
VOLUME 11

Information in this document is subject to change without notice.

THE TRADING STRATEGIES IN THIS BOOK ARE EXAMPLES ONLY, AND HAVE BEEN INCLUDED SOLELY FOR EDUCATIONAL PURPOSES. OMEGA RESEARCH DOES NOT RECOMMEND THAT YOU USE ANY SUCH TRADING STRATEGY, AS THE USE OF ANY SUCH TRADING STRATEGY DOES NOT GUARANTEE THAT YOU WILL MAKE PROFITS, INCREASE PROFITS, OR MINIMIZE LOSSES. THE SOLE INTENDED USES OF THE TRADING STRATEGIES INCLUDED IN THIS BOOK ARE TO DEMONSTRATE THE WAYS IN WHICH EASYLANGUAGE CAN BE USED TO DESIGN PERSONAL TRADING STRATEGIES AND TO SHOW SOME EXAMPLES OF HOW CERTAIN POPULAR, WELL-KNOWN TRADING STRATEGY MAY BE INCORPORATED INTO PERSONAL TRADING STRATEGIES. OMEGA RESEARCH, INC. IS NOT ENGAGED IN RENDERING ANY INVESTMENT OR OTHER PROFESSIONAL ADVICE. IF INVESTMENT OR OTHER PROFESSIONAL ADVICE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL SHOULD BE SOUGHT.

# Contents

INTRODUCTION

# Welcome to Volume 11

**Welcome to Volume 11 of the Omega Research Strategy Testing & Development Club. In this issue, you'll find eight new strategies and two supporting articles.**

We believe the new strategies will challenge you and serve as springboards to even better strategies that you'll design yourself. Our High Five strategy uses five moving averages to find high-probability setups and a price move in the direction of the setup to activate a trigger. Hit the Bull's Eye includes a price-channel breakout and an exponential moving average to determine the trend and a price pattern to locate logical entry points for your trades. JK_Night Train is an elegantly simple short-term strategy by Joe Krutsinger, proprietor of the System Academy.

Our MISER strategy is an adaptation of an intermarket strategy by Murray Ruggiero. The strategy's name stands for Murray's Intermarket Strategy Revisited-MISER for short. We added five stops to Murray's strategy and optimized the strategy's moving averages for our STAD Club version. We haven't published many strategies in STAD Club based on classical chart patterns, but we have an interesting one for you in this volume: the Symmetrical Triangle Strategy. Our TrendScore strategy is based on an idea by Tushar Chande. For STAD Club, we added three stops and optimized two of TrendScore's original components.

In Triple Play, we tried to improve Charles Lindsay's venerable Trident strategy. Our main addition was Bollinger Bands, which enable us to mechanically identify Trident's three chart points. Our final strategy in this volume is Win One for the Gapper. Gapper takes an indicator that measures a market's "trendiness," adds a chart pattern created by Larry Williams, and selects four stops from StrategyBuilder to create a very promising strategy.

In addition to the eight strategies described above, STAD 11 features two articles we hope you'll find helpful. The first article, provided by Chuck LeBeau and Terence Tan, introduces two of their favorite exit techniques. The second article explores the new TradeStation 2000i Strategy Performance Report, offering suggestions on how to use many of the new performance statistics.

We had an interesting and enjoyable time putting this STAD Club volume together for you. Hope you like it as much as we do. Thanks!

# Contents at a Glance



# Additional Educational Services

Omega Research is committed to enhancing individual trading potential through quality education. To learn more about strategy trading, an Omega Research product, or EasyLanguage, visit our web site at www.omegaresearch.com or call **(800) 439-7995** (outside US 305-485-7000) and ask about the following educational services:

## Workshops

Omega Research offers a variety of workshops on Omega Research 2000i products. Workshops are an excellent way to learn about the products, strategy trading, and EasyLanguage. Spend a day with a Product Training Specialist and exchange ideas with other users like yourself. All workshops provide a 100% satisfaction guarantee. Call now for more information or to register — space is limited!

## EasyLanguage Resource Center

One of the best ways to learn is by example, and the EasyLanguage Resource Center on our web site is an excellent source of examples. In this Resource Center, we list all the analysis techniques — indicators and trading strategies — published in the *Technical Analysis of Stocks and Commodities* magazine, as well as popular analysis techniques worth taking a look at. Access to this Resource Center is free of charge. Feel free to download and review any of the analysis techniques and their descriptions. Our web site address is **www.omegaresearch.com**.

# Getting Started

To begin reviewing your strategies, transfer the analysis techniques into your TradeStation® library and then apply the strategy you want to review to a chart. Use the Strategy Performance Report to view the strategy's results and take a look at the EasyLanguage instructions by opening the strategy in the PowerEditor™.

To transfer the analysis techniques into TradeStation:

1. Place the Strategy Testing and Development Club CD in the CD-ROM drive.
2. Start the PowerEditor. In **Windows**, click **Start,** choose **Programs,** choose **Omega Research (OMGA)** and choose **EasyLanguage PowerEditor.**
3. In the PowerEditor, use the **File - Import and Export** menu sequence**.**
4. Select the **Import EasyLanguage Archive File (ELA and ELS)** option and click **NEXT.**
5. Click **Scan.**
6. In the **Enter drive letter to scan** edit box, enter the drive letter for your CD-ROM drive (normally D), and click **OK.**
7. Choose **STAD11.ELS** from the list and click **NEXT.**
8. Below the **Analysis Types** box choose the **Select All** button and click **NEXT.**
9. Below the **Available Analysis Techniques** box choose the **Select All** button and click **FINISH.**
10. Once the files are transferred and verified, a dialog box appears informing you that the transfer was performed successfully. Click **OK.**

For your convenience, the names of the strategies in this volume all begin with STAD11 (although the signals will not have this prefix). You can now open the strategies in the PowerEditor and view the EasyLanguage instructions and/or apply them to a chart in TradeStation. You can remove your CD from the CD-ROM drive and store it in a safe place. As you apply the strategies and work with them, refer to this book for detailed explanations of the strategies and the EasyLanguage used to create them. For instructions on applying strategies and viewing the Strategy Performance Report, please refer to your *TradeStation User's Manual*.

***Note to SuperCharts® Users:*** *To transfer the strategies into SuperCharts, use the Tools - QuickEditor menu sequence and select Transfer. Keep in mind, however, that although you can apply the strategies in SuperCharts, you will not be able to view the EasyLanguage instructions in the QuickEditor. This is because the strategies were designed in the EasyLanguage PowerEditor. Also, if you are using SuperCharts End of Day, some of the strategies will not apply as they are designed for intraday trading. Since the purpose of the STAD Club is to provide you with a learning tool, and viewing the EasyLanguage instructions is an essential part of this learning process, the use of this club for SuperCharts users is limited.*

***Note to TradeStation or SuperCharts 3.x Users:*** *The strategies for the Club were designed using TradeStation 2000i. As such, some of the features used, such as automatic drawing of trendlines and/or text, are not available in previous versions of TradeStation (or SuperCharts). An effort is made to provide a variety of strategies that incorporate both long standing and new features; however, keep in mind that as new features are developed, we will naturally want to showcase and educate users on these features; therefore, users of the most recent version of our software will be able to make the most use of the Club.*

# Obtaining Technical Support

Depending on your question, there are two resources at your disposal: the EasyLanguage Support Department and the STAD Club E-Mail Address.

## EasyLanguage Support Department

The EasyLanguage Support Department provides EasyLanguage support via e-mail or fax and is designed to help you troubleshoot an analysis technique or trading strategy you are currently working on. For example, if you are incorporating a trading strategy from the Club into your own and have a question about the implementation, the EasyLanguage Support Department can answer it.

Please keep in mind that while this department can answer any EasyLanguage question, it cannot answer questions about the STAD Club specifically, such as the theory behind a strategy in the STAD Club, why a strategy was developed a certain way, or why the strategy is not performing as you expect it to, etc.

Fax Number: **(305) 485-7598**

E-Mail Address: **easylang@omegaresearch.com**

Be sure to include the following information in your fax or e-mail:

- Name
- Security Block or Customer ID Number
- Telephone Number
- Fax Number
- Product you own
- EasyLanguage instructions you are working on
- Detailed description of your problem

Please allow 48 hours for a response.

## STAD Club E-Mail Address

Another resource at your disposal is the STAD Club e-mail address.

Please realize that when you send a message to this e-mail address, you will not receive a response directly; your message will be reviewed and the answer incorporated into the next volume of the STAD Club, when applicable. Therefore, if you need technical support on EasyLanguage, please use the above fax number or e-mail address.

**stadclub@omegaresearch.com**

Please send any comment, suggestion, or question regarding the strategies in the Club to the STAD Club e-mail address, and in each subsequent volume we will publish the most common suggestions and questions.

# Benefits of Strategy Trading

There are at least five major benefits of trading in a strategic manner as opposed to trading in a discretionary manner:

**1.** You'll have a strategy that is compatible with your own personality and trading style — a strategy that you are comfortable with and that you can follow.

**2.** You will eliminate overly emotional trading and reduce the stress of constantly making subjective, spur-of-the-moment trading decisions.

**3.** You will have objective entry and exit criteria that have been validated by historical testing of quantifiable data.

**4.** You will know the maximum peak-to-valley drawdown that your strategy has experienced in the past, and you can make sure that you are adequately capitalized (both financially and psychologically) to withstand another worst-case drawdown.

**5.** You will gain confidence in both your strategy and yourself, thus strengthening your ability to follow your strategy and to trade in a highly disciplined manner.

As you continue to become more proficient as a strategy trader, you will almost certainly discover even more benefits of a strategic approach.

# Getting Ideas For Strategies

We can easily think of at least five great ways to get ideas for trading strategies. You'll probably come up with at least a few more. Here's our quick list:

**1.** SuperCharts and TradeStation's built-in indicators, ShowMe™ studies, PaintBars™, and strategies

**2.** *Trading As A Business* by Charlie Wright (available from Omega Research)

**3.** Jack Schwager's Complete Guide to Designing and Testing Trading Systems (12 videos, CD, manual; available from Omega Research)

**4.** OmegaWorld (June, 2000, New York City)

**5.** And, of course, Omega Strategy Testing & Development Club (ten new trading ideas with manual and CD, published six times per year. Club members also receive a password for Omega's STAD Club online forum.)

Once we're convinced that strategy trading is more likely to generate consistent profits than discretionary trading, and once we have an idea for a trading strategy, how do we progress from an idea to a complete strategy?

**Building a Trading Strategy**

We hope the following ten-step plan will prove useful:

**1.** Write your trading idea as a ShowMe study. Scroll through several years of data to develop a sense of how your idea performs.

**2.** Write a very simple strategy based on your idea. For example, you could write a strategy that enters a position based on your idea and exits the position automatically after n-days. Alternatively, you could write a stop-and-reverse strategy that uses your idea to enter, exit, and reverse positions.

**3.** Design a setup for your strategy. A setup alerts you that a trading opportunity has developed. Setups don't get you into a trade, but they do tell you that market conditions have become favorable for a trade. An example of a buy setup is a market posting two consecutive closes above a moving average. An example of a sell setup is the Relative Strength Index (RSI) crossing from above 70 to below 70.

**4.** Design an entry for your strategy. An entry is the criterion that must be met after a setup for a trade to be initiated. An example of a buy entry is a market rallying one average daily range above yesterday's close. An example of a sell entry is a market's decline below the previous week's low.

**5.** Design an exit for your strategy. An exit is the criterion by which a trade is closed out. Trailing stops, profit targets, and exit conditions will account for most of your strategy's exits.

A trailing stop is set below the current price for a long position and above the current price for a short position. When you are in a long position, you raise the trailing stop as the market trades higher to lock in profits; while short, you lower the trailing stop as the market trades lower, locking in profits.

An alternative to exiting on a trailing stop is exiting at a profit target. A profit target closes out a trade when the price reaches a specified objective. One example of a profit-target exit is to close out a position on the second close above the high of the entry day. Another example is to automatically close out a trade when open profits equal three times the initial risk on the trade.

An exit condition gets you out of a trade when a market no longer justifies an open position. Good traders do not always rely on stops to exit their trades. If the technical condition that got you into a trade (e.g. a rising moving average) is no longer in effect, you should exit the trade immediately rather than waiting for your stop to be hit.

**6.** Select the data on which you will test your strategy. For example, you might choose to test your strategy on continuous, back-adjusted data on U.S. Treasury Bonds from January, 1978 through December, 1997.

**7.** Divide the test data into five equal parts. Since you are going to test your strategy on 20 years of data, each part consists of four years. The first four years (01/02/78 - 12/31/81) are reserved for the backward test, and the last four years (01/02/94 - 12/31/97) are reserved for the forward test. The middle 12 years (01/02/82 - 12/31/93) are the data on which you will test and optimize your strategy.

**8.** Test and optimize your strategy on the large, middle section of data. To evaluate the results of testing and optimizing, you should consider several factors including equity curve, net profit, percent profitable, profit factor (dollars won per dollar lost), average trade, and maximum drawdown.

**9.** Backward and forward test your strategy on the out-of-sample data you reserved. The test results will probably not be as good as the results on the data for which your system was optimized. However, for your strategy to be tradable, the backward and forward tests should yield favorable results. Your strategy is unlikely to perform better in the future than it did on the out-of-sample data. Check your strategy's performance on the same key factors that you evaluated during your test of the sample data (equity curve, net profit, percent profitable, profit factor, average trade, and maximum drawdown).

**10.** Trade your strategy with consistency, confidence, and courage.

CHAPTER 1

# High Five

Moving averages, the most popular of all technical indicators, smooth price data so that a trend can be identified easily and objectively. TradeStation 2000i offers several types of moving averages including simple, weighted, exponential, adaptive, displaced, and triangular averages. In addition to the many types of moving averages, there are lots of ways that moving averages can be used. Here are four examples of setups for a long position based on moving averages of closing prices:

1. A moving average is rising
2. The close is above a moving average
3. A fast moving average (e.g. a five-bar average) crosses above a slow moving average (e.g. a 20-bar average)
4. A four-bar moving average is greater than a nine-bar moving average, and the nine-bar moving average is greater than an 18-bar moving average.

Our High Five strategy employs five simple moving averages to find a setup and a price move in the direction of the setup to define a trigger. The default value for the base moving average is five, and the next four averages are each twice the length of the previous average; thus, with a base average of five, the rest of the averages in the series are 10, 20, 40, and 80. (The rules for selling short are analogous to buying long; we'll just describe the rules for the long side of the market here.) A buy setup is in place when all five moving averages are greater than they were one bar ago. The entry trigger is a price move equal to the close plus 50% of the setup bar's range. In other words, when all five moving averages are rising at the close of the previous bar, we'll buy at that close plus half of the difference between the bar's high and low. We'll exit our long position on the next open when at least three of the moving averages are declining. As with many of our other strategies, the entry setup and the exit are based on similar conditions, but the exit is more sensitive than the entry. In High Five, we require all five moving averages to be rising for a buy setup, but only three of the averages to be declining to signal an exit. We enter trades cautiously (when all our ducks are in a row) but we exit aggressively (as soon as some of our ducks get out-of-line).

We added three items to High Five in StrategyBuilder: an ATR (AverageTrueRange) Protective Stop, an ATR Breakeven Stop, and a LastBar Exit. The protective stop keeps losses from getting out-of-hand, and the breakeven stop moves our stop to the trade's entry price when the trade has gone in our favor by a specified amount. The LastBar Exit closes out positions on the last bar of the test data, so that the profits or losses are included in the Performance Summary.

## Defining Our Trading Rules

For the High Five strategy, we defined long and short setups, triggers, orders, and exits. We also calculated five simple moving averages and each bar's range. The strategy's rules are described next.

### Long and Short Setups

a) If all five moving averages are rising, we have a setup to buy.

b) If all five moving averages are declining, we have a setup to sell short.

### Long and Short Triggers

a) The trigger for a long position is a rally to the close of the setup bar plus 50% of the setup bar's range.

b) The trigger for a short position is a decline to the close of the setup bar minus 50% of the setup bar's range.

### Long and Short Orders

a) Place a buy stop order to enter a long position.

b) Place a sell stop order to enter a short position.

### Long and Short Exits

a) Exit long and short positions at an ATR Protective Stop or an ATR Breakeven Stop

b) Exit long positions on the next open when at least three of the five simple moving averages turn down

c) Exit short positions on the next open when at least three of the five simple moving averages turn up

## Designing & Formatting

This section presents the EasyLanguage instructions and formatting for the strategy, with the EasyLanguage instructions broken down and explained line by line.

### EasyLanguage Strategy Components: High Five (STAD11: HighFive)

### Strategy Inputs (STAD11: High Five)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| Breakeven_ATRs | 4 | The floor value, the number of Average True Ranges above/below the Entry Price at which the Stop becomes active for the position |
| ATRLength | 10 | The number of bars used in the calculation of the Average True Range (ATR) |
| Price | Close | The price value the average calculations are based on |
| BaseLength | 5 | The fastest length parameter of the five averages |
| Multiplier | .2 | The portion of the signal bar's Range that is added (or subtracted) from the Close of the signal bar, used to calculate the entry stop value |
| Protective_ATRs | 3 | The number of Average True Ranges (ATR) used as the protective stop |

### Signal Components:

**1.** High Five

**2.** ATR Protective Stop

**3.** ATR Breakeven Stop

**4.** Last Bar Exit

## EasyLanguage Signal: High Five:

```
Inputs: Price(Close), BaseLength(5), Multiplier(.2);
Variables: MP(0), BuySetup(False), SellSetup(False);
Variables: LEntryValue(0), SEntryValue(0), LongCounter(0), ShortCounter(0);
Arrays: Averages[5](0), LongConditions[5](False), ShortConditions[5](False);

MP = MarketPosition;
LongCounter = 0;
ShortCounter = 0;

If MP <> MP[1] Then Begin
        BuySetup = False;
        SellSetup = False;
End;

For Increment = 1 to 5 Begin
        Averages[Increment] = Average(Price, BaseLength * Power(2, Increment-1));
End;

For Increment = 1 to 5 Begin
        LongConditions[Increment] = Averages[Increment] > Averages[Increment][1];
        If LongConditions[Increment] Then
                LongCounter = LongCounter + 1;
End;

For Increment = 1 to 5 Begin
        ShortConditions[Increment] = Averages[Increment] < Averages[Increment][1];
        If ShortConditions[Increment] Then
                ShortCounter = ShortCounter + 1;
End;

If LongCounter = 5 Then Begin
        BuySetup = True;
        LEntryValue = Close + (Range * Multiplier);
End;

If ShortCounter = 5 Then Begin
        SellSetup = True;
        SEntryValue = Close - (Range * Multiplier);
End;

If MarketPosition <> 1 AND BuySetup Then
        Buy Next Bar at LEntryValue Stop;

If MarketPosition <> -1 AND SellSetup Then
        Sell Next Bar at SEntryValue Stop;

If LongCounter <= 2 Then
        ExitLong Next Bar at Market;

If ShortCounter <= 2 Then
        ExitShort Next Bar at Market;
```

## Signal Inputs (High Five)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| Price | Close | The price value the average calculations are based on |
| BaseLength | 5 | The fastest length parameter of the five averages |
| Multiplier | .2 | The portion of the signal bar's Range that is added (or subtracted) from the Close of the signal bar, used to calculate the entry stop value |

## Signal Variables (High Five)

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| MP | 0 | [Numeric] Holds the MarketPosition value |
| BuySetup | False | [True/False] Triggers the Long Entry order after setup occurs, until the condition is filled or expired |
| SellSetup | False | [True/False] Triggers the Short Entry order after setup occurs, until the condition is filled or expired |
| LEntryValue | 0 | [Numeric] Holds the value calculated for the Long Entry stop |
| SentryValue | 0 | [Numeric] Holds the value calculated for the Long Entry stop |
| LongCounter | 0 | [Numeric] Holds the value of how many averages are rising on the current bar |
| ShortCounter | 0 | [Numeric] Holds the value of how many averages are falling on the current bar |
| Increment | 0 | [Numeric] Used to increment through each of the For loops |

## Signal Arrays (High Five)

| ARRAY | ELEMENTS | DEFAULT | DESCRIPTION |
|---|---|---|---|
| Averages | 5 | 0 | [Numeric] Holds the average, with each element calculated with varying lengths based on the Input BaseLength raised to a power of 2 |
| LongConditions | 5 | False | [True/False] Used to determine that a corresponding Average is rising |
| ShortConditions | 5 | False | [True/False] Used to determine that a corresponding Average is falling |

## Setup

First, we will assign the MarketPosition reserved word to a variable so that we can reference previous values in subsequent steps. MarketPosition will return 1 for a Long position, -1 for a Short position and 0 if no position is taken. The LongCounter and ShortCounter variables are reset to 0 at the beginning of each bar, they are used as accumulators for the number of rising or falling averages (the accumulation is discussed in detail below). Finally, if there has been a change in the position of the market either by entry or exit, the BuySetup and SellSetup flags are reset to False, awaiting future evaluation of the rising or falling averages.

```
MP = MarketPosition;
LongCounter = 0;
ShortCounter = 0;

If MP <> MP[1] Then Begin
        BuySetup = False;
        SellSetup = False;
End;
```

The five Averages are calculated and stored in the array Averages, in each of 5 elements. By using a For loop, we can calculate all of the averages with one line, rather than repeating a similar calculation five times. Averages is declared with five elements and each Array element will be assigned a value. The Length parameter is the only item that will be modified, for each new calculation it is based on the BaseLength raised to a power of 2. The exponent used is Increment - 1, where Increment is the variable we are using as a counter.

```
For Increment = 1 to 5 Begin
        Averages[Increment] = Average(Price, BaseLength * Power(2, Increment-1));
End;
```

To count how many averages are either rising or falling, we again use a For loop. Each element of Averages is compared to its previous value, and the result of each condition is stored in the corresponding element of the Arrays LongConditions or ShortConditions. As each element of LongConditions / ShortConditions evaluates to True, we increment the respective LongCounter / ShortCounter variable.

```
For Increment = 1 to 5 Begin
        LongConditions[Increment] = Averages[Increment] > Averages[Increment][1];
        If LongConditions[Increment] Then
                LongCounter = LongCounter + 1;
End;

For Increment = 1 to 5 Begin
        ShortConditions[Increment] = Averages[Increment] < Averages[Increment][1];
        If ShortConditions[Increment] Then
                ShortCounter = ShortCounter + 1;
End;
```

If all of the averages are either rising or falling, the LongCounter or ShortCounter variable (respectively) will reach 5. This triggers the setup for an entry order, and we calculate the stop value for entry with the current Close, plus or minus a portion of the Range.

```
If LongCounter = 5 Then Begin
        BuySetup = True;
        LEntryValue = Close + (Range * Multiplier);
End;

If ShortCounter = 5 Then Begin
        SellSetup = True;
        SEntryValue = Close - (Range * Multiplier);
End;
```

## Long Entry

If BuySetup is set to True and the market is not already in a Long position, the entry order is generated for the next bar at the pre-calculated stop value.

```
If MarketPosition <> 1 AND BuySetup Then
        Buy Next Bar at LEntryValue Stop;
```

## Short Entry

If SellSetup is set to True and the market is not already in a Short position, the entry order is generated for the next bar at the pre-calculated stop value.

```
If MarketPosition <> -1 AND SellSetup Then
        Sell Next Bar at SEntryValue Stop;
```

### Long Exits

Our Long Exit condition is when three of the five averages have turned down. We can determine this by checking that they are no longer rising. If so, the LongCounter variable will not be incremented to a value higher than 2.

```
If LongCounter <= 2 Then
        ExitLong Next Bar at Market;
```

### Short Exits

On the Short side, our Exit condition is when three of the five averages have turned up. Again, we can determine that when they are no longer falling, the ShortCounter variable will not be incremented to a value higher than 2.

```
If ShortCounter <= 2 Then
        ExitShort Next Bar at Market;
```

## EasyLanguage Signal: ATR Breakeven Stop:

** See Common Stops Appendix

## EasyLanguage Signal: ATR Protective Stop:

** See Common Stops Appendix

## EasyLanguage Signal: Last Bar Exit:

** See Common Stops Appendix

# Testing & Improving

We tested the High Five strategy on approximately seven years of daily data for American Express (AXP), Cisco Systems (CSCO), and Japanese Yen futures. For AXP and CSCO, we traded the long side only and deducted 10 cents per share for slippage and five cents per share for commission. For the Yen, we traded both the long and short sides, deducting $40 per contract for slippage and $10 per contract for commission.

