over the whole market. Every stock has an MACD(). If you add the MACD() s and you divide by the population of the market, the result is the MeanMACD.
MeanMACD=(MACD1+MACD2+...+MACDn)/n
If the market is not directional, some stocks will be up, some flats and some others will be down. For this case, the MeanMACD will be almost flat with small fluctuations. If the Market is directional enough, the majority will trend to the same direction for a period of time. For this case, the MeanMACD will be also directional and will present trendy characteristics, overbought and oversold properties etc.

Markets around the world are directional in the short and medium term. Hence, a well weighted Mean Indicator will help to study the Market behavior and give profitable trading systems. Tomasz Janeczko wrote at the end of http://www.amibroker.com/newsletter/12-2001.html "I think that AmiBroker will help us develop a new brand of technical analysis tools making market entry and exit signals much more reliable than before. "Let us try to investigate the design of trading systems based on Breath Indicators. I selected the Mean RSIt indicator, placed at http://www.amibroker.com/library/detail.php?id=159 applied over the doubly smoothed MACD()

Over-Sold and Over-Bought Indicator [Next]

```
/*STOCHD OSI OBI Composites by Dimitris Tsokakis */
s1=StochD(14);
os=s1<=30;
ob=s1>70;
values11 = os>0;
values12= ob>0;
AddToComposite(Values11,"~stochdosi","V");
AddToComposite(Values12,"~stochdobi","V");
Buy=0;
```

```
//Plotting
/*Graph for ~stochdosi and ~stochdobi*/
Graph0 = Foreign("~stochdosi","V");
Graph1=Foreign("~stochdobi","V");
Title=Date()+" : "+"stochd OSI = "+
WriteVal(LastValue( Foreign("~stochdosi","V")))
+", stochd OBI= "+WriteVal(LastValue( Foreign("~stochdobi","V")));
Graph0BarColor=4;
Graph1BarColor=5;
```

Bull-Bear-Congestive Composites [Next]

```
/*BULL-BEAR-CONGESTIVE*/
/* Market direction is described through 3 composite tickers, based on Relative slope
values. You may read on daily basis the % distribution of bullish, congestive and bearish
phase of the Market. By Dimitris Tsokakis. */
K=EMA((H+L+C)/3,10);
S1=2*(K-Ref(K,-1))/(K+Ref(K,-1));
RS=100*EMA(S1,3);
values16 = abs(rs)<=1;
values17=rs>1;
values18=rs<-1;
AddToComposite(Values16,"~rscongestion","V");
AddToComposite(Values17,"~rsbullish","V");
AddToComposite(Values18,"~rsbearish","V");
Buy=0;

/*Market Direction Graph*/
TOT=Foreign("~rsbearish","v")+Foreign("~rsbullish","v")+Foreign("~rscongestion","v");
Graph1=100*Foreign("~rsbullish","v")/TOT;
Graph1Color=5;
Graph0=100*Foreign("~rscongestion","v")/TOT;
Graph0BarColor=7;
Graph2=100*Foreign("~rsbearish","v")/TOT;
Graph2Style=1;
Graph2BarColor=4;
Title="BULLISH = "+WriteVal(Graph1,FORMAT= 1.1)+"% , CONGESTIVE = "
+WriteVal(Graph0,FORMAT=1.1)+"% , BEARISH = "+
WriteVal(Graph2,FORMAT=1.1)+" %";
```

You may increase the sensitivity of the Market Direction indicator replacing the

```
values16 = abs(rs)<=1;
values17=rs>1;
values18=rs<-1;
        with
values16 = abs(rs)<=0.75;
values17=rs>0.75;
values18=rs<-0.75;
        or even
values16 = abs(rs)<=0.5;
values17=rs>0.5;
values18=rs<-0.5;
        for more expressive results.
```

Composite Bull Pressure and Bear Pressure [Next]

This is basically a trend indicator derived from Bullish and Bearish indicator composites. If this technique interests you should search the AmiBroker user list and net for more info.

```
/*****************************************************/
//Composite Bull Pressure and Bear Pressure
//Version 1.0.. By Anthony Faragasso
//Credit to Peter Gialames for formula
//conversion into plugin Dll.

x=afBBTicker();//This Calls the plugin

AddToComposite(BullishPressure,"~NDX100bullishpressure","x");
AddToComposite(BearPressure,"~NDX100bearPressure","x");

Buy=0;

/***For Indicator Builder and Graphing****/

bull=Foreign("~NDX100bullishPressure","x");
bear=Foreign("~NDX100bearPressure","x");

Plot(hbT3A(100*bear/(bull+bear),10,0.5),"",4,1+4);
Plot(hbT3A(100*bull/(bull+bear),10,0.5),"",5,1+4);
//Plot(MA(100*bear/(bull+bear),13),"",5,1);
//Plot(MA(100*bull/(bull+bear),13),"",4,1);

Title="MARKET PRESSURES // "+""+"BULL_PRESSURE ="+"(
"+WriteVal(Graph1,1.1)+"% )"+" , "+" "+"BEAR_PRESSURE ="+"(
```

```
"+WriteVal(Graph0,1.1)+"% )";

/******End of Composite indicator******/
```