Let's start with AXP.

The optimized values for AXP are as follows (using a ten-bar ATR):

ATR Breakeven Stop = 7 ATRs

ATR Protective Stop = 5 ATRs

BaseLength = 20

Multiplier = .20


*Figure 1. AXP chart*

The daily bar chart of AXP [Figure. 1, AXP chart] displays a recent winning trade and the five simple moving averages needed to generate the setup and the exit. Applied to AXP, High Five earned $5,867 (per 100 shares) on ten trades [Figure 2, AXP Strategy Performance Report]. Eighty percent of the trades were profitable, with the average trade netting $586. Even with such a high winning percentage, the average winner was larger than the average loser by a factor of 2.37 to 1. The Profit Factor was a formidable 9.47 to 1, meaning that High Five won $9.47 for each $1.00 it lost. Figure 3 lists the strategy's results for AXP on a trade-by-trade basis [Figure 3, Trades]. The Annual Trading Summary shows the strategy's results on both a year-by-year basis and an annual rolling period basis [Figure 4, Annual Trading Summary].

Let's take a look at our strategy's performance on CSCO.

The optimized values for CSCO are as follows (using a ten-bar ATR):

ATR Breakeven Stop = 5 ATRs

ATR Protective Stop = 4 ATRs

BaseLength = 15

Multiplier = .60

CSCO's daily bar chart, five moving averages, and trade example are shown in Figure 5 [Figure 5, CSCO chart]. The Performance Summary [Figure 6, CSCO Performance Summary] indicates that High Five made 13 trades, resulting in a gain of $4,935 (per 100 shares). Sixty-one percent of the trades were profitable, the average winner was 16.9 times as large as the average loser, and the strategy earned $27.03 for each $1.00 it lost.


*Figure 2. AXP Strategy Performance Report*


*Figure 3. AXP Trades*

TradeStation Strategy Performance Report

## Annual Trading Summary

***Annual Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| YTD | $1,555.00 | 4.58% | 2.36 | 3 | 66.67% |
| 12 month | $1,555.00 | 5.44% | 1.53 | 8 | 50.00% |
| 98 | $745.00 | 2.24% | 1.93 | 2 | 50.00% |
| 97 | $1,045.00 | 3.25% | N/A | 3 | 100.00% |
| 96 | $957.50 | 3.07% | 64.83 | 3 | 66.67% |
| 95 | $1,037.50 | 3.44% | N/A | 1 | 100.00% |
| 94 | $47.50 | 0.16% | N/A | 1 | 100.00% |
| 93 | $140.10 | 0.47% | N/A | 1 | 100.00% |

***Annual Rolling Period Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| 99 | $1,555.00 | 4.58% | 2.36 | 3 | 66.67% |
| 98-99 | $2,300.00 | 6.92% | 2.18 | 5 | 60.00% |
| 97-99 | $3,345.00 | 10.39% | 2.72 | 8 | 75.00% |
| 96-99 | $4,302.50 | 13.78% | 3.20 | 11 | 72.73% |
| 95-99 | $5,340.00 | 17.69% | 3.73 | 12 | 75.00% |
| 94-99 | $5,387.50 | 17.87% | 3.75 | 13 | 76.92% |
| 93-99 | $5,527.60 | 18.43% | 3.82 | 14 | 78.57% |

*Figure 4. Annual Trading Summary*


*Figure 5. CSCO chart*


*Figure 6. CSCO Performance Summary*

Let's see how our strategy performed on Japanese Yen futures.

The optimized values for JY are as follows (using a ten-bar ATR):

ATR Breakeven Stop = 4 ATRs

ATR Protective Stop = 5 ATRs

BaseLength = 5

Multiplier = .70

Figure 7 is a daily bar chart of the Yen with five simple moving averages and an example of a winning trade [Figure 7, Yen chart]. Applied to the Yen, High Five earned $65,812 on 46 trades, with 47 % of the trades profitable [Figure 8, Yen Performance Summary]. The largest winner was $17,625 compared to a largest loser of $4,725. The average winner ($4,377) was 3.44 times as large as the average loser ($1,270) and the average trade (winners and losers) gained $1,430 per contract. The Profit Factor (3.16) indicates that our strategy won $3.16 for each $1.00 it lost. High Five let profits run by staying in winning trades an average of 38 bars, while it cut losses short by exiting losing trades in an average of 13 bars.

Our next graphic, the Annual Trading Summary, shows that the strategy made money in seven of the last eight years and that all of the Annual Rolling Periods were profitable [Figure 9, Yen Annual Trading Summary]. The Equity Curve illustrates the strategy's struggle through the first 20 trades and a strong comeback through the rest of the test period [Figure 10, Yen Equity Curve].

The Total Trades graph [Figure 11, Yen Total Trades] demonstrates the importance of positive outliers in a trendfollowing strategy (a positive outlier is a trade more than three standard deviations greater than the average trade). Our strategy's performance wouldn't look so promising without the one positive outlier (trade 24).


*Figure 7. Yen chart*


TradeStation Strategy Performance Report - STAD11: High Five JY-CONT-Daily

**Performance Summary: All Trades**

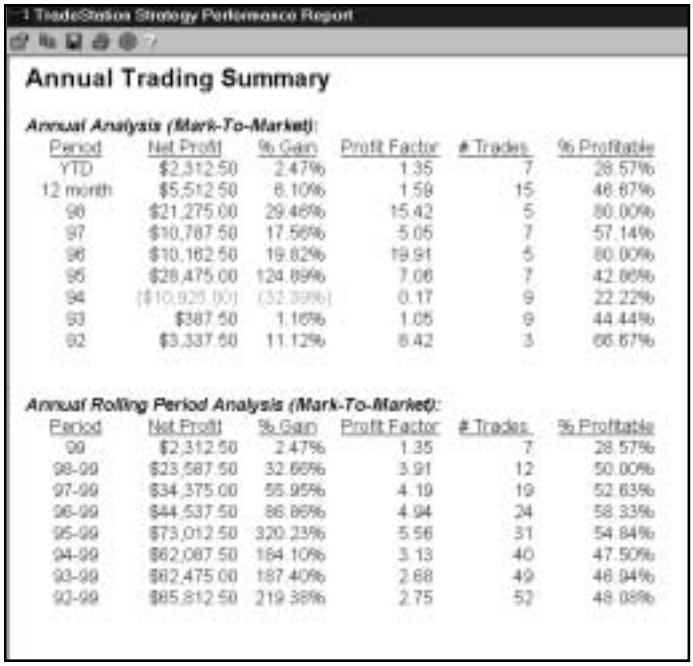
| | | | |
|---|---|---|---|
| Total Net Profit | $85,812.50 | Open position P/L | $0.00 |
| Gross Profit | $96,312.50 | Gross Loss | ($10,500.00) |
| Total # of trades | 46 | Percent profitable | 47.83% |
| Number winning trades | 22 | Number losing trades | 24 |
| Largest winning trade | $17,625.00 | Largest losing trade | ($4,725.00) |
| Average winning trade | $4,377.84 | Average losing trade | ($1,270.83) |
| Ratio avg win/avg loss | 3.44 | Avg trade (win & loss) | $1,430.71 |
| Max consec. Winners | 8 | Max consec. losers | 7 |
| Avg # bars in winners | 36 | Avg # bars in losers | 13 |
| Max intraday drawdown | ($19,525.00) | | |
| Profit Factor | 9.16 | Max # contracts held | 1 |
| Account size required | $22,525.00 | Return on account | 292.18% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $22,212.50 | Open position P/L | $0.00 |
| Gross Profit | $41,625.00 | Gross Loss | ($19,412.50) |
| Total # of trades | 22 | Percent profitable | 40.91% |
| Number winning trades | 9 | Number losing trades | 13 |
| Largest winning trade | $13,412.50 | Largest losing trade | ($4,725.00) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win | Loss | Time | Graphs | Settings

*Figure 8. Yen Performance Summary*



TradeStation Strategy Performance Report

## Annual Trading Summary

***Annual Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| YTD | $2,312.50 | 2.47% | 1.35 | 7 | 28.57% |
| 12 month | $5,512.50 | 6.10% | 1.59 | 15 | 46.67% |
| 98 | $21,275.00 | 29.46% | 15.42 | 5 | 80.00% |
| 97 | $10,787.50 | 17.56% | 5.05 | 7 | 57.14% |
| 96 | $10,162.50 | 19.82% | 19.91 | 5 | 80.00% |
| 95 | $28,475.00 | 124.89% | 7.06 | 7 | 42.86% |
| 94 | ($10,925.00) | (32.39%) | 0.17 | 9 | 22.22% |
| 93 | $387.50 | 1.16% | 1.05 | 9 | 44.44% |
| 92 | $3,337.50 | 11.12% | 6.42 | 3 | 66.67% |

***Annual Rolling Period Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| 99 | $2,312.50 | 2.47% | 1.35 | 7 | 28.57% |
| 98-99 | $23,587.50 | 32.66% | 3.91 | 12 | 50.00% |
| 97-99 | $34,375.00 | 55.95% | 4.19 | 19 | 52.63% |
| 96-99 | $44,537.50 | 86.86% | 4.94 | 24 | 58.33% |
| 95-99 | $73,012.50 | 320.23% | 5.56 | 31 | 54.84% |
| 94-99 | $62,087.50 | 184.10% | 3.13 | 40 | 47.50% |
| 93-99 | $62,475.00 | 187.40% | 2.68 | 49 | 46.94% |
| 92-99 | $65,812.50 | 219.38% | 2.75 | 52 | 48.08% |

*Figure 9. Yen Annual Trading Summary*

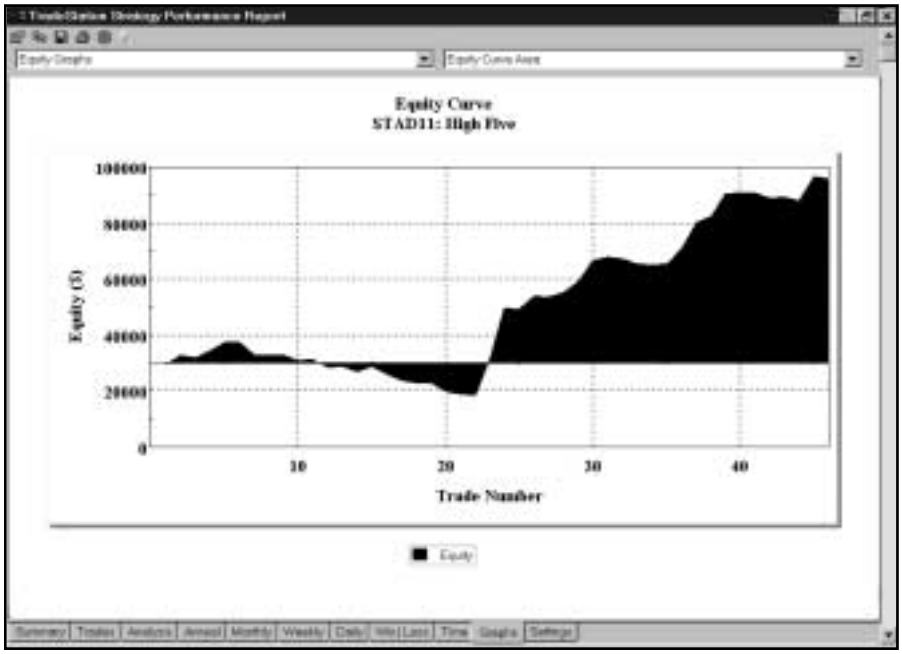



*Figure 10. Yen Equity Curve*

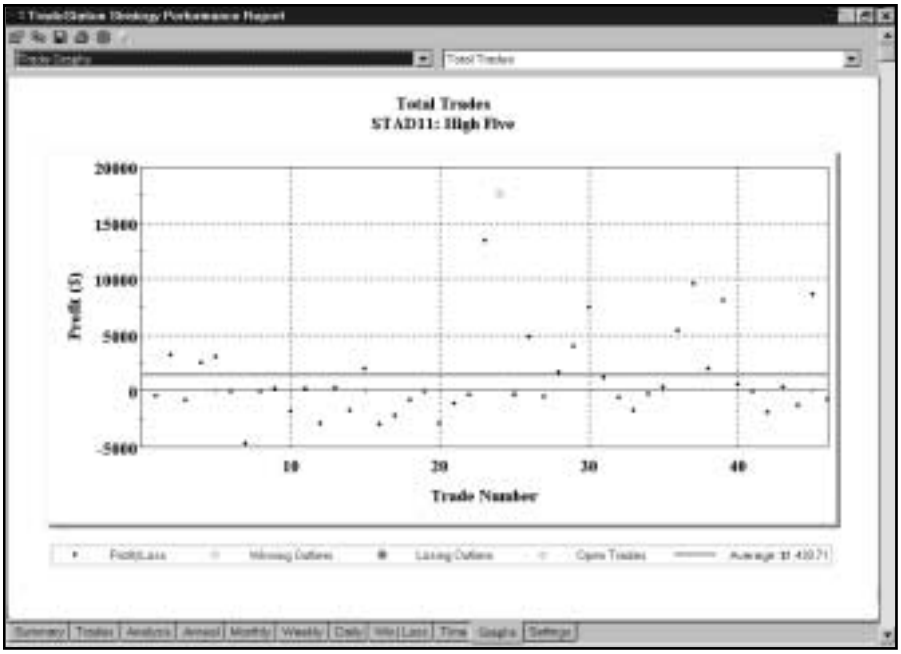



*Figure 11. Yen Total Trades*

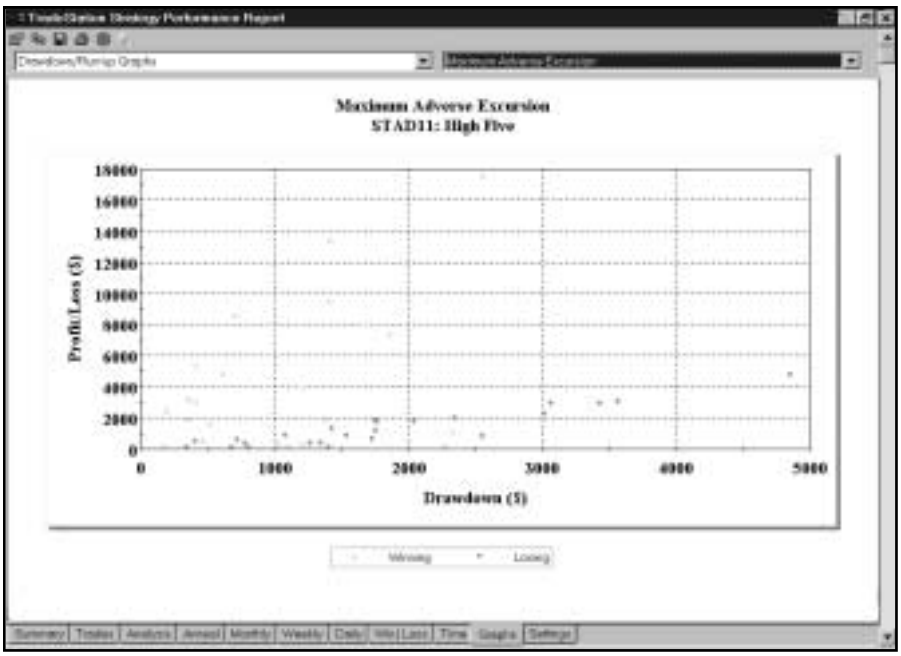



*Figure 12. Yen Maximum Adverse Excursion*

Maximum Adverse Excursion [Figure 12, Yen Maximum Adverse Excursion] plots profit/loss on the vertical axis and each trade's worst drawdown (maximum adverse excursion) on the horizontal axis. Upward-pointing triangles indicate winning trades, and downward-pointing triangles indicate losing trades. Note that no trades turned around and became winners after suffering a drawdown greater than $2,700. This information can help us manage the High Five strategy's trades in this market: we shouldn't hold on to trades that go against us by more than $2,700.

## Suggestions for Improvement

To keep things as simple as possible, we tested and optimized the five moving averages in this strategy by specifying a base number of bars for the lookback period of the first moving average and doubling that number for each of the succeeding moving averages. In other words, if the first moving average was five, the next four averages would be 10, 20, 40, and 80. It's possible that the strategy's performance could be improved by optimizing the multiplier — doubling may not be the best choice. Also, using a constant multiplier (for example, doubling or tripling the length of each moving average) isn't the only possible way to get multiple averages. You might want to optimize the series of averages through a range of values. For example, you could test 3-7 for average one, 8-12 for average two, 13-17 for average three, 18-22 for average four, and 23-27 for average five.

CHAPTER 2

# Hit The Bull’s Eye

Our Hit the Bull's Eye strategy is based on the well-known technique of buying a dip in an uptrend and selling a rally in a downtrend. We'll use a price-channel breakout and an exponential moving average to determine the trend and a return to the midpoint of a recent bar to define the countertrend dips and rallies. An unusual feature of this strategy is that it takes profits at a fixed target instead of trailing a stop in an attempt to catch a major trend.

The rules for the long and short sides of the strategy are analogous (the short side is the mirror image of the long side) so we'll just explain the long side here. The first condition for a buy setup is a breakout to a new 10-bar high with the breakout bar's close above the high two bars ago. The second condition is a rising moving average — an average greater than it was on the previous bar. When both conditions are true, our buy setup is complete, and we begin monitoring for the buy trigger. Call the bar that made the new 10-bar high Bar #1 and call the bar two bars before it Bar #2. The trigger is a decline to the midpoint of Bar #2. The buy setup remains in effect for a specified number of bars. Once we're in a long position, we'll place a limit order to sell (take profits) at the high of Bar #1 plus a multiple of the difference between Bar #1's high and our entry price. For example, say we're long XYZ at $80 per share, and the high of Bar #1 is $90. The difference between our entry price and the Bar #1 high is $10. If the multiplier we're using is 4, our profit target would be $120 ($10 x 4 = $40 + $80 = $120).

# Defining Our Trading Rules

For the Hit the Bull's Eye strategy, we defined long and short setups, triggers, orders, and exits. We also calculated an exponential moving average and the midpoint of each bar. The setups, triggers, orders, and exits are described next.

## Long and Short Setups

a) The setup for a long position is a new 10-bar high, a close above the close two bars ago, and a rising exponential moving average.

b) The setup for a short position is a new 10-bar low, a close below the close two bars ago, and a falling exponential moving average.

## Long and Short Triggers

a) The trigger for a long position is a decline to the midpoint of the bar two bars before the setup bar. The trigger must be hit within three bars of the setup bar, or the setup is cancelled.

b) The trigger for a short position is a rally to the midpoint of the bar two bars before the setup bar. The trigger must be hit within three bars of the setup bar, or the setup is cancelled.

## Long and Short Orders

a) To enter a long position, place a limit order at the midpoint of Bar #2 (the bar two bars before the setup bar).

b) To enter a short position, place a limit order at the midpoint of Bar #2 (the bar two bars before the setup bar).

## Long and Short Exits

a) Exit a long position at the high of the setup bar plus a multiple of the difference between that high and the trade's entry price.

b) Exit a short position at the low of the setup bar plus a multiple of the difference between the trade's entry price and that low.

# Designing & Formatting

This section presents the EasyLanguage instructions and formatting for the strategy, with the EasyLanguage instructions broken down and explained line by line.