Creating Signal Consensus Composites [Next]

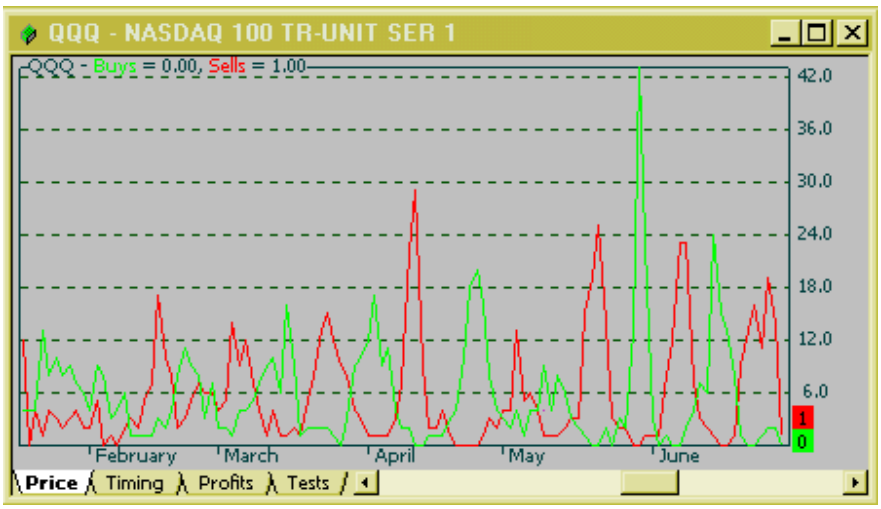
You may have developed a number of different trading system for which you would like to compare signals, in other words you want to know if there is a signal consensus. You could collect many trading systems into one program and create a consensus line or, to do better than that other expensive program out there, create a *consensus indicator*.

Here are a few possible types:

1) The Single Trading System Multiple Security Consensus composite runs a single trading system in the Backtester on a group of stocks and tallies the number of Buy and Sell signals:

```
AddToComposite(Buy,"~Buys","X");
AddToComposite(Sell,"~Sells","X");
Buys = Foreign("~Buys","C");
Sells = Foreign("~Sells","C");
Plot(Buys,"Buys",5,1);
Plot(Sells,"Sells",4,1);
/*
```

If your system gives you generally good signals you can actually “see the signal coming”. I am sure you can create some interesting indicators using the basic Buys and Sells variables. The basic plot would look something like this:



This works best if you backtest a large number of stocks, for example the N100. The result is an indicator that ranges from 0-100. It produces a score of 100 if the system produces an identical signal on exactly the same bar for all stocks. We might assume that this kind of confirmation is a good sign (also assuming your trading system is reasonably good). You might be able to turn this into a TTM system…

2) Backtesting several different trading systems on a single stock or a group of stocks like the N100. Each system would provide its own timing, perhaps derived from completely different principles. Here too you would be looking for confirmation.

3) Backtesting several different Trade-The-Market systems and looking for a consensus between the systems. Same rational as before.

4) Backtesting permutations of the same trading system and creating a consensus signal indicator. For example you could vary the periods by some rule and investigate what happens to the distribution of trading signals.

CCI50 TTM System [Next]

An example of a simple TTM system. It is a nice example of how you can use standard indicators in a TTM system.

```
//CCI50 TTM System by Dimitris Tsokakis
AddToComposite(CCI(40)>0,"~cci40","v");
AddToComposite(CCI(50)>0,"~cci50","v");
AddToComposite(MA(CCI(40),10)>0,"~cci40ma","v");
AddToComposite(MA(CCI(50),10)>0,"~cci50ma","v");

C50=Foreign("~cci50","v");
D=DateNum()>1000401;
L1=Optimize("L1",5,3,10,1);
L2=Optimize("L2",85,80,90,1);
Buy=D*Cross(C50,L1);
Sell=D*Cross(C50,L2);
Short=Sell;
Cover=Buy;
```

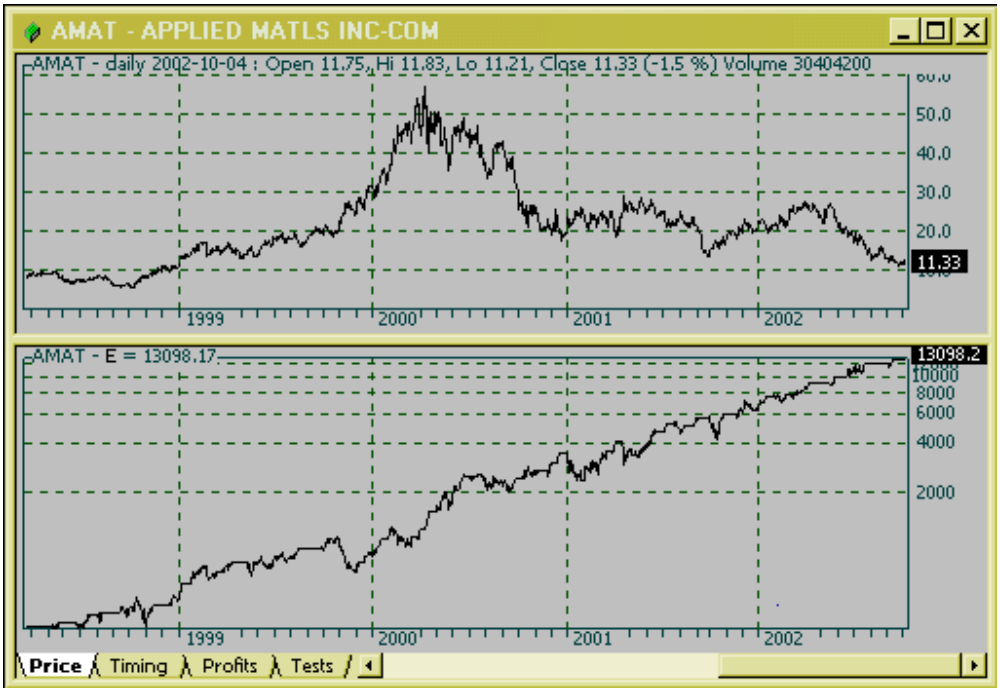
StoRSI-Bull/Bear Pressure trading system [Next]

Earlier I mentioned that you can use composite derived indicators to gate trading signals. Here is a system I posted on the AmiBroker user list (2002-08-12) where I combined Steve Karnish's StoRSI oscillator system with the BullBearPressure indicator origionated by Anthony Faragasso and implemented into a DLL by Peter Gialames. You can substitute a variety of other indicators/systems and get pretty good results:

```
// Composite Bull/Bear Pressure DLLVersion 1.0.. By Anthony Faragasso
// Credit to Peter Gialames for formula conversion into plugin Dll.
// StoRSI by Steve Karnish

x=afBBTicker();//This Calls the plugin
AddToComposite(BullishPressure,"~NDX100bullishpressure","x");
AddToComposite(BearPressure,"~NDX100bearPressure","x");
bull=Foreign("~NDX100bullishPressure","C");
bear=Foreign("~NDX100bearPressure","C");
Pd = 14; //Optimize("PD",10,2,30,2);
SmBULL = hbT3A(100*bull/(bull+bear),Pd,0.5);
SmBEAR = hbT3A(100*bear/(bull+bear),Pd,0.5);
EnL = SmBull > 50;
EnS = SmBear > 50;
StochRsi=EMA((RSI(8)-LLV(RSI(8),8))/(HHV(RSI(8),8)-
LLV(RSI(8),8)),3)*100;
Buy=Cross(17,StochRsi) AND EnL;
Sell=Cross(StochRsi,83);
Short=Cross(StochRsi,83) AND EnS;
Cover=Cross(17,StochRsi);

E = Equity(1);
Plot(E,"E", 1,1);
Plot(EnL*10,"EnL",5,1);
Plot(EnS*10,"EnL",4,1);
/*
```



The StoRSI-BullBearPressure system in action on AMAT.

Optimizing Trading Systems for Expectancy [To Index]

The final topic is just to show that with a little effort you can do just about anything with AmiBroker you like. Applying some of the techniques above I will show you how to optimize your trading system for Expectancy or for any other parameter you like. There is some controversy about which is the exact formula to use; I'll be using one that was posted on the AmiBroker user list by William Wong on Fri 2002-10-18.