## EasyLanguage Strategy Components: Hit the Bull's Eye (STAD11: The Bullseye)

### Strategy Inputs (STAD11: The Bullseye)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| RangeLength | 5 | The number of bars used to determine a new High or Low has occurred |
| XAvgLength | 50 | The length used to calculate a rising or falling Exponential Average |
| BarstoEnter | 5 | The number of bars that an entry signal will remain valid |
| Factor | 3 | The factor used to calculate the actual stop levels for profit stops based on the signal bar |
| ATRs | 3 | The number of Average True Ranges (ATR) used as the protective stop |
| ATRLength | 10 | The number of bars used in the calculation of the Average True Range (ATR) |

### Signal Components:

1. Hit the Bull's Eye
2. ATR Protective Stop
3. Last Bar Exit

### EasyLanguage Signal: Hit the Bull's Eye:

```
Inputs: RangeLength(5), XAvgLength(50), BarstoEnter(5), Factor(3);
Variables: BuyEntry(0), BuySetup(False), BuyCounter(0), LongExitTarget(0);
Variables: SellEntry(0), SellSetup(False), SellCounter(0), ShortExitTarget(0);

Condition1 = High = Highest(High, RangeLength);
Condition2 = Close > High[2];
Condition3 = XAverage(Close, XAvgLength) > XAverage(Close, XAvgLength)[1];
Condition4 = Low = Lowest(Low, RangeLength);
Condition5 = Close < Low[2];
Condition6 = XAverage(Close, XAvgLength) < XAverage(Close, XAvgLength)[1];
```

```
If MarketPosition <> 1 AND Condition1 AND Condition2 AND Condition3 Then Begin
        BuyEntry = MedianPrice[2];
        BuyCounter = 0;
        BuySetup = True;
        LongExitTarget = High + Factor * (High - BuyEntry);
End;

If BuyCounter > BarstoEnter Then
        BuySetup = False
Else
        BuyCounter = BuyCounter + 1;

If MarketPosition = 1 Then Begin
        BuySetup = False;
        ExitLong Next Bar at LongExitTarget Limit;
End;

If BuySetup Then
        Buy Next Bar at BuyEntry Limit;

If MarketPosition <> -1 AND Condition4 AND Condition5 AND Condition6 Then Begin
        SellEntry = MedianPrice[2];
        SellCounter = 0;
        SellSetup = True;
        ShortExitTarget = Low - Factor * (SellEntry - Low);
End;

If SellCounter > BarstoEnter Then
        SellSetup = False
Else
        SellCounter = SellCounter + 1;

If MarketPosition = -1 Then Begin
        SellSetup = False;
        ExitShort Next Bar at ShortExitTarget Limit;
End;

If SellSetup Then
        Sell Next Bar at SellEntry Limit;
```

## Signal Inputs (Hit the Bull's Eye)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| RangeLength | 5 | The number of bars used to determine a new High or Low has occurred |
| XAvgLength | 50 | The length used to calculate a rising or falling Exponential Average |
| BarstoEnter | 5 | The number of bars that an entry signal will remain valid |
| Factor | 3 | The factor used to calculate the actual stop levels for profit stops based on the signal bar |

## Signal Variables (Main Signal Name)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| BuyEntry | 0 | [Numeric] Holds the value calculated for the Long Entry order |
| BuySetup | False | [True/False] Triggers the Long Entry order after setup occurs, until the condition is filled or expired |
| BuyCounter | 0 | [Numeric] Counts the number of bars that have passed since the setup of a Long Entry |
| LongExitTarget | 0 | [Numeric] Stores the value to be used to Exit a Long position at a profit |
| SellEntry | 0 | [Numeric] Holds the value calculated for the Short Entry order |
| SellSetup | False | [True/False] Triggers the Short Entry order after setup occurs, until the condition is filled or expired |
| SellCounter | 0 | [Numeric] Counts the number of bars that have passed since the setup of a Short Entry |
| ShortExitTarget | 0 | [Numeric] Stores the value to be used to Exit a Short position at a profit |

## Setup

The setup conditions for Long Entry are a new "n"-bar high where the Close of that bar is greater than the High of two bars ago. In addition, the exponential moving average should be rising. We can evaluate these conditions by using the RangeLength (Input) for "n" and check if the current High is equal to the highest High of RangeLength bars. Conditions 2 and 3 are used to evaluate the Close being Higher than 2 bars ago and a rising value returned by the XAverage() function. On the Short side, the setup conditions are the opposite, a new lowest Low of RangeLength bars, the Close lower than the Low of two bars ago and a falling exponential moving average. These conditions are evaluated in a similar manner in the independent conditions 4, 5 and 6.

```
Condition1 = High = Highest(High, RangeLength);
Condition2 = Close > High[2];
Condition3 = XAverage(Close, XAvgLength) > XAverage(Close, XAvgLength)[1];

Condition4 = Low = Lowest(Low, RangeLength);
Condition5 = Close < Low[2];
Condition6 = XAverage(Close, XAvgLength) < XAverage(Close, XAvgLength)[1];
```

If conditions 1, 2 and 3 occur on the same bar and the market is not already in a Long position, the BuyEntry variable is set to the MedianPrice of 2 bars ago, which is used as our entry price into a Long position. We set the BuySetup True/False variable to True, allowing us to generate the Buy order based on its True/False status. BuyCounter is set to 0, allowing us to limit the number of bars the setup will last. In addition LongExitTarget is calculated based on the signal bar for use in the Exit, if the Long position is taken later on.

```
If MarketPosition <> 1 AND Condition1 AND Condition2 AND Condition3 Then Begin
        BuyEntry = MedianPrice[2];
        BuyCounter = 0;
        BuySetup = True;
        LongExitTarget = High + Factor * (High - BuyEntry);
End;
```

On the Short side, if conditions 4, 5 and 6 occur on the same bar and the market is not already in a Short position, the current bar becomes our setup bar. The SellEntry variable is set to the MedianPrice of 2 bars ago to be used as our entry price into a Short position. We set the SellSetup True/False variable to True, allowing us to generate the Sell order based on its True/False status and the SellCounter is set to 0, allowing us to limit the number of bars the setup will last. Corresponding, the ShortExitTarget is calculated based on the signal bar for use in the Exit, if a Short position is taken later on.

```
If MarketPosition <> -1 AND Condition4 AND Condition5 AND Condition6 Then Begin
        SellEntry = MedianPrice[2];
        SellCounter = 0;
        SellSetup = True;
        ShortExitTarget = Low - Factor * (SellEntry - Low);
End;
```

## Long Entry

BuyCounter is used to count the number of bars elapsed where the BuySetup variable remains True. BuyCounter is compared to BarstoEnter (Input). If BuyCounter is greater, BuySetup will be turned to False, preventing further Long Entry orders. Otherwise, BuyCounter is incremented by one.

```
If BuyCounter > BarstoEnter Then
        BuySetup = False
Else
        BuyCounter = BuyCounter + 1;
```

If BuySetup remains True, the Long Entry order will be generated for the next bar at the value represented with BuyEntry, the MedianPrice two bars previous to the signal bar.

```
If BuySetup Then
        Buy Next Bar at BuyEntry Limit;
```

## Short Entry

SellCounter is used to count the number of bars elapsed where the SellSetup variable remains True. SellCounter is compared to BarstoEnter. If SellCounter is greater, SellSetup will be turned to False, preventing further Short Entry orders. Otherwise, SellCounter is incremented by one.

```
If SellCounter > BarstoEnter Then
        SellSetup = False
Else
        SellCounter = SellCounter + 1;
```

If the SellSetup remains True, the Short Entry order will be generated for the next bar at the value represented with SellEntry, the MedianPrice two bars previous to the signal bar.

```
If SellSetup Then
        Sell Next Bar at SellEntry Limit;
```

## Profit Stops

If a Long position is entered, the BuySetup variable is set to False and the Exit order for a profit is generated. LongExitTarget is the value of the High of the Signal bar plus a multiple of the difference between that High and the MedainPrice of two bars previous.

```
If MarketPosition = 1 Then Begin
        BuySetup = False;
        ExitLong Next Bar at LongExitTarget Limit;
```

```
End;
```

If a Short position is entered, the SellSetup variable is set to False and the Exit order for a profit is generated. ShortExitTarget is the value of the Low of the Signal bar minus a multiple of the difference between that Low and the MedianPrice of two bars previous.

```
If MarketPosition = -1 Then Begin
        SellSetup = False;
        ExitShort Next Bar at ShortExitTarget Limit;
End;
```

## EasyLanguage Signal: ATR Protective Stop

** See Common Stops Appendix

## EasyLanguage Signal: Last Bar Exit

** See Common Stops Appendix

# Testing & Improving

We tested the Hit the Bull's Eye strategy on daily data for IBM, Microsoft (MSFT), and Crude Oil (CL) from 2/92 - 11/99. For IBM and MSFT, we traded a fixed unit of 100 shares, deducting $.10 per share for slippage and $.05 per share for commission. For CL, we traded one contract and deducted $40 per trade for slippage and $10 per trade for commission.

Let's begin our survey of the test results with IBM. Following are the optimized values (using a 10-bar ATR):

ATR Protective Stop = 5 ATRs

ATR Breakeven Stop = 5 ATRs

LE Range Length = 5

LE Xavg Length = 10

LE Bars to Enter = 5

LE Factor = 5

The daily bar chart [Figure 1, IBM chart] shows a series of three recent Bull's Eye trades. Note the excellent exits and prompt re-entries. Applied to IBM, our strategy earned $9,413 on 27 trades, with 48% of the trades profitable [Figure 2, IBM Performance Summary]. The average winning trade ($899) was 5.5 times as large as the average losing trade ($162), and the average trade (wins & losses) was $348. Bull's Eye earned $5.15 for each $1.00 it lost.

The equity curve fluctuates in a narrow range above and below breakeven (the $30,000 account size) before

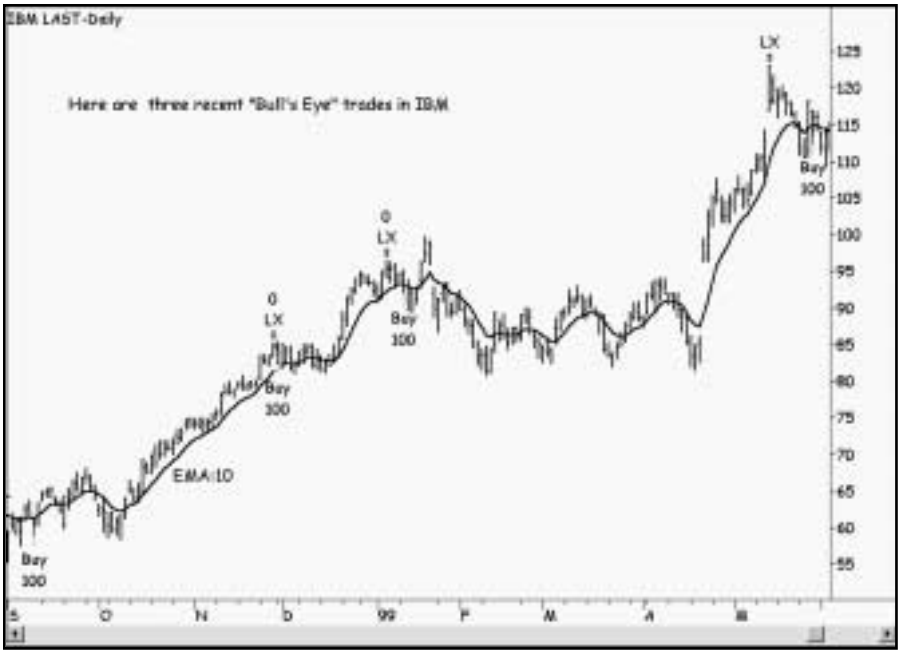



*Figure 1. IBM chart*


*Figure 2. IBM Performance Summary*

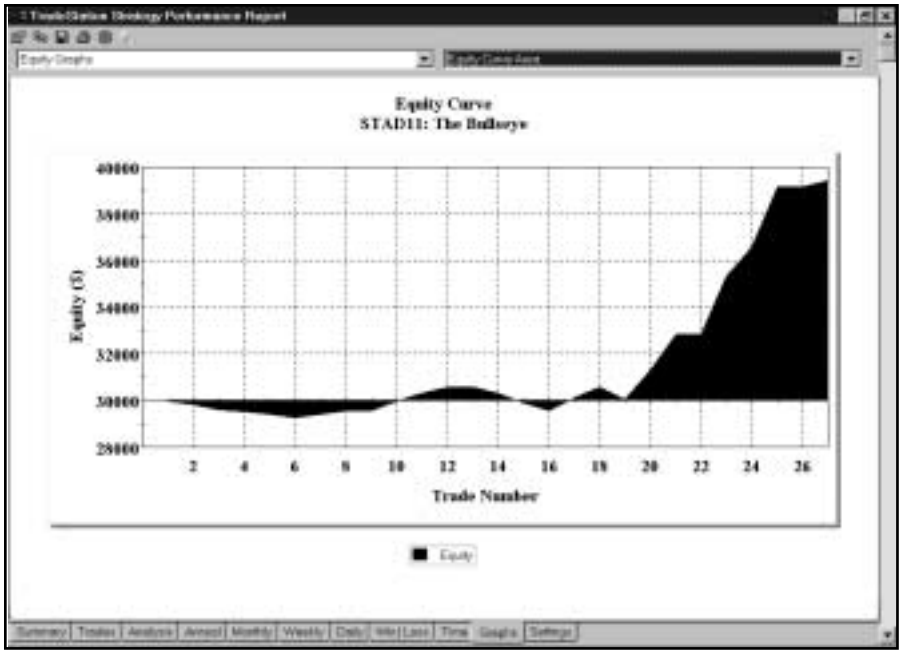



*Figure 3. IBM Equity Curve*

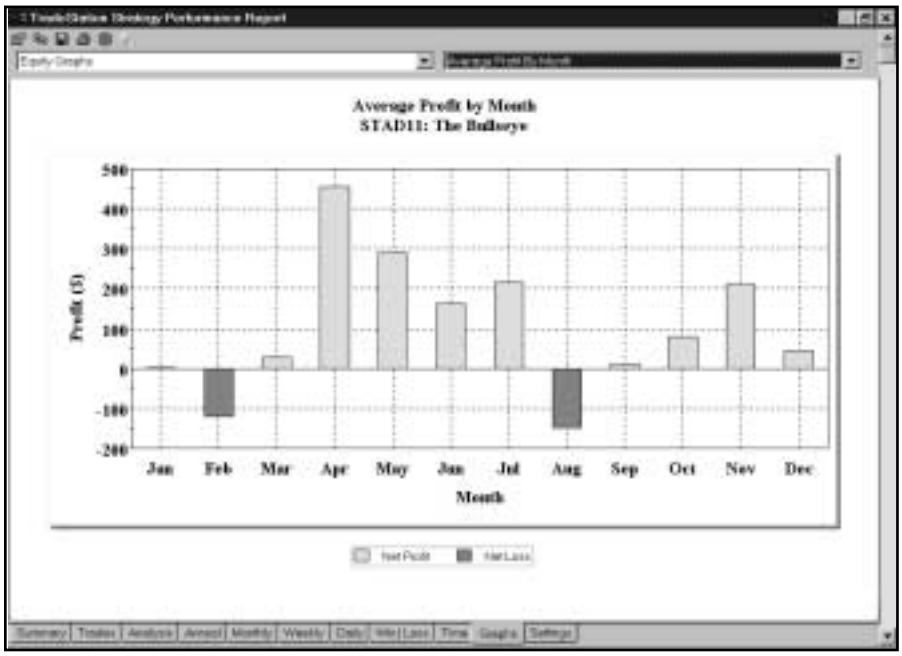



*Figure 4. IBM Average Profit by Month*

rising sharply from trade 19 to trade 26 (Figure 3, IBM Equity Curve). The graph of Average Profit by Month shows that the strategy lost money in only two months (February and August) when monthly returns are averaged over the length of the test period (Figure 4, IBM Average Profit by Month).

Bull's Eye's total trades are graphed in Figure 5 [Figure 5, IBM Total Trades]. Note that our strategy produced no positive or negative outliers (trades three standard deviations or more greater than the average trade). Our ATR Protective Stop kept us from losing too much on trades that moved against us; our profit target gave us quicker and more consistent profits than we would have gotten with a pure trendfollowing strategy and a trailing stop.

Next, let's turn our attention to Bull's Eye's performance on MSFT. The optimized values are as follows:

ATR Protective Stop = 5 ATRs

ATR Breakeven Stop = 6 ATRs

LE Range Length = 5

LE Xavg Length = 30

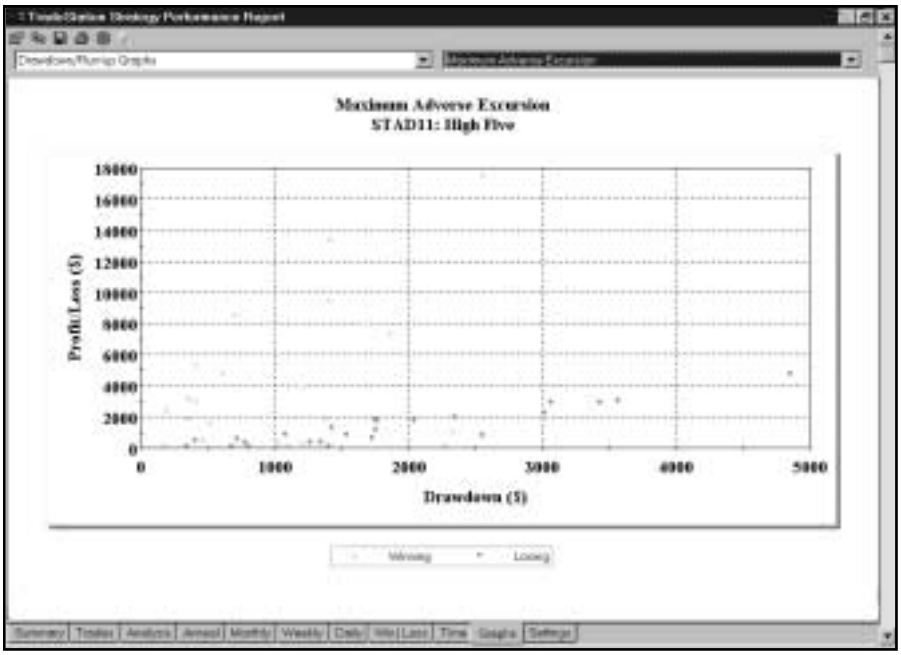



*Figure 5. IBM Total Trades*

LE Bars to Enter = 3

LE Factor = 5

Applied to MSFT, Bull's Eye netted a profit of $8,502 on 19 trades. Seventy-three percent of the trades were profitable, and the average winning trade was 2.16 times as large as the average losing trade [Figure 6, MSFT Performance Summary]. The average trade (wins and losses) made $447 for each 100 shares traded and made $6.06 for each $1.00 it lost.

There were nine consecutive winners but only two consecutive losers. The equity curve illustrates the strategy's


*Figure 6. MSFT Performance Summary*

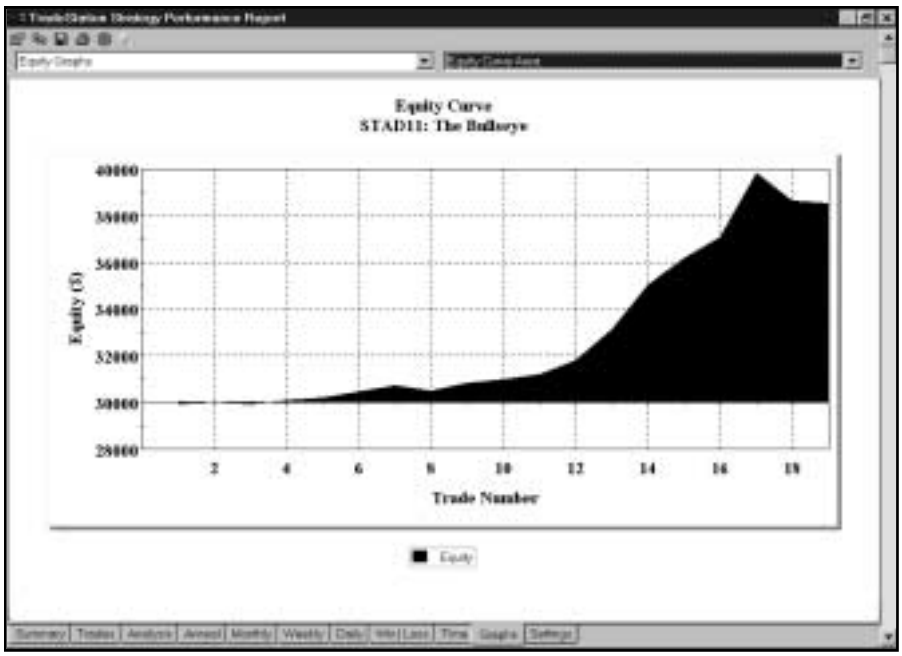



*Figure 7. MSFT Equity Curve*

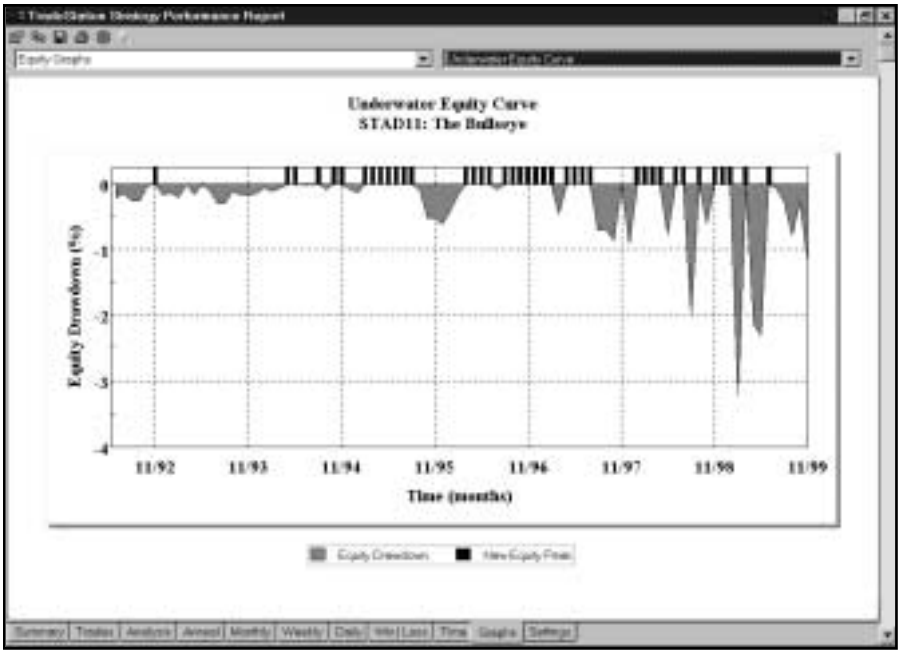



*Figure 8. MSFT Underwater Equity Curve*

modest results over the first ten trades and its strong results through trade 17 [Figure 7, MSFT Equity Curve]. Drawdowns were very manageable, with the worst three drawdowns temporarily giving back only two to three percent of equity [Figure 8, MSFT Underwater Equity Curve].

Finally, let's consider Bull's Eye's results on Crude Oil futures. Here are the optimized values:

ATR Protective Stop = 5 ATRs

ATR Breakeven Stop = 6 ATRs

LE Range Length = 15

LE Xavg Length = 60

LE Bars to Enter = 3

LE Factor = 3

SE Range Length = 20

SE Xavg Length = 60

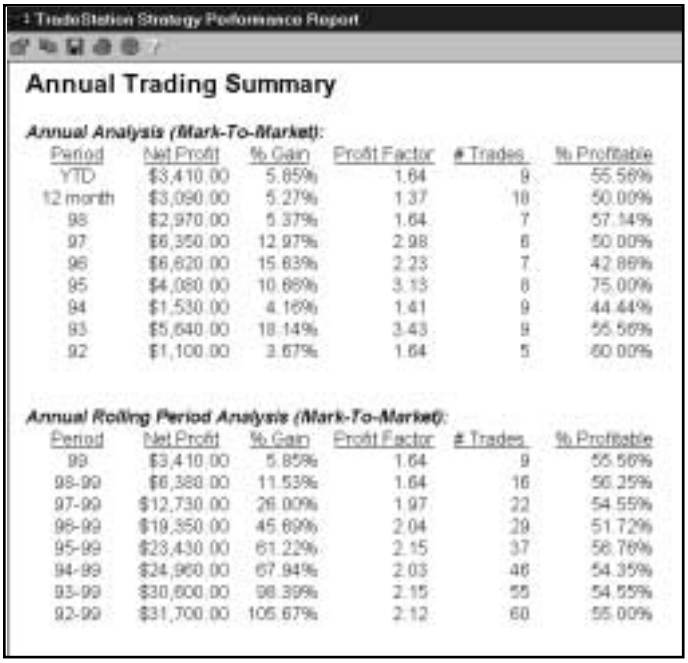
SE Bars to Enter = 2

SE Factor = 4

The daily bar chart of CL displays several recent Bull's Eye trades [Figure 9, CL chart]. Our strategy generated


*Figure 9. CL chart*


*Figure 10. CL Performance Summary*



TradeStation Strategy Performance Report

## Annual Trading Summary

*Annual Analysis (Mark-To-Market):*

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| YTD | $3,410.00 | 5.85% | 1.64 | 9 | 55.56% |
| 12 month | $3,090.00 | 5.27% | 1.37 | 18 | 50.00% |
| 98 | $2,970.00 | 5.37% | 1.64 | 7 | 57.14% |
| 97 | $6,350.00 | 12.97% | 2.98 | 6 | 50.00% |
| 96 | $6,620.00 | 15.63% | 2.23 | 7 | 42.86% |
| 95 | $4,080.00 | 10.66% | 3.13 | 8 | 75.00% |
| 94 | $1,530.00 | 4.16% | 1.41 | 9 | 44.44% |
| 93 | $5,640.00 | 18.14% | 3.43 | 9 | 55.56% |
| 92 | $1,100.00 | 3.67% | 1.64 | 5 | 60.00% |

*Annual Rolling Period Analysis (Mark-To-Market):*

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| 99 | $3,410.00 | 5.85% | 1.64 | 9 | 55.56% |
| 98-99 | $6,380.00 | 11.53% | 1.64 | 16 | 56.25% |
| 97-99 | $12,730.00 | 26.00% | 1.97 | 22 | 54.55% |
| 96-99 | $19,350.00 | 45.69% | 2.04 | 29 | 51.72% |
| 95-99 | $23,430.00 | 61.22% | 2.15 | 37 | 56.76% |
| 94-99 | $24,960.00 | 67.94% | 2.03 | 46 | 54.35% |
| 93-99 | $30,600.00 | 98.39% | 2.15 | 55 | 54.55% |
| 92-99 | $31,700.00 | 105.67% | 2.12 | 60 | 55.00% |

*Figure 11. CL Annual Trading Summary*

$29,550 in net profits on 52 trades with 55% winners [Figure 10, CL Performance Summary]. The average trade (wins and losses) earned $568, and the strategy gained $2.13 for each $1.00 it lost. The Annual Trading Summary [Figure 11, CL Annual Trading Summary] indicates that all eight years in the test period were profitable.

Bull's Eye's equity curve is encouraging [Figure 12, CL Equity Curve], and the Underwater Equity Curve confirms our strategy's strong performance: only three drawdowns reached eight percent of equity, while 27 new

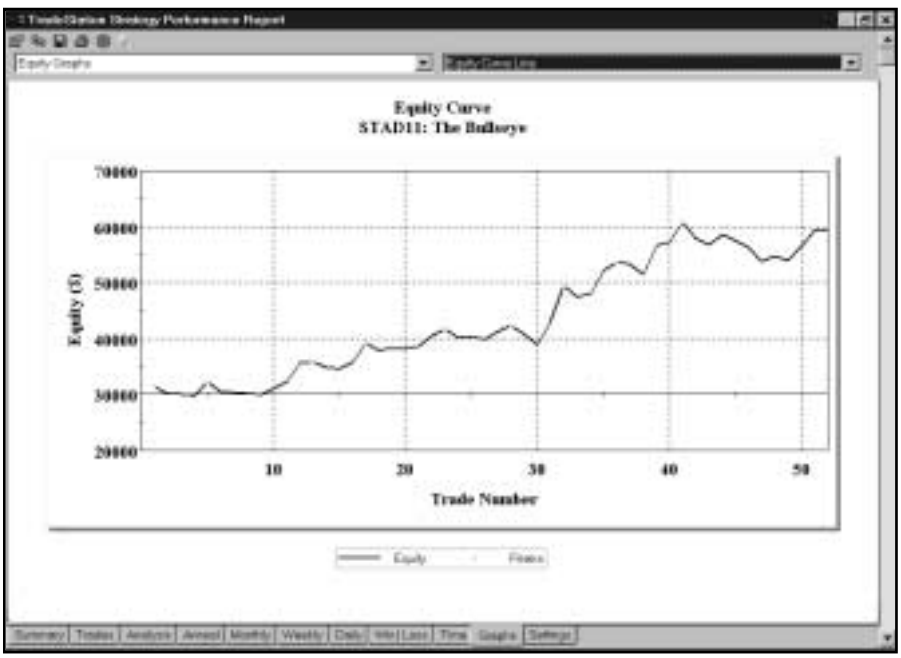



*Figure 12. CL Equity Curve*

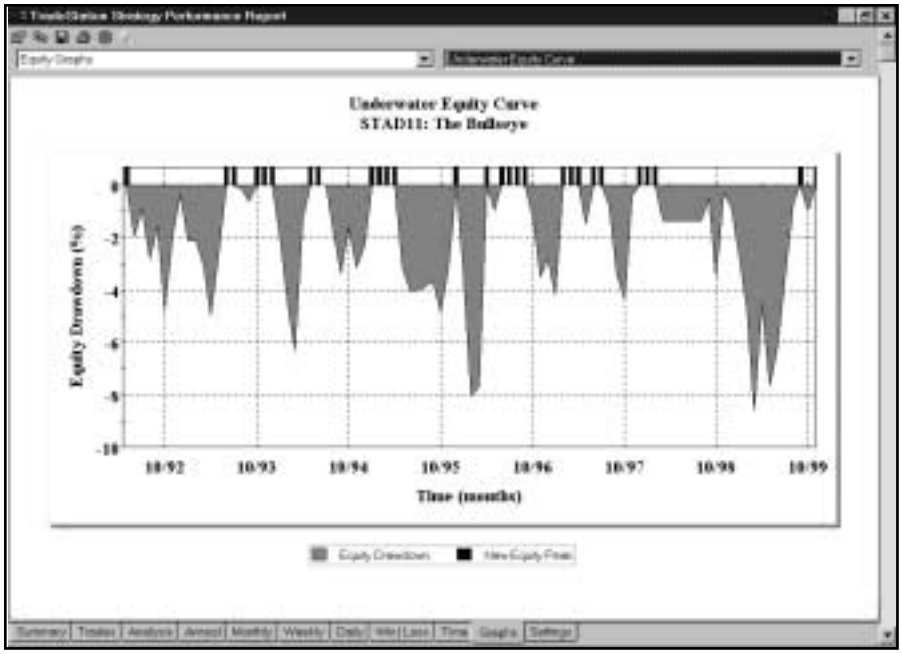



*Figure 13. CL Underwater Equity Curve*

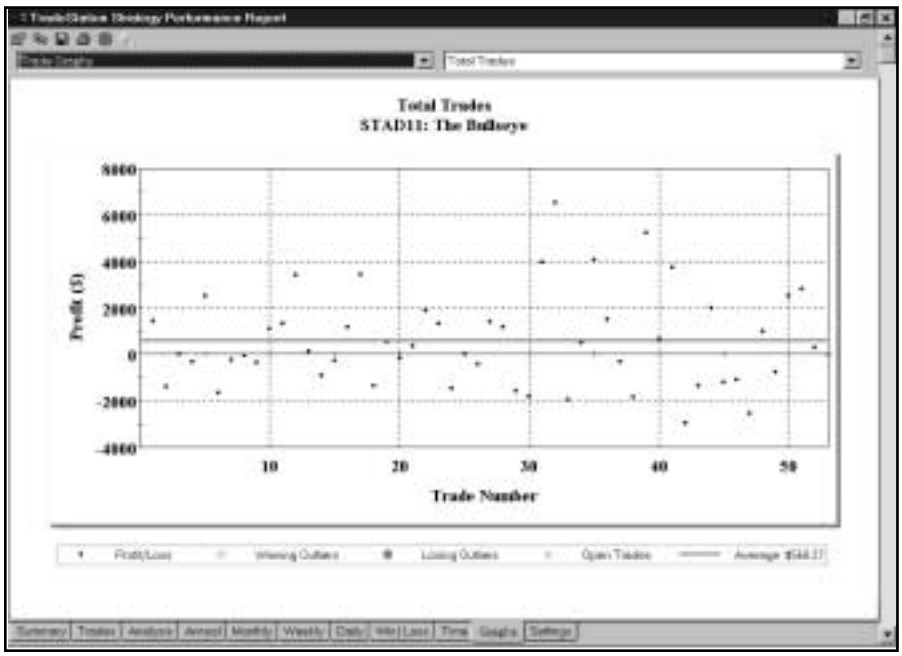



*Figure 14. CL Total Trades*

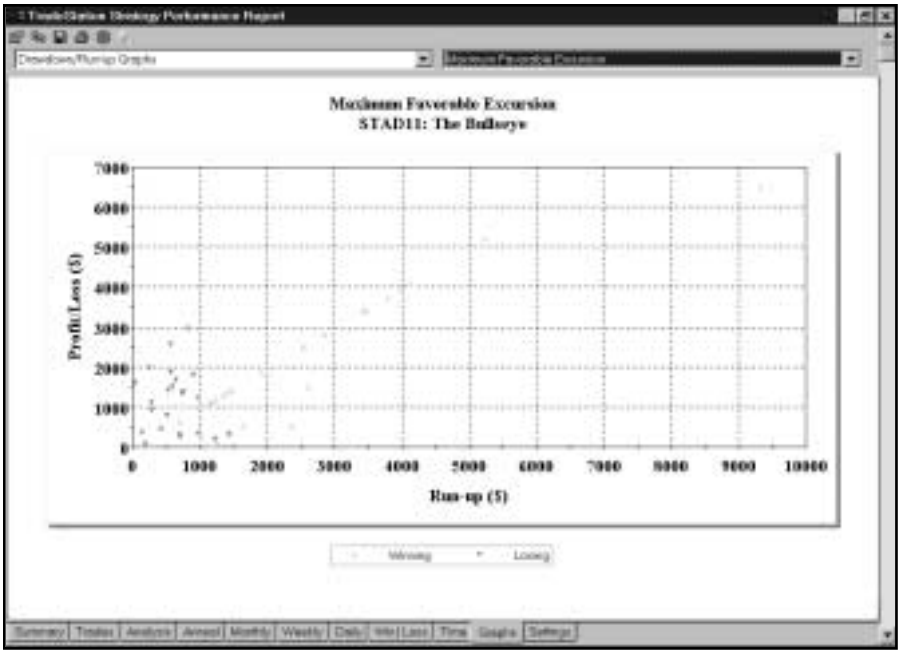



*Figure 15. CL Maximum Favorable Excursion*

equity highs were achieved [Figure 13, CL Underwater Equity Curve]. The graph of Total Trades shows that there were no positive or negative outliers (trades three standard deviations or more greater than the average trade [Figure 14, CL Total Trades]. We're always pleased to see a lack of negative outliers (big losers) and we understand that there were no positive outliers because we exited winning trades at a relatively modest profit target instead of trying to hit an occassional homerun by exiting at a trailing stop.

Figure 15 graphs the dollar value of each trade's result and its maximum favorable excursion [Figure 15, CL Maximum Favorable Excursion]. No trades turned into losers once they had attained an open profit of at least $1,500. Also note that since we exited winning trades at a profit target rather than at a trailing stop, the winners-by definition-- exited at their equity peaks. When you take profits at a target, you don't give back any gains you have reached, but you also don't let profits run until the trend runs out of steam. Many successful traders take profits on half their position at a target and let profits run with a trailing stop on the other half.

## Suggestions For Improvement

Although Bull's Eye employs both a price-channel breakout and an exponential moving average to determine the trend, it sometimes signals setups when a market has been trading in a relatively choppy manner. You might be able to improve the strategy by adding a filter that tells you when a market is trending well enough to be a good candidate for a trade. The ADX indicator is well-suited to this purpose. If ADX is greater than 20 and rising, chances are good that the market is trending. You may want to test Bull's Eye with the ADX filter-taking a pass on setups that occur when ADX is below 20 and/or falling.

CHAPTER 3

# MISER (Murray's Intermarket Strategy Revisited)

Our MISER strategy is based on an idea by Murray Ruggiero, the author of *Cybernetic Trading Strategies*, the publisher of the *Inside Advantage* newsletter, a contributing editor to Futures magazine, and the subject of an Omega Research Magazine interview in the fall of 1998.

Murray's Intermarket Strategy takes advantage of the correlation between US Treasury Bond futures (US) and S&P 500 Index futures (SP). His premise is that strong Bonds and a weak S&P make a bullish scenario for stocks, while weak Bonds and a strong S&P make a bearish scenario for stocks. The rules for Murray's strategy are incredibly simple: when Bonds close above their 26-day simple moving average, and the S&P closes below its 16-day simple moving average, buy the S&P on the next open. When Bonds close below their 26-day simple moving average, and the S&P closes above its 16-day simple moving average, sell the S&P on the next open. For STAD Club, we just reoptimized the values for the moving averages and added our favorite stops from TradeStation 2000i's Strategy Builder. We named our version Murray's Intermarket Strategy Revisited – MISER for short.

## Defining Our Trading Rules

For the MISER strategy, we defined long and short entries and exits. We also calculated two simple moving averages and the ten-day ATR (Average True Range). The entries and exits are described next.

### Long and Short Entries

a) If US is above its simple moving average (SMA), and SP is below its SMA, buy SP on the next open.

b) If US is below its SMA, and SP is above its SMA, sell short SP on the next open.

### Long and Short Exits

a) If long, stop-and-reverse on a sell signal; also, exit a long position at the protective stop, the breakeven stop, the trailing stop, the volatility stop, or the big-profit stop.

b) If short, stop-and-reverse on a buy signal; also, exit a short position at the protective stop, the breakeven stop, the trailing stop, the volatility stop, or the big-profit stop.

## Designing & Formatting

This section presents the EasyLanguage instructions and formatting for the strategy, with the EasyLanguage instructions broken down and explained line by line.

## EasyLanguage Strategy Components: MISER (STAD11: MISER)

### Strategy Inputs (STAD11: MISER)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| TrailingStop | 3 | The number of Average True Ranges to use |
| BreakevenStop | 4 | The floor value, the number of Average True Ranges above/below the Entry Price at which the Stop becomes active for the position |
| ATRLength | 10 | Length, expressed in bars, used to calculate the Average True Range |
| SPLength | 16 | The length parameter used to calculated the moving average of the S&P |
| BondsLength | 26 | The length parameter used to calculated the moving average of the Bond |
| VolatilityStop | 2 | The number of Average True Ranges that are used to determine the required volatility to place an Exit order |
| ProtectiveStop | 3 | The number of Average True Ranges that are risked in the position |
| BigProfitStop | 7 | Number of Average True Ranges used to determine the "Big Profit" level |
| ExitBarLen | 3 | Length, expressed in bars, used to determine the number of bars used in the trailing stop after the "Big Profit" level has been achieved |

### Signal Components:

1. MISER
2. ATR Breakeven Stop
3. ATR Big Profit Stop
4. ATR Protective Stop
5. ATR Trailing LX
6. ATR Volatility Stop
7. Last Bar Exit

## EasyLanguage Signal: MISER:

```
Inputs: SPLength(16), BondsLength(26);
Variables: BondsAvg(0), SPAvg(0);

SPAvg = Average(Close of Data1, SPLength);
BondsAvg = Average(Close of Data2, BondsLength);

If Close of Data1 < SPAvg AND Close of Data2 > BondsAvg Then
        Buy next bar at market;

If Close of Data1 > SPAvg AND Close of Data2 < BondsAvg Then
        ExitLong next bar at market;
```

## Signal Inputs (MISER)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| SPLength | 16 | The length parameter used to calculated the moving average of the S&P |
| BondsLength | 26 | The length parameter used to calculated the moving average of the Bonds |

## Signal Variables (MISER)

| VARIABLES | DEFAULT | DESCRIPTION |
|---|---|---|
| BondsAvg | 0 | [Numeric] Stores the average calculated for the second data stream, the Bonds |
| SPAvg | 0 | [Numeric] Stores the average calculated for the first data stream, the S&P |

## Setup

The setup of this system is simple enough, we calculate the average for both the S&P (data1) and the Bonds (data2).

```
SPAvg = Average(Close of Data1, SPLength);
BondsAvg = Average(Close of Data2, BondsLength);
```

## Long Entry and Exit

If the S&P is below its average and the Bonds are above their respective average, the combination is a signal to Buy. If the opposite conditions occur, where the S&P is above its average and the Bonds are below theirs, then we exit the trade.

```
If Close of Data1 < SPAvg AND Close of Data2 > BondsAvg Then
        Buy next bar at market;

If Close of Data1 > SPAvg AND Close of Data2 < BondsAvg Then
        ExitLong next bar at market;
```

## EasyLanguage Signal: ATR Trailing LX:

```
Inputs: ATRs(3);
Variables: PosHigh(0), ATRVal(0);

ATRVal = AvgTrueRange(10) * ATRs;

If BarsSinceEntry = 0 Then
        PosHigh = High;

If MarketPosition = 1 Then Begin
        If High > PosHigh Then
                PosHigh = High;
        ExitLong ("ATR") Next Bar at PosHigh - ATRVal Stop;
End
else
        ExitLong ("ATR eb") Next Bar at High - ATRVal Stop;
```

## Signal Inputs (ATR Trailing LX)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| ATRs | 3 | The number of Average True Ranges to use |

## Signal Variables (ATR Trailing LX)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| PosHigh | 0 | [Numeric] Used to store the highest value of the position |
| ATRVal | 0 | [Numeric] Holds the average true range multiplied by the number of ATRs |

### Setup

In the Setup portion of the signal, the Average True Range is calculated and multiplied by the number of ATRs specified in the Inputs. Additionally, if BarSinceEntry returns 0, PosHigh gets assigned the value of the current High.

```
ATRVal = AvgTrueRange(10) * ATRs;

If BarsSinceEntry = 0 Then
        PosHigh = High;
```

### Long Exit

The PosHigh variable is used to keep track of the highest High of the position. The trailing Stop is placed at the PosHigh value minus the ATR calculation. We also generate an exit for the bar of entry at a price of the current High minus the ATR calculation.

```
If MarketPosition = 1 Then Begin
        If High > PosHigh Then
                PosHigh = High;
        ExitLong ("ATR") Next Bar at PosHigh - ATRVal Stop;
End
else
        ExitLong ("ATR eb") Next Bar at High - ATRVal Stop;
```

## EasyLanguage Signal: ATR Volatility Stop:

```
Inputs: VolatilityATRs(2), ATRLength(10);
Variable: ATRVal(0);

ATRVal = AvgTrueRange(ATRLength) * VolatilityATRs;

If MarketPosition = 1 Then
        ExitLong Next Bar at EntryPrice - ATRVal Stop;

If MarketPosition = -1 Then
        ExitShort Next Bar at EntryPrice + ATRVal Stop;
```

### Signal Inputs (ATR Volatility Stop)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| VolatilityATRs | 2 | The number of Average True Ranges that are used to determine the required volatility to place an Exit order |
| ATRLength | 10 | Length, expressed in bars, used to calculate the Average True Range |

### Signal Variables (ATR Volatility Stop)

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| ATRVal | 0 | [Numeric] Holds the value of the Average True Range multiplied by the number of Volatility ATRs |

### Setup

In the setup portion of this signal, the Average True Range is calculated and multiplied by the number of VolatilityATRs specified in the Inputs.

```
ATRVal = AvgTrueRange(ATRLength) * VolatilityATRs;
```

### Long Exit

When the market position is Long, a Long Exit is placed at the entry price minus the Volatility Average True Range calculation (ATRVal)

```
If MarketPosition = 1 Then
        ExitLong Next Bar at EntryPrice - ATRVal Stop;
```

### Short Exit

When the market position is Short, a Short Exit is placed at the entry price plus the Volatility Average True Range calculation (ATRVal).

```
If MarketPosition = -1 Then
        ExitShort Next Bar at EntryPrice + ATRVal Stop;
```

## EasyLanguage Signal: ATR Big Profit Stop

** See Common Stops Appendix

## EasyLanguage Signal: ATR Breakeven Stop

** See Common Stops Appendix

## EasyLanguage Signal: ATR Protective Stop

** See Common Stops Appendix

## EasyLanguage Signal: Last Bar Exit

** See Common Stops Appendix

## Testing & Improving

We described and coded both the long and short sides of this strategy; however (with stocks in the biggest bull market in history during our test period) we tested only the long side for this article. We tested MISER on daily data from 2/92 - 11/99, deducting $40 per trade for slippage and $10 per trade for commission. Margin for SP was set to $20,000 per contract, and maximum bars back was set to 50.

Following are the optimized values for MISER, our version of Murray's intermarket strategy:

ATR Breakeven Stop = 4 ATRs (using a ten-day ATR)

ATR Protective Stop = 4 ATRs

BigProfit Stop = 9 ATRs

ExitBarLength = 2

Volatility Stop = 3 ATRs

ATR Trailing Stop = 5 ATRs

SP SMA Length = 14

US SMA Length = 40

Figure 1 shows the daily chart of SP with its 14-day SMA and US with its 40-day SMA [Figure 1, SP and US chart]. We bought SP when it was weak (below its SMA) and US was strong (above its SMA). The Total Net Profit was $178,727 on 42 trades [Figure 2, SP Performance Summary]. Although the average winning trade was only .86 times as large as the average losing trade, 78% of the trades were profitable, and the average trade (wins and losses) earned $4,255 per contract. MISER posted seven consecutive winners versus only two consecutive losers and gained $314 for each $1.00 it lost.

The Equity Curve rose only gradually through the first 18 trades but improved thereafter [Figure 3, SP Equity Curve]. The Underwater Equity Curve shows only two significant drawdowns (one of about 12% and the other of about 14 %), while the strategy posted 35 new equity highs [Figure 4, SP Underwater Equity Curve].