A system performance is governed by a combination of %winning trades and pay off ratio (avg win$ / avg loss$). If an expectation of a system is negative, no matter how good the money management, it only slows down an eventual death.

Expectation = (1 + pay off ratio) * %win - 1

e.g. If a system gives pay off ratio of 2:1 but only profitable 30% of the time,

Exp = (1 + 2) * 0.3 - 1 = -0.1, not worth using

But if the accuracy is improved to 40%,

Exp = (1 + 2) * 0.4 - 1 = 0.2, worth using

The higher the expectation, the better the system. Rule of thumb is to improve to at least 0.7.
[William Wong - For the complete post do a search on Yahoo]

Since you have come this far I assume I can dispense explaining all the details of my programming effort. Just be aware this is rough code and for sure you can cleaned-it-up and fancy-it-up to your hearts content. I used the same trading system I introduced earlier.

```
// Composite Bull/Bear Pressure DLLVersion 1.0.. By Anthony Faragasso
// Credit to Peter Gialames for formula conversion into plugin Dll.
// StoRSI by Steve Karnish
x=afBBTicker();//This Calls the plugin
AddToComposite(BullishPressure,"~NDX100bullishpressure","x");
AddToComposite(BearPressure,"~NDX100bearPressure","x");
bull=Foreign("~NDX100bullishPressure","C");
bear=Foreign("~NDX100bearPressure","C");
Pd = 14; //Optimize("PD",10,2,30,2);
SmBULL = hbT3A(100*bull/(bull+bear),Pd,0.5);
SmBEAR = hbT3A(100*bear/(bull+bear),Pd,0.5);
EnL = SmBull > 50;
EnS = SmBear > 50;
Pd1=5;Pd2=12;LB1=8;LB2=12;
Pd = Optimize("Pd",8,Pd1,Pd2,1);
LB = Optimize("LB",8,LB1,LB2,1);
OptStart = (Pd == 5) AND (LB == 8);
StochRsi=EMA((RSI(Pd)-LLV(RSI(Pd),LB))/(HHV(RSI(Pd),LB)-
LLV(RSI(Pd),LB)),3)*100;
Buy=Cross(17,StochRsi) AND EnL;
Sell=Cross(StochRsi,83);
Short=Cross(StochRsi,83) AND EnS;
Cover=Cross(17,StochRsi);
E = Equity(1);

// Expectation analysis
dBuy   = Ref(  Buy,-1);
dSell  = Ref( Sell,-1);
dShort = Ref(Short,-1);
dCover = Ref(Cover,-1);

LongProfit = IIf(dSell,E - ValueWhen(dBuy,E),0);
ShortProfit = IIf(dCover,E-ValueWhen(dShort,E),0);
```

```
Winnings = Cum(IIf(LongProfit>0,LongProfit,0) +
IIf(ShortProfit>0,ShortProfit,0));
Losses   = Cum(IIf(LongProfit<0,LongProfit,0) +
IIf(ShortProfit<0,ShortProfit,0));
NumWinTrades = Cum(((LongProfit>0) OR (ShortProfit>0)) AND
Status("BarInRange"));
NumLosTrades = Cum(((LongProfit<0) OR (ShortProfit<0)) AND
Status("BarInRange"));
TotalTrades   = Cum(dSell OR dCover);
WinningTrades = Cum((LongProfit > 0) OR (ShortProfit >0));
LosingTrades  = Cum((shortProfit < 0) OR (LongProfit < 0));
AveWinTrade   = Winnings / WinningTrades;
AveLosTrade   = Losses / LosingTrades;
PercentWinners = WinningTrades / TotalTrades;
Expectation = ( 1 + AveWinTrade/abs(AveLosTrade)) * PercentWinners - 1;
Ex = LastValue(expectation);
IC = Pd-Pd1+LB-LB1+(Pd2-Pd1)*(LB-LB1);
IC = LastValue(Cum(1)) - IC;

AddToComposite(IIf(Cum(1)== IC,Ex,0),"~Expectation","C",10+OptStart);
AddToComposite(IIf(Cum(1)== IC,Pd,0),"~Expectation","V",10+OptStart);
AddToComposite(IIf(Cum(1)== IC,LB,0),"~Expectation","I",10+OptStart);
AddToComposite(IIf(Cum(1)== IC,LastValue(E),0),"~Exp-
Equity","X",10+OptStart);

AddColumn(AveWinTrade,"AveWTr",1.2);
AddColumn(AveLosTrade,"AveLTr",1.2);
AddColumn(PercentWinners*100,"%Winners",1.4);
AddColumn(Expectation, "Expectation",1.3);

GraphXSpace = 5;
Filter = Status("LastBarInRange");
Plot(E,"Equity",1,1);
/*
```