MISER traded profitably in seven of the test period's eight years (losing $387 in 1994) [Figure 5, SP Annual Trading Summary]. Average Profit by Month was also very consistent, with the strategy earning money in eleven of twelve months when monthly results were averaged over the eight-year test [Figure 6, SP Average Profit by Month].


*Figure 1. SP and US chart*


*Figure 2. SP Performance Summary*

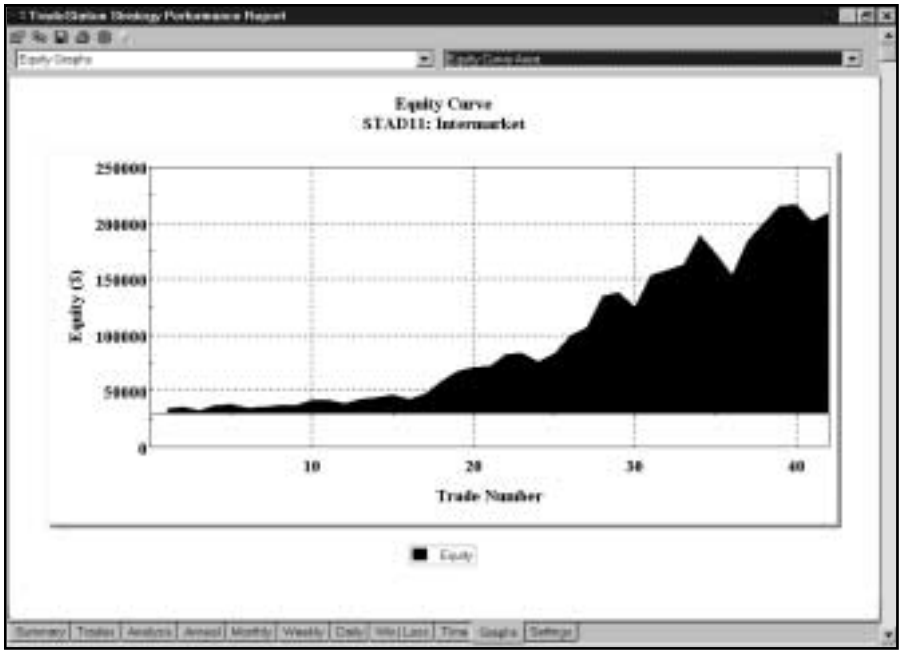



*Figure 3. SP Equity Curve*

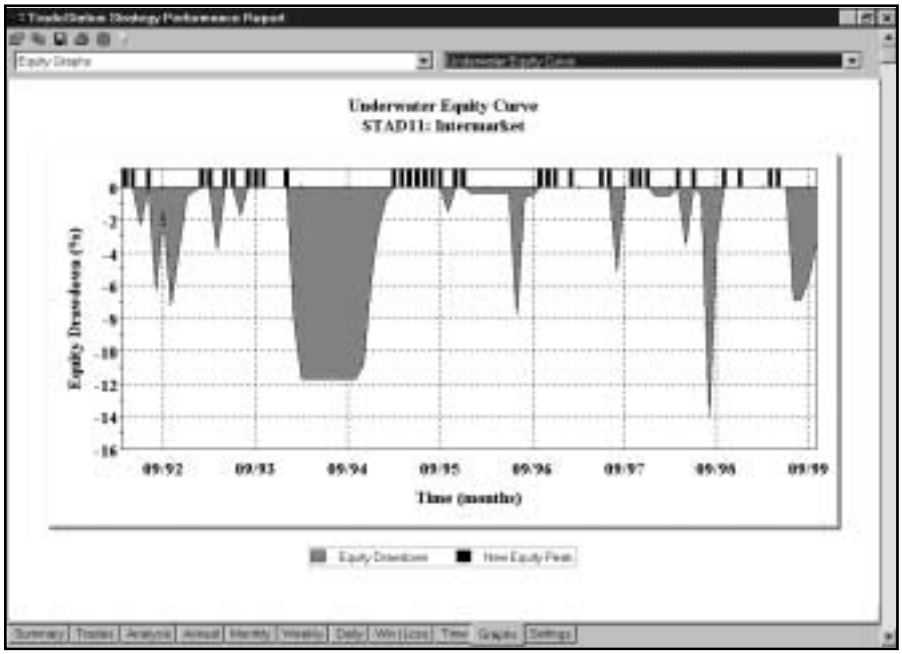
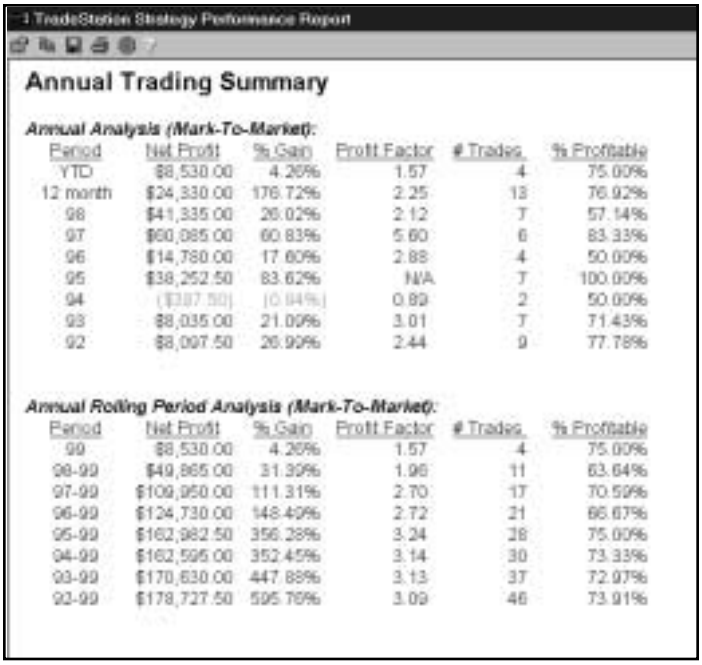



*Figure 4. SP Underwater Equity Curve*



TradeStation Strategy Performance Report

## Annual Trading Summary

***Annual Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| YTD | $8,530.00 | 4.26% | 1.57 | 4 | 75.00% |
| 12 month | $24,330.00 | 176.72% | 2.25 | 13 | 76.92% |
| 98 | $41,335.00 | 26.02% | 2.12 | 7 | 57.14% |
| 97 | $60,085.00 | 60.83% | 5.60 | 6 | 83.33% |
| 96 | $14,760.00 | 17.60% | 2.88 | 4 | 50.00% |
| 95 | $38,252.50 | 83.62% | N/A | 7 | 100.00% |
| 94 | ($387.50) | (0.84%) | 0.89 | 2 | 50.00% |
| 93 | $8,035.00 | 21.00% | 3.01 | 7 | 71.43% |
| 92 | $8,097.50 | 26.99% | 2.44 | 9 | 77.78% |

***Annual Rolling Period Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| 99 | $8,530.00 | 4.26% | 1.57 | 4 | 75.00% |
| 98-99 | $49,865.00 | 31.39% | 1.96 | 11 | 63.64% |
| 97-99 | $109,950.00 | 111.31% | 2.70 | 17 | 70.59% |
| 96-99 | $124,710.00 | 148.49% | 2.72 | 21 | 66.67% |
| 95-99 | $162,962.50 | 356.28% | 3.24 | 28 | 75.00% |
| 94-99 | $162,595.00 | 352.45% | 3.14 | 30 | 73.33% |
| 93-99 | $170,630.00 | 447.88% | 3.13 | 37 | 72.97% |
| 92-99 | $178,727.50 | 595.76% | 3.09 | 46 | 73.91% |

*Figure 5. SP Annual Trading Summary*

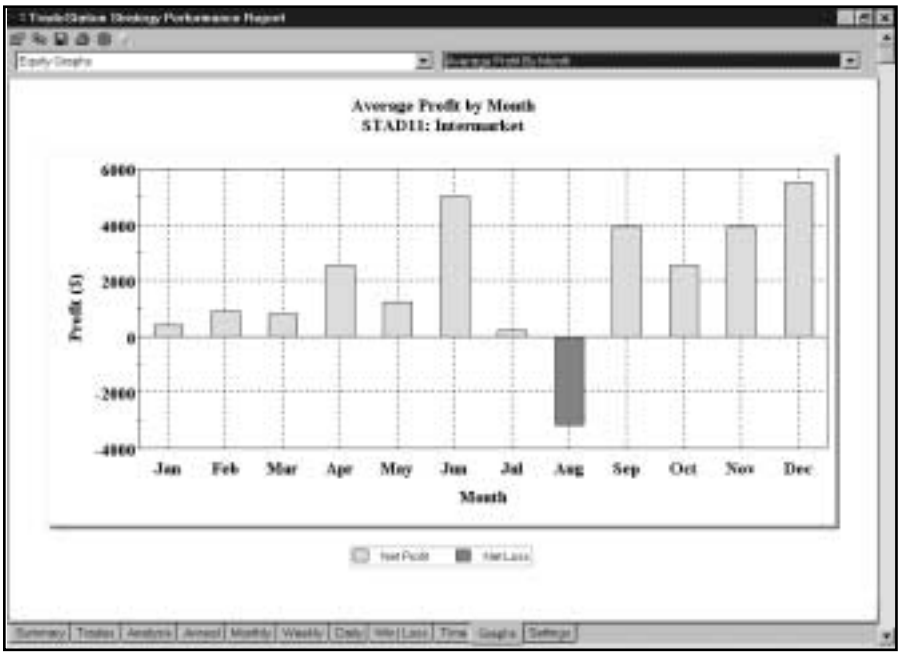



*Figure 6. SP Average Profit by Month*

## Suggestions for Improvement

In STAD Club, we usually specify both setups and triggers in the entry process; however, to keep MISER a bit closer to Murray's original strategy, we specified only a setup and simply entered a new position at the market on the next open. Our strategy could probably be improved by requiring an entry trigger that would provide confirmation of the setup. You might want to test placing your entry order for a long position a tick above the high of the setup bar or at the setup bar's close plus a multiple (.50 - 1.50) of the Average True Range.

CHAPTER 4

# Two Favorite Exit Strategies

***by Chuck LeBeau and Terence Tan***

We have often commented that too much attention is given to entry methods and too little attention is given to exit strategies. There have been studies done that demonstrate that the futures markets can be traded profitably with random entries, while using a systematic and logical exit strategy. For an example, Dr. Van K. Tharp, in his excellent book *Trade Your Way to Financial Freedom*, page 200, describes exactly such a study. The secret of course is to have good exits. In this brief article we will present two of our favorite exits and include some helpful advice on coding them. By the way, the Chandelier exit described below was the exit used by Dr. Tharp to make profits from his study of random entries.

Exit strategies serve specific functions. In some cases, exit strategies protect against a catastrophic loss in the trade. In other cases, exit strategies serve to protect profits from turning into losses. The Chandelier Exit and Yo-Yo exit are excellent strategies that help to protect open trade profits, while offering good protection against adverse price moves. As you will see, both exits are adaptable to current market volatility, widening during periods of higher volatility, and tightening during periods of lower volatility. We believe that adaptable exit strategies perform better than exit strategies that are fixed at a certain point or dollar distance away from the market.

# THE CHANDELIER EXIT

The Chandelier Exit is a trend-following exit. It is one of our preferred primary exits for trend following systems. The name is derived from the fact that the exit is hung downward from the ceiling or highest point of a trade. The Chandelier exit is designed to protect profit by limiting the maximum adverse move from the highest point reached (for a long trade) or the lowest point reached (for a short trade). The exit trails higher (or lower) as the prices move in the direction of the trade, and does not reverse in direction.

Specifically, the Chandelier exit stop is placed at a fixed multiple of the Average True Range (ATR) away from the highest high or highest close (for a long position) or lowest low or lowest close (for a short position) since the entry of the trade. As the prices move in the direction of the trade, the stop moves in the direction of the trade to protect profits. Although it is not specifically designed to move against a position, the stop may make minor retracements when the ATR expands.

How to calculate Average True Range (ATR).

Range: This is simply the difference between the high point and the low point of any bar.

True Range: This is the GREATEST of the following:

1. The distance from today's high to today's low
2. The distance from yesterday's close to today's high, or
3. The distance from yesterday's close to today's low

True range is different from range whenever there is a gap in prices from one bar to the next.

Average True Range is simply the true range averaged over a number of bars of data.

## Examples of the Chandelier Exit:

• Exit at the highest high since entry minus 3 ATR on a stop.

• Exit at the highest close since entry minus 2.5 ATR on a stop.

This exit is extremely effective at letting profits run in the direction of a trend while still offering some protection against a major reversal in trend. Our research has shown that when used for long term trend following, the best values for the multiple of ATR in most markets range between 2.5 and 4.0.

## THE YO-YO EXIT

This exit is similar to the Chandelier Exit except that the ATR stop is always pegged to the most recent close instead of the highest or lowest price reached in a trade. As a result of this logic, the Yo-Yo exit has an important difference from the Chandelier: the stop is able to move up and down depending on the current closing price (hence the Yo-Yo name). The Yo-Yo Exit is a classic volatility stop that is intended to recognize an abnormal adverse price fluctuation that occurs in one day (or in one bar if you are using bars other than daily). This abnormal volatility is often the result of a news event or some important technical reversal that is likely to signal the end of a trend. This logic makes the Yo-Yo exit very effective, and we seldom regret being stopped out whenever this exit is triggered.

We should caution that the Yo Yo stop should never be your only loss protection because there is a possibility of prices moving severely against your position without ever triggering this exit. For instance, if the closing price moves slowly against your position, the Yo-Yo stop also moves away each day and, in theory, there is a possibility that the stop may never be hit, and that you could suffer an infinite loss on a position.

The Yo-Yo exit should therefore always be used in conjunction with another exit strategy that guarantees an exit in the event of consecutive adverse price movements against your position. One possibility is to use a channel exit in conjunction with the Yo-Yo exit. Another possibility is to use a money management stop to limit the adverse price movement. This would, of course, not protect profits. Generally, we feel that the best combination is to use the Yo-Yo in conjunction with the Chandelier exit.

The Yo-Yo and the Chandelier exits work best when used together. The Chandelier exit sets a lower limit that a trade can reach before it is stopped out, and the Yo-Yo exit protects against sudden adverse volatility expansions. In our testing, we have observed good results with the Chandelier exit set at about 3 ATRs or more from a high point. Since this stop is never lowered, it will protect us against any gradual reversal of trend. With the Chandelier exit in place, good results can be obtained with the Yo-Yo exit set at a smaller multiple of ATR, for example, 1.5 to 2.0 ATRs from the most recent close to protect the position from unusual one day spikes in volatility. In actual trading, when these two exits are used together, the operative stop each day would be whichever of the two stops is closest to the position.

## How to code the EXITS

### Yo-Yo Exit

The coding of the Yo-Yo exit in Easylanguage is straightforward. The example below shows a segment of code implementing a Yo-Yo exit at 2 Average True Ranges from the most recent close. As discussed previously, this exit should never be used alone, but only in conjunction with another exit that sets a maximum limit to the loss that may be sustained in a trade.

```
If marketposition = 1 then ExitLong Close - 2 * AvgTrueRange(20) stop;
If marketposition = -1 then ExitShort Close + 2 * AvgTrueRange (20) stop;
```

## Chandelier Exit

The coding of the Chandelier exit is slightly more complex because there are no internal TradeStation functions that keep track of the highest high or lowest low reached in a trade.

The important programming problem therefore is to correctly determine the extreme price reached during a trade, and to make sure that none of the bars outside of the trade are taken into account. Using the simple expression Highest(High, N) with a constant N is obviously not acceptable, since the trade may last for a duration less than N, and we need to exclude all bars prior to the trade entry. Also, if the trade lasts for more than N bars, this method will also generate an incorrect result if the highest high occurred more than N bars ago.

Using the expression Highest(High, BarsSinceEntry) to track the highest high reached in a trade appears to be an obvious solution. However, this solution is not robust, since it generates an error when BarsSinceEntry exceeds the current setting of MaxBarsBack. TradeStation cannot refer to prices beyond the range of MaxBarsBack.

A more robust solution is to keep track of the extreme price using a variable, and to reset the variable whenever a trade is exited. The variable must be properly initialized at the beginning of each trade in order to ensure valid results, otherwise prices stored in the variable from previous trades would interfere with the results for the present trade. Based on this concept, we present two possible solutions for coding the Chandelier exit.

The first solution involves embedding the code into the system itself. The advantage of this method is that the initialization code can be included within the buy-sell condition to ensure that the variables are always correctly initialized.

In the example below, the variables MaxHigh and MinLow are used to keep track of the highest high and lowest low reached in long and short trades respectively:

## Example of Chandelier Exit

```
Var: MaxHigh(0), MinLow(0);

If Buy_Condition then begin
    Buy Open;
    MaxHigh = -999999;
End;

If Sell_Condition then begin
    Sell Open;
    MinLow = 999999;
End;

If MarketPosition = 1 then begin
    If H > MaxHigh then MaxHigh = H;
    ExitLong MaxHigh - 3 * AvgTrueRange(20) stop;
End;

If MarketPosition = -1 then begin
    If L < MinLow then MinLow = L;
    ExitShort MinLow + 3 * AvgTrueRange(20) stop;
End;
```

To ensure proper initialization, the initialization statements for the MaxHigh and MinLow variables are embedded within the buy and sell conditions respectively. This ensures that a new trade cannot be entered without proper initialization of these variables. Notice that MaxHigh is set to a very negative number and MinLow to a very large positive number at the beginning of each trade. MaxHigh should not be initialized to 0, since many back-adjusted continuous price series contain negative numbers, which may produce inaccurate results with an initial MaxHigh value of 0.

The other important pieces of code are embedded in the exit conditions. As the trade progresses through each bar in a long trade, the MaxHigh value is compared with the current bar's high price. If the current high exceeds MaxHigh, then MaxHigh is assigned the latest high price. If the present high price is lower than MaxHigh, MaxHigh automatically retains its previous higher value. In this way, MaxHigh keeps track of the maximum high reached in a trade. The reverse logic applies: MinLow keeps track of the lowest low in a short trade. Once MaxHigh and MinLow are correctly coded in the system, the implementation of the Chandelier exit is simple:

```
ExitLong MaxHigh - 3 * AvgTrueRange(20) stop;
ExitShort MinLow + 3 * AvgTrueRange(20) stop;
```

The disadvantage of coding MaxHigh and MinLow within the system code is that you need to repeat the same code over and over in different systems if you want to apply the same exit logic. The next solution presented overcomes this disadvantage.

In the second solution, we develop user functions that will keep track of the extreme price reached during a trade. The advantage of this method is that the code can then be easily re-used in any system that requires these functions without tedious re-coding.

Since we are no longer able to embed the initialization statements within the entry conditions, we must somehow "tell" the function when to reset its variables. The functions must therefore be able to detect when a trade has exited or when a new trade has been entered. The functions must also be able to correctly detect reversal trades. The functions must ensure that their values are reset whenever a trade is exited, and must keep track of the highest high and lowest low of long and short trades respectively. We also need to make sure that the functions are calculated on every bar, regardless of where they are placed in the system code. With these considerations in mind, we will proceed to present a workable version, and then explain why we coded it this way.

## Function: MaxTradeHigh

```
If Marketposition <> 1 then MaxTradeHigh = -999999;
If marketposition = 1 and H > MaxTradeHigh[1] then MaxTradeHigh = H;
```

## Function: MinTradeLow

```
If MarketPosition <> -1 then MinTradeLow = 999999;
If marketposition = -1 and L < MinTradeLow[1] then MinTradeLow = L;
```

The function MaxTradeHigh will keep track of the highest high reached during long trades, and the function MinTradeLow will keep track of the lowest low reached in short trades. We need to be clear on what these functions do not do. As it is presently coded, you cannot use the MaxTradeHigh function to detect the highest high in a short trade, nor use MinTradeLow to detect the lowest low in a long trade.

The functions need to recognize when a trade has exited or when a new trade has initiated. There are several options for doing this, which include the use of built-in functions such as MarketPosition and BarsSinceEntry. We opted to use MarketPosition to detect when a trade has exited, so that we can reset the function properly for the next trade. For instance, on the bar that a long trade is exited, TradeStation sets MarketPosition to 0. This is a convenient indication for our function to reset MaxTradeHigh to its initial negative value. However this will not work for trades that reverse long to short or vice versa. In reversal trades, MarketPosition goes from 1 to -1 or from -1 to 1 without going through 0. Using MarketPosition <> 1, as in the code above, solves this problem and correctly detects when a long trade has exited, and using MarketPosition <> -1 correctly detects when a short trade has been exited. These conditions will work for reversal trades as well, and trigger a resetting of the functions whenever trades are exited.

Next, each function compares the current high or low price with the previous value of itself and stores the highest or lowest value as required. In this way the functions keep track of the highest high and lowest low reached in long and short trades respectively.

As a final point of interest, consider this code:

```
If C < C[10] then ExitLong MaxTradeHigh - 2 * AvgTrueRange (20) stop
Else ExitLong Lowest(Low, 20) stop;
```

Programmers will recognize that the function call to MaxTradeHigh is contained within a conditional code, i.e. it is only called when C < C[10]. The immediate concern is that if MaxTradeHigh is not calculated on every bar, its results might be invalid, since the highest high may have occurred on the bar in which the condition C < C[10] was not true. There is however no cause for worry in TradeStation. Since MaxTradeHigh refers to a previous value of itself (i.e. MaxTradeHigh refers to MaxTradeHigh[1]), TradeStation considers it a "series" function, and as such, evaluates it on every bar, regardless of whether it appears within a conditional code block. If you are still concerned about this, simply go to the PowerEditor, open up MaxTradeHigh and MinTradeLow and set their User Function properties to "Series". This will ensure that the functions are executed on every bar.

Now that we have covered the details about coding the functions, the good news is that once the functions are defined, using them is extremely simple. As shown in the examples below, they can be used directly to in the exit code logic. There is no need to worry about proper initialization; the function includes all the code necessary to correctly initialize itself before each trade.

```
If marketposition = 1 then ExitLong MaxTradeHigh - 3 * AvgTrueRange(20) stop;
If marketposition = -1 then ExitLong MinTradeLow + 3 * AvgTrueRange(20) stop;
```

You may wish to go back to our examples and re-write them to track the highest or lowest close reached in a trade. You could use a variable within the main program to keep track of the extreme closed reached, or you may define custom functions to calculate the MaxTradeClose and MinTradeClose similar to the MaxTradeHigh and MinTradeLow functions we have described.

The final code shows an example of using a combination of Yo-Yo and Chandelier in an exit code. Note that although there are multiple exit statements, TradeStation always evaluates and executes the stop that is closest to the trade, i.e. the two exit statements always reduce to 1 stop which is closest to the trade.

```
If marketposition = 1 then begin
    ExitLong MaxTradeHigh - 3 * AvgTrueRange(20) stop;
    ExitLong Close - 1.5 * AvgTrueRange(20) stop;
End;

If marketposition = -1 then begin
    ExitShort MinTradeLow + 3 * AvgTrueRange(20) stop;
    ExitShort Close + 1.5 * AvgTrueRange(20) stop;
End;
```

***Editor's Note:*** *The functions MaxTradeHigh and MinTradeLow as described above will only provide the highest price of a Long position and the lowest price of a Short position. In order to get the lowest price of a Long position or the highest price of a Short position, the following modifications may be made to the functions:*

## Function: MaxTradeHigh

```
Variable: MP(0);

MP = MarketPosition;

If MP <> 0 Then Begin
    If MP <> MP[1] Then
        MaxTradeHigh = High
    Else If High > MaxTradeHigh[1] Then
        MaxTradeHigh = H;
End
Else
   MaxTradeHigh = -999999;
```

## Function: MinTradeLow

```
Variable: MP(0);

MP = MarketPosition;

If MP <> 0 Then Begin
    If MP <> MP[1] Then
        MinTradeLow = Low
    Else If Low < MinTradeLow[1] Then
        MinTradeLow = Low;
End
Else
   MinTradeLow = 999999;
```

*Chuck LeBeau is the co-author (with David Lucas) of the highly regarded text Computer Analysis of the Futures Market. Chuck is also the proprietor of Chuck LeBeau's System Traders Club (www.traderclub.com). You can reach Chuck by e-mail at chuck@tradersclub.com.*

*Terence Tan, who wrote the EasyLanguage code for this article, frequently contributes the ELS code for the systems offered by Chuck LeBeau's System Traders Club.*

CHAPTER 5

# JK Night Train

JK Night Train is the second strategy contributed to STAD Club by Joe Krutsinger, one of Omega's first customers and longtime friends. Night Train is a simple but effective strategy that can be implemented without the necessity of watching the markets during the trading day. Joe designed the strategy for trading the NASDAQ 100 Index, but it can be used to trade other stock indices or even individual stocks.

Night Train's setup for a long position is a close less than the close two bars ago; the short setup is a close greater than or equal to the close two bars ago. The long and short triggers and exits are analogous, so we'll describe only the long side here. The trigger for a long position is a rally to the previous bar's high. The protective stop can be either a Dollar Risk money-management stop or an ATR (Average True Range) Protective Stop. Joe prefers the Dollar Risk Stop, while we have a slight preference for the ATR Protective Stop (which automatically adapts to changes in a market's volatility). Night Train exits trades on the first profitable open (the Bailout technique pioneered by Larry Williams in 1978).

# Defining Our Trading Rules

For the Night Train strategy, we defined long and short setups, triggers, and exits. These components are described next.

## Long and Short Setups

The setup for a long position is a close below the close two bars ago.

The setup for a short position is a close above (or equal to) the close two bars ago.

## Long and Short Triggers

Buy at the high of the previous bar.

Sell short at the low of the previous bar.

## Long and Short Exits

If the open is greater than the entry price, exit your long position.

If the open is less than the entry price, exit your short position.

# Designing & Formatting

This section presents the EasyLanguage instructions and formatting for the strategy, with the EasyLanguage instructions broken down and explained line by line.

## EasyLanguage Strategy Components: JK Night Train (STAD11: JKNightTrain)

### Strategy Inputs (STAD11: JKNightTrain)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| ProtectiveATRs | 3 | The number of Average True Ranges that are risked in the position |
| ATRLength | 10 | Length, expressed in bars, used to calculate the Average True Range |

## Signal Components:

1. JK Night Train

2. ATR Protective Stop

3. Last Bar Exit

## EasyLanguage Signal: JK Night Train:

```
If Close < Close[2] Then
        Buy next bar at High Stop;

If MarketPosition = 1 AND Open of tomorrow > EntryPrice Then
        ExitLong next bar at market;

If Close >= Close[2] Then
        Sell next bar at Low Stop;

If MarketPosition = -1 AND Open of tomorrow < EntryPrice Then
        ExitShort next bar at market;
```

***This Signal does not contain any Inputs or Variables.***

## Long Entry and Exit

Our Buy statement is dependent on the Close value of the current bar being below the Close of two bars ago. If this is the case, we will enter a Long position on the next bar at a price equal to or greater than the current High. Our exit is for the first profitable opening.

```
If Close < Close[2] Then
        Buy next bar at High Stop;

If MarketPosition = 1 AND Open of tomorrow > EntryPrice Then
        ExitLong next bar at market;
```

## Short Entry and Exit

Our Sell statement is dependent on the Close value of the current bar being above or equal to the Close of two bars ago. If this is the case, we will enter a Short position on the next bar at a price equal to or less than the current Low. As in the Long position, our exit is for the first profitable opening.

```
If Close >= Close[2] Then
        Sell next bar at Low Stop;

If MarketPosition = -1 AND Open of tomorrow < EntryPrice Then
        ExitShort next bar at market;
```

## EasyLanguage Signal: ATR Protective Stop

** See Common Stops Appendix

## EasyLanguage Signal: Last Bar Exit

** See Common Stops Appendix

# Testing & Improving

Joe Krutsinger tested the Night Train strategy on both the long and the short sides of the NASDAQ 100 Index (ND) from 4/1996 to 10/1999, deducting $55 per trade for slippage and commission and using a $1,000 Dollar Risk Stop. We tested the strategy on the long side of the S&P 500 Futures Index (SP) from 1/98 to 10/99, deducting $50 per trade for slippage and commission, and placing an ATR Protective Stop at the entry price minus 2.5 times the ten-bar ATR. We also tested the long side of Microsoft (MSFT) from 11/95 to 10/99, trading 100 shares per trade, deducting fifteen cents per share for slippage and commission, and using an ATR (Average True Range) Protective Stop of 6.5 times the ten-bar ATR.

Let's begin with Joe's test of Night Train on the NASDAQ 100 Index. The strategy earned $87,050 on 333 trades, with 49% of the trades profitable and an average trade (wins and losses) of $261 per contract. The average winning trade was 1.6 times as large as the average losing trade, and the strategy earned $1.54 for each $1.00 it lost.

Next, let's see how the strategy fared on the S&P. Figure 1 displays a series of five trades [Figure 1, SP chart]. The highlighted trade shows that the close of bar B was less than the close of bar A (the close two bars ago), that the entry was triggered at the high of bar B (the setup bar), and that the exit (bar D) occurred on the first open that was greater than the entry price. The Performance Summary [Figure 2, SP Performance Summary] tells us that Night Train netted $63,300 on 59 trades, of which 88 % were profitable. The average winning trade was only .26 times as large as the average losing trade, yet the average trade (wins and losses) earned $1,072. The Profit Factor (1.93) indicates that the strategy earned $1.93 for each $1.00 it lost. Figure 3 [Figure 3, SP Equity Curve] shows a strong performance over the first 24 trades, a drawdown between trades 25 and 35, a rally to the highest equity high on trade 43, and a drawdown through the rest of the test period. The last three trades have been profitable, so perhaps the strategy is back on track.

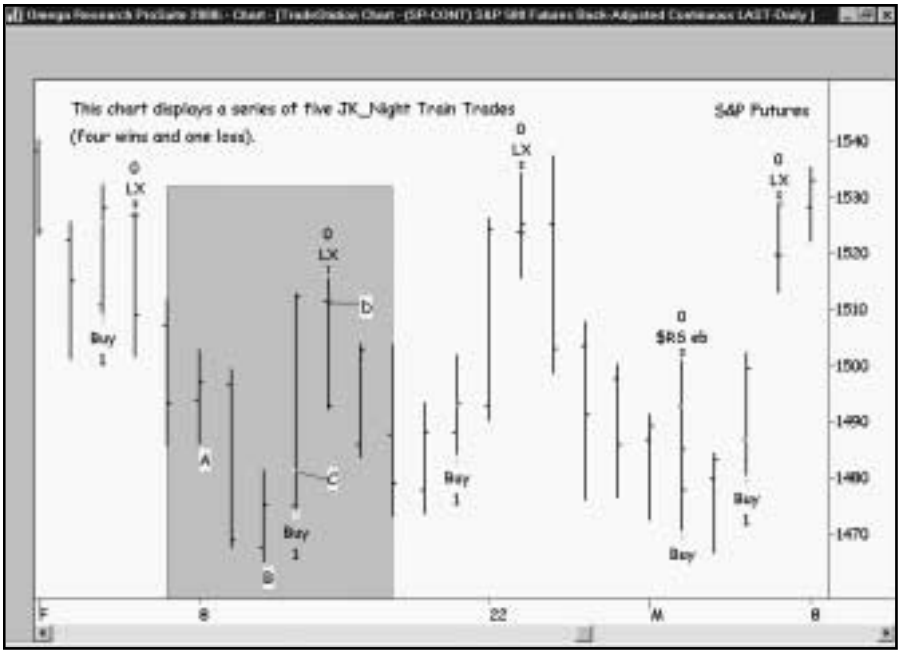



*Figure 1. SP chart*

Finally, let's see Night Train's performance on MSFT. The Performance Summary tells us that the strategy garnered profits of $3,609 (per 100 shares) on 81 trades, of which 64% were profitable [Figure 4, MSFT Performance Summary]. The average winner ($80.23) was 4.14 times greater than the average loser ($19.40). The strategy posted 14 consecutive winners compared to only five consecutive losers. Night Train's strongest number in the MSFT test is the 7.42 Proft Factor-the strategy won $7.42 for each $1.00 it lost.

The Equity Curve [Figure 5, MSFT Equity Curve] shows that profits in MSFT began to accrue around trade 28, increased slowly through trade 53, and skyrocketed for the rest of the test period, which ends with trade 80. Monthly Rolling Net Profit [Figure 6, MSFT MRNP] graphs equity on a marked-to-market basis on the last trading day of each month (marked to market means that all open positions are considered closed out). It's like taking a snapshot of the trading account on the last trading day of each month and charting the result. The graph in this case indicates that our strategy suffered a worse drawdown in mid-1998 (based on MRNP) than it did on the trade-by-trade equity curve (in Figure 6). The Underwater Equity Curve [Figure 7, MSFT Underwater Equity Curve] shows that the mid-1998 drawdown reached only about three percent of the current equity before equity rose to a new high watermark early in 1999.


*Figure 2. SP Performance Summary*

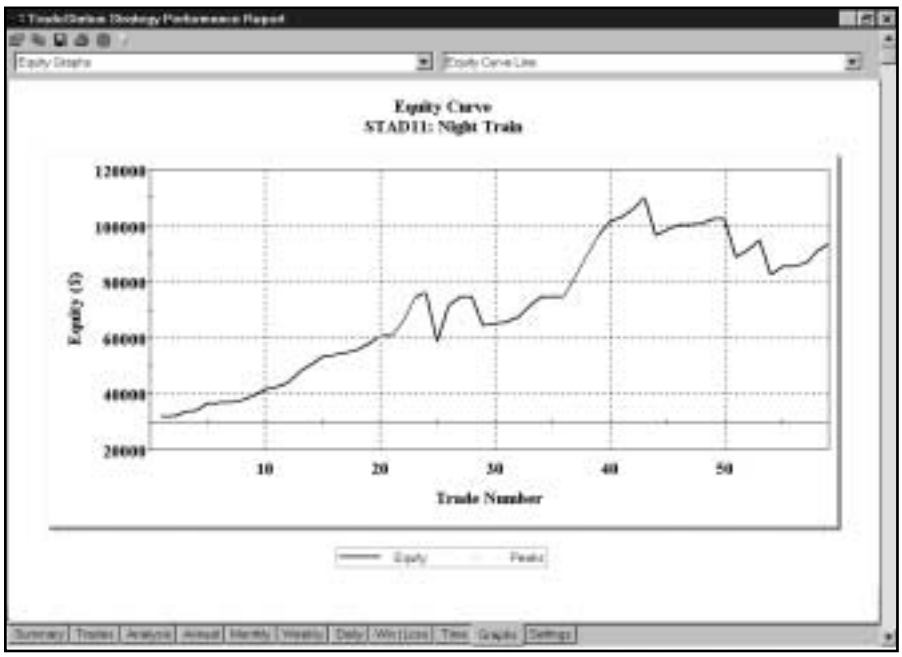



*Figure 3. SP Equity Curve*


*Figure 4. MSFT Performance Summary*

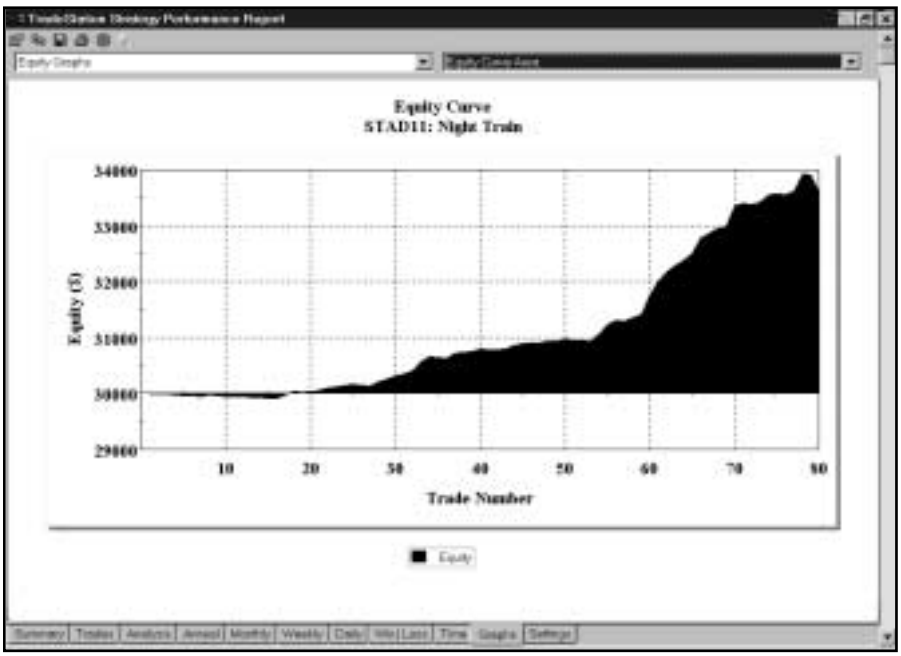



*Figure 5. MSFT Equity Curve*

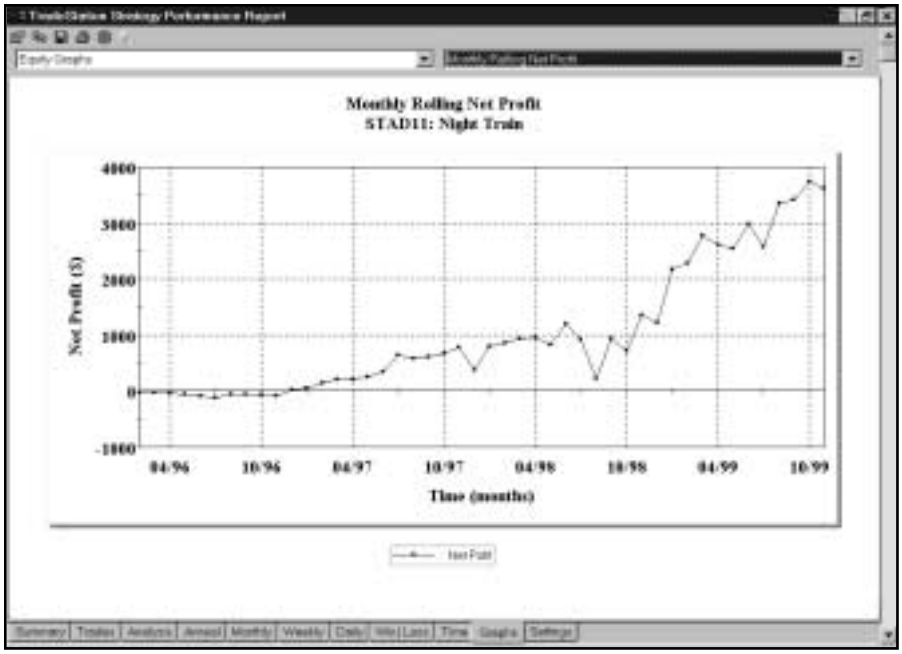



*Figure 6. MSFT MRNP*


*Figure 7. MSFT Underwater Equity Curve*

## Suggestions for Improvement

The beauty of this strategy is its simplicity. Keeping its essentials (setup, trigger, protective stop, and exit) uncomplicated illustrates our belief that a strategy doesn't have to be complex to be valuable. Here's one simple addition, however, that may prove beneficial: calculate a moving average of closing prices and only take the buy setups when the moving average is rising; only take the sell setups when the moving average is declining. Any type of moving average can be considered for this addition (TradeStation 2000i offers several), and a wide range of moving-average lengths can be tested (e.g. 10 - 50 in increments of five). Testing and optimizing are quick and easy-why not give the moving-average addition a try?

STAD Club thanks Joe Krutsinger for contributing his JK Night Train strategy. Joe is the author of two books about strategic trading: *Trading Systems Toolkit* and *Trading Systems: Secrets of the Masters*. (Both books are available from Traders' Press at 800.927.8222 or at www.traderspress.com.) Joe is also the proprietor of System Academy, which offers a new trading system every month. You can contact Joe by e-mail at joekrut@aol.com or by phone at 800.767.2508.

CHAPTER 6

# The Symmetrical Triangle Strategy

Most beginning traders try to pick tops and bottoms — in other words, to buy low and sell high. However, you can only buy low when a market is in a downtrend, and you can only sell high when a market is in an uptrend. If it's true that "the trend is your friend," it might make more sense (especially for inexperienced traders) to buy high and sell higher. A symmetrical triangle is one of many chart formations known as continuation patterns that represent areas of price consolidation within a trending move and that are usually resolved by a breakout in the direction of the preceding trend. When a trader buys a breakout above the top line of a symmetrical triangle formation in an uptrend, he's "buying high" with the expectation that he will be able to "sell higher."

Symmetrical triangles are composed of a descending trendline connecting swing highs and an ascending trendline connecting swing lows (a swing high is higher than the highs to its immediate left and right; a swing low is lower than the lows to its immediate left and right). The swing highs and lows must alternate on the horizontal axis (i.e. point C cannot occur before point B, and point D cannot occur before point C). In the daily bar chart of IBM [Figure 1, IBM symmetrical triangle]), line A-C forms the top of the triangle (resistance), and line B-D forms the bottom of the triangle (support).

Here's how our Symmetrical Triangle Strategy works (we'll discuss only the long side here, as the rules for the short side are just the mirror image — the inverse — of the long side). First, we use the DMI Spread indicator in TradeStation 2000i to find a trending price move (the DMI Spread calculates a market's "trendiness"). When the DMI Spread reaches 15 or higher, the market is currently in a trending mode. Second, we look for a symmetrical triangle (the text that accompanies the EasyLanguage section of this chapter explains how the strategy identifies the triangle). Third, we buy at one point above the resistance line (the top line) of the triangle.

Fourth, we set a protective stop, a breakeven stop, and a profit target. The protective stop is placed at the triangle's second swing low (see Figure 1, point D) minus 50% of the 10-bar Average True Range. The stop is raised to breakeven (our entry price) when the close is greater than or equal to the entry price plus two times the trade's initial risk. For example, if the difference between the entry price and the protective stop is $300, we'll raise the stop to the entry price after a close $600 or more above the entry price. The profit target is determined by subtracting the low at point B from the high at point A and adding that difference to the point at which prices broke above the triangle's resistance line (see Figure 1).


*Figure 1*

Let's summarize: the four steps above identify a trending move, a symmetrical triangle, an entry price, an initial protective stop, a breakeven stop, and a profit target. The strategy's rules are provided below.

## Defining Our Trading Rules

For our Symmetrical Triangle Strategy, we defined long and short setups, triggers, orders, and exits. We also calculated a 14-bar DMI Spread. The setups, triggers, orders, and exits are described next.

### Long and Short Setups

a) A long setup is in effect when the DMI Spread reached 15 or higher at the swing high that formed point 1 of a symmetrical triangle.

b) A short setup is in effect when the DMI Spread reached -15 or lower at the swing low that formed point 1 of a symmetrical triangle.

### Long and Short Triggers

a) The trigger to buy is a breakout above a bullish symmetrical triangle's resistance line.

b) The trigger to sell is a breakout below a bearish symmetrical triangle's support line.

### Long and Short Orders

a) Enter a long position on a stop order placed one point above the triangle's resistance line.

b) Enter a short position on a stop order placed one point below the triangle's support line.

### Long and Short Exits

a) Exit a long position at the protective stop, the breakeven stop, or the profit target.

b) Exit a short position at the protective stop, the breakeven stop, or the profit target.

## Designing & Formatting

This section presents the EasyLanguage instructions and formatting for the strategy, with the EasyLanguage instructions broken down and explained line by line.