I use two indicators to display the results:

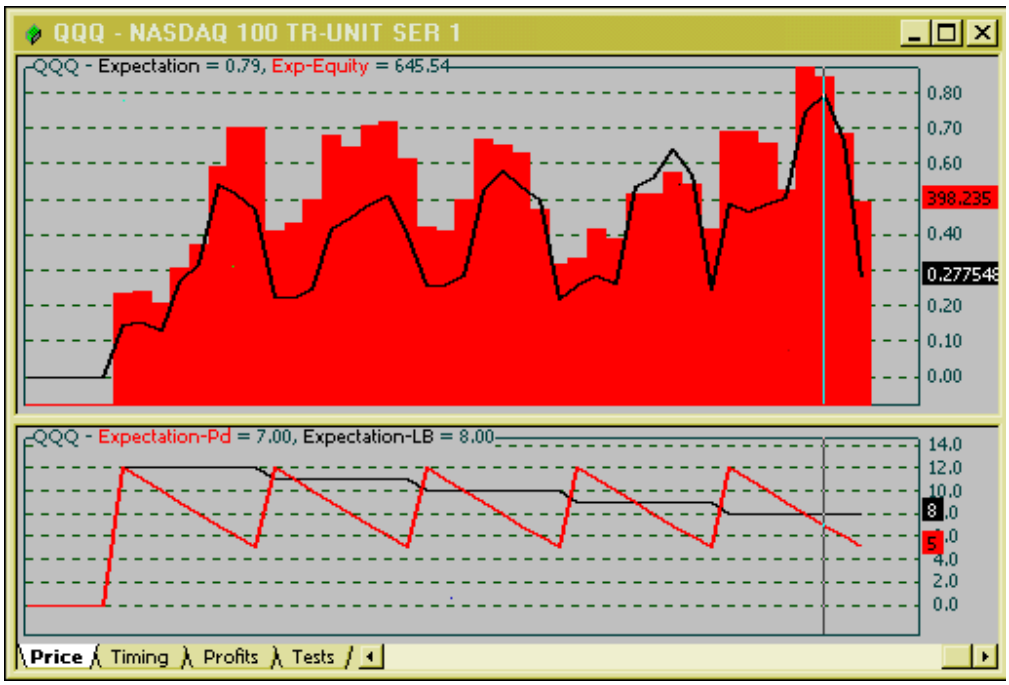
Displaying the Expectancy and Equity curves:

```
Plot(Foreign("~Expectation","C"),"Expectation",1,1+4);
Plot(Foreign("~Exp-Equity","X"),"Exp-Equity",4,16384 +65536 );
GraphXSpace = 10;
```

Displaying the parameters being optimized**:**

```
Plot(Foreign("~Expectation","V"),"Expectation-Pd",4,1+4);
Plot(Foreign("~Expectation","I"),"Expectation-LB",6,1+4);
GraphXSpace = 10;
```

To test for Expectancy you copy the above code to your AA and Indicator formula windows and run the optimizer. Use any range and stock you like, I suggest you use at least several years of data. The more trades you have the better the results. The chart is displayed at the extreme right of the X-axes: be sure to zoom in until you see it. Below is a typical chart.



You can read the Optimization parameters in the lower pane

The upper pane displays the Equity (Red, independent scale) and the Expectancy for the trading system. I am no going to interpret this chart, I only learned about Expectancy a few days ago so please do not email me on this topic. But everybody will be able to see the correlation between Equity and Expectancy at a glance.

The saw tooth waveform in the lower pane shows the corresponding optimization values as they are processed by the Optimizer. It allows you to read of the values for the Optimizing parameters at any given point. In this example I placed a vertical line at the highest expectancy and you can read the values in the lower pane as Pd=7 and LB=8.

A few final references

There are many more good applications for the Atc I would have liked to include but this document is getting too long.

A very good example that you should study is “The Magic OUT3 [and the STOCHRSI]” posted on the AmiBroker user list by Dimitris Tsokakis. It explains how you can use the Atc to design Filters that can improve almost any trading system. See posts #20284, 20294, 20297 etc. He explains how to design a Filter that will improve the performance of almost any trading system.

~ ~ ~

Happy Trading,
Herman van den Bergen
psytek@rogers.com