### EasyLanguage Strategy Components: Symmetrical Triangles (STAD11: SymTriangles)

#### Strategy Inputs (STAD1: SymTriangles)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| EvaluationRange | 40 | The maximum number of bars that can be elapsed by the Triangle formation |
| DMILength | 14 | The Length parameter used by the DMI functions |
| Strength | 1 | Determines the strength of the swing points |
| ExitATRs | 10 | The number of Average True Ranges to use in the calculations for the initial protective stop |
| ExitValueFactor | .5 | The factor to multiply the Average True Range calculation by, used for the initial protective stop |
| Multiplier | 2 | The factor by which to multiply risk in order to determine the secondary protective stop value |
| Spacing | 1 | The number of bars that must occur between swing points |

## Signal Components:

1. DMILongTriangle

2. DMIShortTriangle

## EasyLanguage Signal: DMILongTriangle:

```
Inputs: EvaluationRange(40), DMILength(14), Strength(1), ExitATRs(10);
Inputs: ExitValueFactor(.50), Multiplier(2), Spacing(1);
Variables: MP(0), DMISpread(0), Increment(0), LongTriangle(False);
Variables: LongFormation(False), TrendlinesDrawn(False), Support(0), Resistance(0);
Variables: BuySetup(False), ProtectiveStopValue(0), ProfitStopValue(0);
Variables: HighSwingPoint(0), HighSwingBar(0), LowSwingPoint(0), LowSwingBar(0);
Array: PivotHigh[2](0), PivotLow[2](0);
Array: PivotHighBar[2](0), PivotLowBar[2](0);

MP = MarketPosition;
DMISpread = DMIPlus(DMILength) - DMIMinus(DMILength);
HighSwingPoint = SwingHigh(1, High, Strength, Strength + 1);
HighSwingBar = SwingHighBar(1, High, Strength, Strength + 1);
LowSwingPoint = SwingLow(1, Low, Strength, Strength + 1);
LowSwingBar = SwingLowBar(1, Low, Strength, Strength + 1);

For Increment = 1 to 2 Begin
        PivotHighBar[Increment] = PivotHighBar[Increment][1] + 1;
        PivotLowBar[Increment] = PivotLowBar[Increment][1] + 1;
End;

If PivotHighBar[2] > EvaluationRange Then Begin
        PivotHigh[2] = PivotHigh[1];
        PivotHigh[1] = 0;
        PivotHighBar[2] = PivotHighBar[1];
End
Else If DMISpread crosses below -15 Then
        PivotHigh[2] = 0;

If HighSwingPoint <> -1 Then Begin
        If HighSwingPoint > PivotHigh[2] Then Begin
                PivotHigh[2] = HighSwingPoint;
                PivotHigh[1] = 0;
                PivotHighBar[2] = HighSwingBar;
        End
        Else If HighSwingPoint > PivotHigh[1] Then Begin
                PivotHigh[1] = HighSwingPoint;
                PivotHighBar[1] = HighSwingBar;
        End
        Else If PivotLowBar[2] < PivotHighBar[1] Then Begin
                PivotHigh[1] = HighSwingPoint;
```

```
                PivotHighBar[1] = HighSwingBar;
        End;
End;

If LowSwingPoint <> -1 Then Begin
        If LowSwingPoint < PivotLow[2] OR PivotHighBar[2] < PivotLowBar[2] Then Begin
                PivotLow[2] = LowSwingPoint;
                PivotLow[1] = 99999;
                PivotLowBar[2] = LowSwingBar;
        End
        Else If LowSwingPoint < PivotLow[1] Then Begin
                PivotLow[1] = LowSwingPoint;
                PivotLowBar[1] = LowSwingBar;
        End
        Else If PivotHighBar[1] < PivotLowBar[1] Then Begin
                PivotLow[1] = LowSwingPoint;
                PivotLowBar[1] = LowSwingBar;
        End;
End;

Condition1 = PivotHigh[2] > PivotHigh[1] AND PivotLow[2] < PivotLow[1];
Condition2 = PivotHighBar[2] > PivotLowBar[2] + Spacing AND
        PivotLowBar[2] > PivotHighBar[1] + Spacing AND
        PivotHighBar[1] > PivotLowBar[1] + Spacing;
Condition3 = PivotHighBar[2] - PivotLowBar[1] <= EvaluationRange;
If PivotHighBar[2] <= EvaluationRange + Strength Then
        Condition4 = DMISpread[PivotHighBar[2]] >= 15
Else
        Condition4 = False;
Condition5 = LowSwingBar <> -1;
Condition6 = TLValue(PivotHigh[2], PivotHighBar[2], PivotHigh[1], PivotHighBar[1], 0) >
        TLValue(PivotLow[2], PivotLowBar[2], PivotLow[1], PivotLowBar[1], 0);

LongTriangle = Condition1 AND Condition2 AND Condition3 AND
        Condition4 AND Condition5 AND Condition6;

If LongTriangle Then Begin
        LongFormation = True;
        Value1 = TL_New(Date[PivotHighBar[2]], Time[PivotHighBar[2]], PivotHigh[2],
                Date[PivotHighBar[1]], Time[PivotHighBar[1]], PivotHigh[1]);
        TL_SetColor(Value1, Cyan);
        Value2 = TL_New(Date[PivotLowBar[2]], Time[PivotLowBar[2]], PivotLow[2],
                Date[PivotLowBar[1]], Time[PivotLowBar[1]], PivotLow[1]);
        TL_SetColor(Value2, Cyan);
        TrendlinesDrawn = True;
        BuySetup = True;
End;

If TrendlinesDrawn Then Begin

        Resistance = TL_GetValue(Value1, Date, Time);
        Support = TL_GetValue(Value2, Date, Time);
```

```
        If Resistance <= Support Then
                BuySetup = False;

        If LongFormation Then Begin
                TL_SetEnd(Value1, Date, Time, Resistance);
                TL_SetEnd(Value2, Date, Time, Support);
                If Low crosses below Support Then
                        BuySetup = False;
                If BuySetup AND MP <> 1 Then
                        Buy Next Bar at Resistance + 1 Point Stop;
        End;
        If Resistance <= Support Then
                LongFormation = False;

        If MP = 1 Then Begin
                BuySetup = False;
                If MP <> MP[1] Then Begin
                        ProfitStopValue = EntryPrice + ( PivotHigh[2] - PivotLow[2] );
                        ProtectiveStopValue = PivotLow[1] - AvgTrueRange(ExitATRs) * ExitValueFactor;
                End;
                If Close >= EntryPrice + Multiplier * (EntryPrice - ProtectiveStopValue) Then
                        ProtectiveStopValue = EntryPrice;
                ExitLong next bar at ProtectiveStopValue Stop;
                ExitLong next bar at ProfitStopValue Limit;
        End;

End;
```

## Signal Inputs (DMILongTriangle)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| EvaluationRange | 40 | The maximum number of bars that can be elapsed by the Triangle formation |
| DMILength | 14 | The Length parameter used by the DMI functions |
| Strength | 1 | Determines the strength of the swing points |
| ExitATRs | 10 | The number of Average True Ranges to use in the calculations for the initial protective stop |
| ExitValueFactor | .5 | The factor to multiply the Average True Range calculation by, used for the initial protective stop |
| Multiplier | 2 | The factor by which to multiply risk in order to determine the secondary protective stop value |
| Spacing | 1 | The number of bars that must occur between swing points |

## Signal Variables (DMILongTriangle)

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| MP | 0 | [Numeric] Holds the value of MarketPosition on the current bar |
| DMISpread | 0 | [Numeric] Stores the value of the spread between the DMI Plus and the DMI Minus |
| Increment | 0 | [Numeric] Used as a reference for the counter variable in each For loop |
| LongTriangle | False | [True/False] Signals the bar when the Long Triangle formation is complete |
| LongFormation | False | [True/False] Used as a trigger to draw trendlines and generate the entry order |
| TrendlinesDrawn | False | [True/False] Used to prevent error codes for the trendline functions by checking that trendlines are drawn before the functions are referenced |
| Support | 0 | [Numeric] Holds the value of the bottom trendline |
| Resistance | 0 | [Numeric] Holds the value of the upper trendline |
| BuySetup | False | [True/False] Used to trigger the Buy order |
| ProtectiveStopValue | 0 | [Numeric] Used to store the value of the protective stop |
| ProfitStopValue | 0 | [Numeric] Used to store the value of the profit stop |
| HighSwingPoint | 0 | [Numeric] Stores the value returned by the function SwingHigh |
| HighSwingBar | 0 | [Numeric] Stores the value returned by the function SwingHighBar |
| LowSwingPoint | 0 | [Numeric] Stores the value returned by the function SwingLow |
| LowSwingBar | 0 | [Numeric] Stores the value returned by the function SwingLowBar |

## Signal Arrays (DMILongTriangle)

| ARRAY | ELEMENTS | DEFAULT | DESCRIPTION |
|---|---|---|---|
| PivotHigh | 2 | 0 | [Numeric] Holds the value of each High pivot point |
| PivotLow | 2 | 0 | [Numeric] Holds the value of each Low pivot point |
| PivotHighBar | 2 | 0 | [Numeric] Holds the number of bars ago each High pivot point occurred |
| PivotLowBar | 2 | 0 | [Numeric] Holds the number of bars ago each Low pivot point occurred |

## Setup

There are a number of calculations that we will need to refer to on an ongoing basis and they are the first items to be addressed. We store MarketPosition and the spread for the DMI into variables, allowing them to be referred to on a bar by bar basis. In addition, we store the SwingHigh, SwingHighBar, SwingLow and SwingLowBar functions into variables for the ease of referring to the variable name later in the signal, rather than writing the function multiple times.

```
MP = MarketPosition;
DMISpread = DMIPlus(DMILength) - DMIMinus(DMILength);
HighSwingPoint = SwingHigh(1, High, Strength, Strength + 1);
HighSwingBar = SwingHighBar(1, High, Strength, Strength + 1);
LowSwingPoint = SwingLow(1, Low, Strength, Strength + 1);
LowSwingBar = SwingLowBar(1, Low, Strength, Strength + 1);
```

The arrays PivotHighBar and PivotLowBar hold the number of bars ago that the far and near pivot points occurred. We increment their values by one on each bar by using the following For loop.

```
For Increment = 1 to 2 Begin
        PivotHighBar[Increment] = PivotHighBar[Increment][1] + 1;
        PivotLowBar[Increment] = PivotLowBar[Increment][1] + 1;
End;
```

If the far PivotHighBar is further out in the past than our requested range, we move the values that have been stored for the near High point into the storage position of the far High point, and reset the near point to 0. If DMISpread falls all the way down to -15, we reset the process developing the formation by resetting the far High point.

```
If PivotHighBar[2] > EvaluationRange Then Begin
        PivotHigh[2] = PivotHigh[1];
        PivotHigh[1] = 0;
        PivotHighBar[2] = PivotHighBar[1];
End
Else If DMISpread crosses below -15 Then
        PivotHigh[2] = 0;
```

If the current bar returns a swing high point, we check for a number of items. First, if the current swing high point is greater than the far High that we have stored, we replace the far High and reset the near High. If the current swing high point is not greater than the far point but is greater than the near point, we replace the near position. Finally, if the new swing point is less than the near High, however the current far Low point is more recent than the stored near High point, we replace the near High in order to get the preferred order of upswings and downswings.

```
If HighSwingPoint <> -1 Then Begin
        If HighSwingPoint > PivotHigh[2] Then Begin
                PivotHigh[2] = HighSwingPoint;
                PivotHigh[1] = 0;
                PivotHighBar[2] = HighSwingBar;
        End
        Else If HighSwingPoint > PivotHigh[1] Then Begin
                PivotHigh[1] = HighSwingPoint;
                PivotHighBar[1] = HighSwingBar;
        End
        Else If PivotLowBar[2] < PivotHighBar[1] Then Begin
                PivotHigh[1] = HighSwingPoint;
                PivotHighBar[1] = HighSwingBar;
        End;
End;
```

In an identical fashion, if the current bar returns a swing low point, we check for similar items. First, if the current swing low point is less than the far Low that we have stored or if the far High is more recent than the far Low, we replace the far Low and reset the near Low. If the current swing low point is greater than the far point but is less than the near point, we replace the near position. Finally, if the new swing point is greater than the near Low, however the near High point is more recent than the stored near Low point, we replace the near Low in order to get the preferred order of upswings and downswings.

```
If LowSwingPoint <> -1 Then Begin
        If LowSwingPoint < PivotLow[2] OR PivotHighBar[2] < PivotLowBar[2] Then Begin
                PivotLow[2] = LowSwingPoint;
                PivotLow[1] = 99999;
                PivotLowBar[2] = LowSwingBar;
        End
        Else If LowSwingPoint < PivotLow[1] Then Begin
                PivotLow[1] = LowSwingPoint;
                PivotLowBar[1] = LowSwingBar;
        End
        Else If PivotHighBar[1] < PivotLowBar[1] Then Begin
                PivotLow[1] = LowSwingPoint;
                PivotLowBar[1] = LowSwingBar;
        End;
End;
```

Once all of the swing points are stored properly in our arrays, we test to see if they are in the formation that yields a triangle. We need descending swing highs, ascending swing lows and the order of the points should be high-low-high-low (with a sufficient amount of spacing). The entire formation should not extend beyond our acceptable range and the DMISpread at the far high point should be above 15. Additionally, the formation should be complete only on the bar that detects the near swing low, and the formation is only valid if the trendlines between the swing points do not already cross each other on that bar. These conditions are outlined below in conditions 1-6. When all of the conditions are true, then LongTriangle is complete.

```
Condition1 = PivotHigh[2] > PivotHigh[1] AND PivotLow[2] < PivotLow[1];
Condition2 = PivotHighBar[2] > PivotLowBar[2] + Spacing AND
        PivotLowBar[2] > PivotHighBar[1] + Spacing AND
        PivotHighBar[1] > PivotLowBar[1] + Spacing;
Condition3 = PivotHighBar[2] - PivotLowBar[1] <= EvaluationRange;
If PivotHighBar[2] <= EvaluationRange + Strength Then
        Condition4 = DMISpread[PivotHighBar[2]] >= 15
Else
        Condition4 = False;
Condition5 = LowSwingBar <> -1;
Condition6 = TLValue(PivotHigh[2], PivotHighBar[2], PivotHigh[1], PivotHighBar[1], 0) >
        TLValue(PivotLow[2], PivotLowBar[2], PivotLow[1], PivotLowBar[1], 0);

LongTriangle = Condition1 AND Condition2 AND Condition3 AND
        Condition4 AND Condition5 AND Condition6;
```

If the LongTriangle is formed, we set LongFormation to True, allowing the extension of the trendlines and the generation of the Buy order. The top and bottom trendlines are drawn and their color is set to Cyan. In order to prevent other functions from returning error codes with the trendlines, TrendlinesDrawn is set to True, indicating that a trendline has now been drawn on the chart, allowing the use of functions that make trendline references. BuySetup is also set to True.

```
If LongTriangle Then Begin
        LongFormation = True;
        Value1 = TL_New(Date[PivotHighBar[2]], Time[PivotHighBar[2]], PivotHigh[2],
                Date[PivotHighBar[1]], Time[PivotHighBar[1]], PivotHigh[1]);
        TL_SetColor(Value1, Cyan);
        Value2 = TL_New(Date[PivotLowBar[2]], Time[PivotLowBar[2]], PivotLow[2],
                Date[PivotLowBar[1]], Time[PivotLowBar[1]], PivotLow[1]);
        TL_SetColor(Value2, Cyan);
        TrendlinesDrawn = True;
        BuySetup = True;
End;

If TrendlinesDrawn Then Begin
```

As a final step in the setup process, the resistance and support values are calculated to determine when the triangle crosses itself. If the apex of the triangle is past on the current bar, the BuySetup is cancelled.

```
Resistance = TL_GetValue(Value1, Date, Time);
Support = TL_GetValue(Value2, Date, Time);

If Resistance <= Support Then
        BuySetup = False;
```

## Long Entry

With LongFormation set to True, the trendlines are extended to the current bar, and if the price falls below the Support value, the BuySetup is cancelled. If the position has not yet entered a Long position and the BuySetup has not been cancelled, we generate an order for the next bar if the price crosses the Resistance level. If the Resistance and Support lines cross, LongFormation is reset to False.

```
If LongFormation Then Begin
        TL_SetEnd(Value1, Date, Time, Resistance);
        TL_SetEnd(Value2, Date, Time, Support);
        If Low crosses below Support Then
                BuySetup = False;
        If BuySetup AND MP <> 1 Then
                Buy Next Bar at Resistance + 1 Point Stop;
End;
If Resistance <= Support Then
        LongFormation = False;
```

## Long Exits

If the market moves into a Long position, MarketPosition will return 1. We turn off BuySetup and on the first bar in the Long direction, the profit stop is calculated as the price of entry plus the difference between the far High and Low points. Our initial protective stop is the near Low point minus the calculated value of the Average True Range multiplied by a factor. If our position hits a close that provides a profit greater than or equal to twice our initial risk, we will raise our protective stop to breakeven. Both the profit and protective stops are generated for the next bar.

```
If MP = 1 Then Begin
        BuySetup = False;
        If MP <> MP[1] Then Begin
                ProfitStopValue = EntryPrice + ( PivotHigh[2] - PivotLow[2] );
                ProtectiveStopValue = PivotLow[1] - AvgTrueRange(ExitATRs) * ExitValueFactor;
        End;
        If Close >= EntryPrice + Multiplier * (EntryPrice - ProtectiveStopValue) Then
                ProtectiveStopValue = EntryPrice;
        ExitLong next bar at ProtectiveStopValue Stop;
        ExitLong next bar at ProfitStopValue Limit;
End;
```

## EasyLanguage Signal: DMIShortTriangle:

```
Inputs: EvaluationRange(40), DMILength(14), Strength(1), ExitATRs(10);
Inputs: ExitValueFactor(.50), Multiplier(2), Spacing(1);
Variables: MP(0), DMISpread(0), Increment(0), ShortTriangle(False);
Variables: ShortFormation(False), TrendlinesDrawn(False), Support(0), Resistance(0);
Variables: SellSetup(False), ProtectiveStopValue(0), ProfitStopValue(0);
Variables: HighSwingPoint(0), HighSwingBar(0), LowSwingPoint(0), LowSwingBar(0);
Array: PivotHigh[2](0), PivotLow[2](0);
Array: PivotHighBar[2](0), PivotLowBar[2](0);

MP = MarketPosition;
DMISpread = DMIPlus(DMILength) - DMIMinus(DMILength);
HighSwingPoint = SwingHigh(1, High, Strength, Strength + 1);
HighSwingBar = SwingHighBar(1, High, Strength, Strength + 1);
LowSwingPoint = SwingLow(1, Low, Strength, Strength + 1);
LowSwingBar = SwingLowBar(1, Low, Strength, Strength + 1);

For Increment = 1 to 2 Begin
        PivotHighBar[Increment] = PivotHighBar[Increment][1] + 1;
        PivotLowBar[Increment] = PivotLowBar[Increment][1] + 1;
End;

If PivotLowBar[2] > EvaluationRange Then Begin
        PivotLow[2] = PivotLow[1];
        PivotLow[1] = 99999;
        PivotLowBar[2] = PivotLowBar[1];
End
Else If DMISpread crosses above 15 Then
        PivotLow[2] = 99999;

If HighSwingPoint <> -1 Then Begin
        If HighSwingPoint > PivotHigh[2] OR PivotLowBar[2] < PivotHighBar[2] Then Begin
                PivotHigh[2] = HighSwingPoint;
                PivotHigh[1] = 0;
                PivotHighBar[2] = HighSwingBar;
        End
        Else If HighSwingPoint > PivotHigh[1] Then Begin
                PivotHigh[1] = HighSwingPoint;
                PivotHighBar[1] = HighSwingBar;
        End
        Else If PivotLowBar[1] < PivotHighBar[1] Then Begin
                PivotHigh[1] = HighSwingPoint;
                PivotHighBar[1] = HighSwingBar;
        End;
End;

If LowSwingPoint <> -1 Then Begin
        If LowSwingPoint < PivotLow[2] Then Begin
                PivotLow[2] = LowSwingPoint;
```

```
                PivotLow[1] = 99999;
                PivotLowBar[2] = LowSwingBar;
        End
        Else If LowSwingPoint < PivotLow[1] Then Begin
                PivotLow[1] = LowSwingPoint;
                PivotLowBar[1] = LowSwingBar;
        End
        Else If PivotHighBar[2] < PivotLowBar[1] Then Begin
                PivotLow[1] = LowSwingPoint;
                PivotLowBar[1] = LowSwingBar;
        End;
End;

Condition1 = PivotHigh[2] > PivotHigh[1] AND PivotLow[2] < PivotLow[1];
Condition2 = PivotLowBar[2] > PivotHighBar[2] + Spacing AND
                PivotHighBar[2] > PivotLowBar[1] + Spacing AND
                PivotLowBar[1] > PivotHighBar[1] + Spacing;
Condition3 = PivotLowBar[2] - PivotHighBar[1] <= EvaluationRange;
If PivotLowBar[2] <= EvaluationRange + Strength Then
        Condition4 = DMISpread[PivotLowBar[2]] <= -15
Else
        Condition4 = False;
Condition5 = HighSwingBar <> -1;

Condition6 = TLValue(PivotHigh[2], PivotHighBar[2], PivotHigh[1], PivotHighBar[1], 0) >
        TLValue(PivotLow[2], PivotLowBar[2], PivotLow[1], PivotLowBar[1], 0);

ShortTriangle = Condition1 AND Condition2 AND Condition3 AND
        Condition4 AND Condition5 AND Condition6;

If ShortTriangle Then Begin
        ShortFormation = True;
        Value1 = TL_New(Date[PivotHighBar[2]], Time[PivotHighBar[2]], PivotHigh[2],
                Date[PivotHighBar[1]], Time[PivotHighBar[1]], PivotHigh[1]);
        TL_SetColor(Value1, Red);
        Value2 = TL_New(Date[PivotLowBar[2]], Time[PivotLowBar[2]], PivotLow[2],
                Date[PivotLowBar[1]], Time[PivotLowBar[1]], PivotLow[1]);
        TL_SetColor(Value2, Red);
        TrendlinesDrawn = True;
        SellSetup = True;
End;

If TrendlinesDrawn Then Begin

        Resistance = TL_GetValue(Value1, Date, Time);
        Support = TL_GetValue(Value2, Date, Time);

        If Resistance <= Support Then
                SellSetup = False;
```

```
        If ShortFormation Then Begin
                TL_SetEnd(Value1, Date, Time, Resistance);
                TL_SetEnd(Value2, Date, Time, Support);
                If High crosses above Resistance Then
                        SellSetup = False;
                If SellSetup AND MP <> -1 Then
                        Sell Next Bar at Support - 1 Point Stop;
        End;
        If Resistance <= Support Then
                ShortFormation = False;

        If MP = -1 Then Begin
                SellSetup = False;
                If MP <> MP[1] Then Begin
                        ProfitStopValue = EntryPrice - ( PivotHigh[2] - PivotLow[2] );
                        ProtectiveStopValue = PivotHigh[1] + AvgTrueRange(ExitATRs) * ExitValueFactor;
                End;
                If Close >= EntryPrice - Multiplier * (EntryPrice - ProtectiveStopValue) Then
                        ProtectiveStopValue = EntryPrice;
                ExitShort next bar at ProtectiveStopValue Stop;
                ExitShort next bar at ProfitStopValue Limit;
        End;

End;
```

## Signal Inputs (DMIShortTriangle)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| EvaluationRange | 40 | The maximum number of bars that can be elapsed by the Triangle formation |
| DMILength | 14 | The Length parameter used by the DMI functions |
| Strength | 1 | Determines the strength of the swing points |
| ExitATRs | 10 | The number of Average True Ranges to use in the calculations for the initial protective stop |
| ExitValueFactor | .5 | The factor to multiply the Average True Range calculation by, used for the initial protective stop |
| Multiplier | 2 | The factor by which to multiply risk in order to determine the secondary protective stop value |
| Spacing | 1 | The number of bars that must occur between swing points |

## Signal Variables (DMIShortTriangle)

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| MP | 0 | [Numeric] Holds the value of MarketPosition on the current bar |
| DMISpread | 0 | [Numeric] Stores the value of the spread between the DMI Plus and the DMI Minus |
| Increment | 0 | [Numeric] Used as a reference for the counter variable in each For loop |
| ShortTriangle | False | [True/False] Signals the bar when the Short Triangle formation is complete |
| ShortFormation | False | [True/False] Used as a trigger to draw trendlines and generate the entry order |
| TrendlinesDrawn | False | [True/False] Used to prevent error codes for the trendline functions by checking that trendlines are drawn before the functions are referenced |
| Support | 0 | [Numeric] Holds the value of the bottom trendline |
| Resistance | 0 | [Numeric] Holds the value of the upper trendline |
| SellSetup | False | [True/False] Used to trigger the Sell order |
| ProtectiveStopValue | 0 | [Numeric] Used to store the value of the protective stop |
| ProfitStopValue | 0 | [Numeric] Used to store the value of the profit stop |
| HighSwingPoint | 0 | [Numeric] Stores the value returned by the function SwingHigh |
| HighSwingBar | 0 | [Numeric] Stores the value returned by the function SwingHighBar |
| LowSwingPoint | 0 | [Numeric] Stores the value returned by the function SwingLow |
| LowSwingBar | 0 | [Numeric] Stores the value returned by the function SwingLowBar |

## Signal Arrays (DMIShortTriangle)

| ARRAY | ELEMENTS | DEFAULT | DESCRIPTION |
|---|---|---|---|
| PivotHigh | 2 | 0 | [Numeric] Holds the value of each High pivot point |
| PivotLow | 2 | 0 | [Numeric] Holds the value of each Low pivot point |
| PivotHighBar | 2 | 0 | [Numeric] Holds the number of bars ago each High pivot point occurred |
| PivotLowBar | 2 | 0 | [Numeric] Holds the number of bars ago each Low pivot point occurred |

## Setup

There are a number of calculations that we will need to refer to on an ongoing basis and they are the first items to be addressed. We store MarketPosition and the spread for the DMI into variables, allowing them to be referred to on a bar by bar basis. In addition, we store the SwingHigh, SwingHighBar, SwingLow and SwingLowBar functions into variables for the ease of referring to the variable name later in the signal, rather than writing the function multiple times.

```
MP = MarketPosition;
DMISpread = DMIPlus(DMILength) - DMIMinus(DMILength);
HighSwingPoint = SwingHigh(1, High, Strength, Strength + 1);
HighSwingBar = SwingHighBar(1, High, Strength, Strength + 1);
LowSwingPoint = SwingLow(1, Low, Strength, Strength + 1);
LowSwingBar = SwingLowBar(1, Low, Strength, Strength + 1);
```

The arrays PivotHighBar and PivotLowBar hold the number of bars ago that the far and near pivot points occurred. We increment their values by one on each bar by using the following For loop.

```
For Increment = 1 to 2 Begin
        PivotHighBar[Increment] = PivotHighBar[Increment][1] + 1;
        PivotLowBar[Increment] = PivotLowBar[Increment][1] + 1;
End;
```

If the far PivotLowBar is further out in the past than our requested range, we move the values that have been stored for the near Low point into the storage position of the far point, and reset the near point to 99999. If DMISpread rises to 15, we reset the process of developing the formation by resetting the far Low point.

```
If PivotLowBar[2] > EvaluationRange Then Begin
        PivotLow[2] = PivotLow[1];
        PivotLow[1] = 99999;
        PivotLowBar[2] = PivotLowBar[1];
End
Else If DMISpread crosses above 15 Then
        PivotLow[2] = 99999;
```

If the current bar returns a swing high point, we check for a number of items. First, if the current swing high point is greater than the far High or if the far Low point is more recent than the far High, we replace the far High and reset the near High. If the current swing high point is not greater than the far point but is greater than the near point, we replace the near position. Finally, if the new swing point is less than the near High, however the near Low point is more recent than the stored near High point, we replace the near High in order to get the preferred order of upswings and downswings.

```
If HighSwingPoint <> -1 Then Begin
        If HighSwingPoint > PivotHigh[2] OR PivotLowBar[2] < PivotHighBar[2] Then Begin
                PivotHigh[2] = HighSwingPoint;
                PivotHigh[1] = 0;
                PivotHighBar[2] = HighSwingBar;
        End
        Else If HighSwingPoint > PivotHigh[1] Then Begin
                PivotHigh[1] = HighSwingPoint;
                PivotHighBar[1] = HighSwingBar;
        End
        Else If PivotLowBar[1] < PivotHighBar[1] Then Begin
                PivotHigh[1] = HighSwingPoint;
                PivotHighBar[1] = HighSwingBar;
        End;
End;
```

In an identical fashion, if the current bar returns a swing low point, we check for similar items. First, if the current swing low point is less than the far Low that we have stored, we replace the far Low and reset the near Low. If the current swing Low point is greater than the far point but is less than the near point, we replace the near position. Finally, if the new swing point is greater than the near Low, however the far High point is more recent than the stored near Low point, we replace the near Low in order to get the preferred order of upswings and downswings.

```
If LowSwingPoint <> -1 Then Begin
        If LowSwingPoint < PivotLow[2] Then Begin
                PivotLow[2] = LowSwingPoint;
                PivotLow[1] = 99999;
                PivotLowBar[2] = LowSwingBar;
        End
        Else If LowSwingPoint < PivotLow[1] Then Begin
                PivotLow[1] = LowSwingPoint;
                PivotLowBar[1] = LowSwingBar;
        End
        Else If PivotHighBar[2] < PivotLowBar[1] Then Begin
                PivotLow[1] = LowSwingPoint;
                PivotLowBar[1] = LowSwingBar;
        End;
End;
```

Once all of the swing points are stored properly in our arrays, we test to see if they are in the formation that yields a triangle. We need descending swing highs, ascending swing lows and the order of the points should be low-high-low-high (with a sufficient amount of spacing). The entire formation should not extend beyond our acceptable range and the DMISpread at the far low point should be below -15. Additionally, the formation should be complete only on the bar that detects the near swing high, and the formation is only valid if the trendlines between the swing points do not already cross each other on that bar. These conditions are outlined below in conditions 1-6. When all of the conditions are true, then ShortTriangle is complete.

```
Condition1 = PivotHigh[2] > PivotHigh[1] AND PivotLow[2] < PivotLow[1];
Condition2 = PivotLowBar[2] > PivotHighBar[2] + Spacing AND
                PivotHighBar[2] > PivotLowBar[1] + Spacing AND
                PivotLowBar[1] > PivotHighBar[1] + Spacing;
Condition3 = PivotLowBar[2] - PivotHighBar[1] <= EvaluationRange;
If PivotLowBar[2] <= EvaluationRange + Strength Then
        Condition4 = DMISpread[PivotLowBar[2]] <= -15
Else
        Condition4 = False;
Condition5 = HighSwingBar <> -1;

Condition6 = TLValue(PivotHigh[2], PivotHighBar[2], PivotHigh[1], PivotHighBar[1], 0) >
        TLValue(PivotLow[2], PivotLowBar[2], PivotLow[1], PivotLowBar[1], 0);

ShortTriangle = Condition1 AND Condition2 AND Condition3 AND
        Condition4 AND Condition5 AND Condition6;
```

If the ShortTriangle is formed, we set ShortFormation to True, allowing the extension of the trendlines and the generation of the Sell order. The top and bottom trendlines are drawn and their color is set to Red. In order to prevent other functions from returning error codes with the trendlines, TrendlinesDrawn is set to True, indicating that a trendline has now been drawn on the chart, allowing the use of functions that make trendline references. SellSetup is also set to True.

```
If ShortTriangle Then Begin
        ShortFormation = True;
        Value1 = TL_New(Date[PivotHighBar[2]], Time[PivotHighBar[2]], PivotHigh[2],
                Date[PivotHighBar[1]], Time[PivotHighBar[1]], PivotHigh[1]);
        TL_SetColor(Value1, Red);
        Value2 = TL_New(Date[PivotLowBar[2]], Time[PivotLowBar[2]], PivotLow[2],
                Date[PivotLowBar[1]], Time[PivotLowBar[1]], PivotLow[1]);
        TL_SetColor(Value2, Red);
        TrendlinesDrawn = True;
        SellSetup = True;
End;

If TrendlinesDrawn Then Begin
```

As a final step in the setup process, the resistance and support values are calculated to determine when the triangle crosses itself. If the apex of the triangle is past on the current bar, the SellSetup is cancelled.

```
Resistance = TL_GetValue(Value1, Date, Time);
Support = TL_GetValue(Value2, Date, Time);

If Resistance <= Support Then
        SellSetup = False;
```

## Short Entry

With ShortFormation set to True, the trendlines are extended to the current bar, and if the price rises above the Resistance value, the SellSetup is cancelled. If the position has not yet entered a Short position and the SellSetup has not been cancelled, we generate an order for the next bar if the price crosses the Support level. If the Resistance and Support lines cross, ShortFormation is reset to False.

```
If ShortFormation Then Begin
        TL_SetEnd(Value1, Date, Time, Resistance);
        TL_SetEnd(Value2, Date, Time, Support);
        If High crosses above Resistance Then
                SellSetup = False;
        If SellSetup AND MP <> -1 Then
                Sell Next Bar at Support - 1 Point Stop;
End;
If Resistance <= Support Then
        ShortFormation = False;
```

## Short Exits

If the market moves into a Short position MarketPosition will return -1. We turn off SellSetup and on the first bar in the Short direction, the profit stop is calculated as the price of entry minus the difference between the far High and Low points. Our initial protective stop is the near high point plus the calculated value of the Average True Range multiplied by a factor. If our position hits a close that provides a profit greater than or equal to twice our initial risk, we will lower our protective stop to breakeven. Both the profit and protective stops are generated for the next bar.

```
If MP = -1 Then Begin
        SellSetup = False;
        If MP <> MP[1] Then Begin
                ProfitStopValue = EntryPrice - ( PivotHigh[2] - PivotLow[2] );
                ProtectiveStopValue = PivotHigh[1] + AvgTrueRange(ExitATRs) * ExitValueFactor;
        End;
        If Close >= EntryPrice - Multiplier * (EntryPrice - ProtectiveStopValue) Then
                ProtectiveStopValue = EntryPrice;
        ExitShort next bar at ProtectiveStopValue Stop;
        ExitShort next bar at ProfitStopValue Limit;
End;
```

## Testing & Improving

We tested our Symmetrical Triangle Strategy on daily data of IBM, Microsoft (MSFT), and Crude Oil (CL) from 5/84 to 11/99. For IBM and MSFT, we bought 100 shares per trade and deducted $.10 per share for slippage and $.05 per share for commission. For CL, we traded one contract and deducted $40 per trade for slippage and $10 per trade for commission.

Let's see how our strategy performed on IBM. The optimized values are as follows:

EvaluationRange = 30

DMILength = 14

Strength = 4

ExitATRs = 10

ExitValueFactor = .10

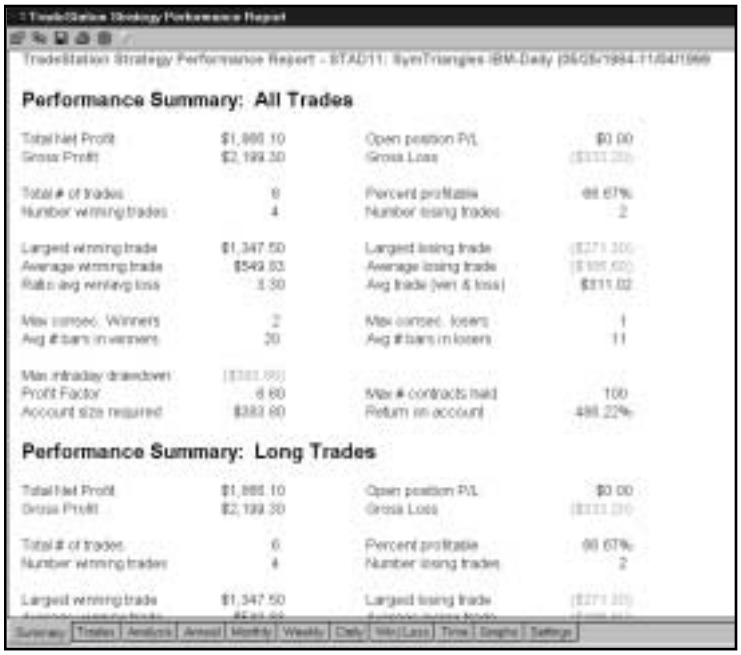
Multiplier = 5

Spacing = 2

The most recent symmetrical triangle trade in IBM is shown in Figure 2 [Figure 2, IBM chart]. Note the "trending" move up to point A that caused the DMI Spread to rise far above the threshold level of 15. Next, note how the high and low swing points alternate on the horizontal axis, i.e. the swing low at B comes after the swing high at A, the swing high at C comes after the swing low at B, etc. We bought at E on the breakout above the triangle's resistance line, and we exited at the entry price plus the height of the triangle from point A to point B.


*Figure 2. IBM chart*

Applied to IBM, our Symmetrical Triangle Strategy generated profits of $1,866 (per 100 shares) on six trades. Sixty-six percent of the trades were profitable, and the average winner ($549) was 3.3 times as large as the average loser ($166). The average trade (wins & losses) earned $311 [Figure 3, IBM Performance Summary]. The trade-by-trade results are provided in Figure 4 [Figure 4, IBM Trades].


TradeStation Strategy Performance Report

TradeStation Strategy Performance Report - STAD11: SymTriangles IBM-Daily (05/25/1984-11/04/1999

**Performance Summary: All Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $1,866.10 | Open position P/L | $0.00 |
| Gross Profit | $2,199.30 | Gross Loss | ($333.20) |
| Total # of trades | 6 | Percent profitable | 66.67% |
| Number winning trades | 4 | Number losing trades | 2 |
| Largest winning trade | $1,347.50 | Largest losing trade | ($271.30) |
| Average winning trade | $549.83 | Average losing trade | ($166.60) |
| Ratio avg win/avg loss | 3.30 | Avg trade (win & loss) | $311.02 |
| Max consec. Winners | 2 | Max consec. losers | 1 |
| Avg # bars in winners | 20 | Avg # bars in losers | 11 |
| Max intraday drawdown | ($383.90) | | |
| Profit Factor | 6.60 | Max # contracts held | 100 |
| Account size required | $383.90 | Return on account | 486.22% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $1,866.10 | Open position P/L | $0.00 |
| Gross Profit | $2,199.30 | Gross Loss | ($333.20) |
| Total # of trades | 6 | Percent profitable | 66.67% |
| Number winning trades | 4 | Number losing trades | 2 |
| Largest winning trade | $1,347.50 | Largest losing trade | ($271.30) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win/Loss | Time | Graphs | Settings

*Figure 3. IBM Performance Summary*

TradeStation Strategy Performance Report

Std.

| Trade # | Date | Type | Cnts | Price | Signal Name | Entry P/L | Cumulative |
|---|---|---|---|---|---|---|---|
| 1 | 09/11/1985 | Buy | 100 | 31.97 | Buy | | |
| | 09/25/1985 | LExit | 100 | 31.50 | LX | (61.90) | (61.90) |
| 2 | 06/17/1987 | Buy | 100 | 40.56 | Buy | | |
| | 08/14/1987 | LExit | 100 | 43.75 | LX#2 | 303.70 | 241.80 |
| 3 | 04/18/1990 | Buy | 100 | 27.72 | Buy | | |
| | 05/21/1990 | LExit | 100 | 29.25 | LX#2 | 138.10 | 379.90 |
| 4 | 01/09/1997 | Buy | 100 | 39.94 | Buy | | |
| | 01/24/1997 | LExit | 100 | 37.38 | LX | (271.30) | 108.60 |
| 5 | 12/22/1998 | Buy | 100 | 88.75 | Buy | | |
| | 12/24/1998 | LExit | 100 | 93.00 | LX#2 | 410.00 | 518.60 |
| 6 | 06/18/1999 | Buy | 100 | 121.13 | Buy | | |
| | 07/08/1999 | LExit | 100 | 134.75 | LX#2 | 1347.50 | 1866.10 |

*Figure 4. IBM Trades*

Next, let's take a look at the Symmetrical Triangle Strategy in MSFT.

The optimized values are as follows:

EvaluationRange = 20

DMILength = 14

Strength = 3

ExitATRs = 10

ExitValueFactor = .50

Multiplier = 1

Spacing = 1

Figure 5 is an example of a triangle trade in MSFT [Figure 5, MSFT chart]. Applied to MSFT, our strategy netted $720 per trade (per 100 shares) on seven trades [Figure 6, MSFT Performance Summary]. Only one of the seven trades lost money (-$52), and the average win ($128) was 2.45 times as large as the one loss. The strategy let profits run by staying in winning trades for an average of 13 bars, while it cut losses short by exiting the losing trade in only three bars. The Profit Factor (14.71) is extremely good: the strategy won $14.71 for each $1.00 it lost. The trade-by-trade results are provided in Figure 7 [Figure 7, MSFT Trades].


*Figure 5. MSFT chart*

TradeStation Strategy Performance Report – STA011: SymTriangles MSFT-Daily

**Performance Summary: All Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $720.00 | Open position P/L | $0.00 |
| Gross Profit | $772.50 | Gross Loss | ($52.50) |
| Total # of trades | 7 | Percent profitable | 85.71% |
| Number winning trades | 6 | Number losing trades | 1 |
| Largest winning trade | $435.00 | Largest losing trade | ($52.50) |
| Average winning trade | $128.75 | Average losing trade | ($52.50) |
| Ratio avg win/avg loss | 2.45 | Avg trade (win & loss) | $102.86 |
| Max consec. Winners | 4 | Max consec. losers | 1 |
| Avg # bars in winners | 13 | Avg # bars in losers | 3 |
| Max intraday drawdown | ($88.90) | | |
| Profit Factor | 14.71 | Max # contracts held | 100 |
| Account size required | $88.90 | Return on account | 767.59% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $720.00 | Open position P/L | $0.00 |
| Gross Profit | $772.50 | Gross Loss | ($52.50) |
| Total # of trades | 7 | Percent profitable | 85.71% |
| Number winning trades | 6 | Number losing trades | 1 |
| Largest winning trade | $435.00 | Largest losing trade | ($52.50) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win/Loss | Time | Graphs | Settings

*Figure 6. MSFT Performance Summary*

TradeStation Strategy Performance Report

| Trade # | Date | Type | Cnts | Price | Signal Name | Entry P/L | Cumulative |
|---|---|---|---|---|---|---|---|
| 1 | 01/04/1991 | Buy | 100 | 2.13 | Buy | | |
| | 01/17/1991 | LExit | 100 | 2.38 | LX#2 | 10.00 | 10.00 |
| 2 | 09/24/1991 | Buy | 100 | 3.63 | Buy | | |
| | 11/08/1991 | LExit | 100 | 4.00 | LX#2 | 22.50 | 32.50 |
| 3 | 09/16/1994 | Buy | 100 | 7.25 | Buy | | |
| | 09/21/1994 | LExit | 100 | 6.88 | LX | (52.50) | (20.00) |
| 4 | 04/17/1996 | Buy | 100 | 13.16 | Buy | | |
| | 05/01/1996 | LExit | 100 | 14.25 | LX#2 | 94.40 | 74.40 |
| 5 | 10/15/1996 | Buy | 100 | 17.34 | Buy | | |
| | 11/14/1996 | LExit | 100 | 18.38 | LX#2 | 88.10 | 162.50 |
| 6 | 01/16/1997 | Buy | 100 | 21.25 | Buy | | |
| | 01/20/1997 | LExit | 100 | 22.63 | LX#2 | 122.50 | 285.00 |
| 7 | 03/23/1998 | Buy | 100 | 40.75 | Buy | | |
| | 03/25/1998 | LExit | 100 | 45.25 | LX#2 | 435.00 | 720.00 |

*Figure 7. MSFT Trades*

Finally, let's find out how the Symmetrical Triangle Strategy performed on Crude Oil. The optimized values are as follows:

EvaluationRange = 40

DMILength = 14

Strength = 4

ExitATRs = 10

ExitValueFactor = .25

Multiplier = 3

Spacing = 3

Our strategy made $9,150 on 10 trades in CL [Figure 8, CL Performance Summary]. Fifty percent of the trades were profitable, and the average winner ($2,642) was 3.25 times as large as the average loser ($812). The largest winning trade ($4,010) far surpassed the largest losing trade ($1,310). The average trade (wins & losses) earned an impressive $915 per contract.


TradeStation Strategy Performance Report – 8TAD11: SymTriangles CL-CONT-Daily

**Performance Summary: All Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $9,150.00 | Open position P/L | $0.00 |
| Gross Profit | $13,210.00 | Gross Loss | ($4,060.00) |
| Total # of trades | 10 | Percent profitable | 50.00% |
| Number winning trades | 5 | Number losing trades | 5 |
| Largest winning trade | $4,010.00 | Largest losing trade | ($1,310.00) |
| Average winning trade | $2,642.00 | Average losing trade | ($812.00) |
| Ratio: avg winning loss | 3.25 | Avg trade (win & loss) | $915.00 |
| Max consec. Winners | 2 | Max consec. losers | 3 |
| Avg # bars in winners | 20 | Avg # bars in losers | 12 |
| Max intraday drawdown | ($2,210.00) | | |
| Profit Factor | 3.25 | Max # contracts held | 1 |
| Account size required | $5,210.00 | Return on account | 175.62% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $8,700.00 | Open position P/L | $0.00 |
| Gross Profit | $11,300.00 | Gross Loss | ($2,600.00) |
| Total # of trades | 7 | Percent profitable | 57.14% |
| Number winning trades | 4 | Number losing trades | 3 |
| Largest winning trade | $4,010.00 | Largest losing trade | ($1,310.00) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win / Loss | Time | Graphs | Settings

*Figure 8. CL Performance Summary*

## Suggestions For Improvement

We think we've identified symmetrical triangles as well as we can. Also, we're satisfied with this strategy's setups, triggers and orders. Perhaps we can improve the strategy's exits. In its present form, the Symmetrical Triangle Strategy moves its initial protective stop to breakeven and then takes profits at a fixed target. It's possible that a series of stops similar to those used in this volume's MISER strategy could do a better job of managing the Symmetrical Triangle Strategy's exits.

CHAPTER 7

# Making the Most of Your Strategy Performance Reports

TradeStation's Strategy Performance Report provides you with in-depth information about how well or how poorly your trading strategies performed on historical data [Figure 1, sample Performance Summary]. The evaluation of a strategy should encompass many important factors in addition to the obvious Net Profit or Net Loss. Here's a brief rundown on several of the most important items featured in the Strategy Performance Report.

***Total Net Profit:*** Don't rely solely on this number; it's important, but it doesn't tell the whole story. *How* a strategy earned the profit is crucial. For example, if a strategy with an acceptable Total Net Profit had 15% winners and a 70% drawdown before hitting a grandslam homerun or two, the strategy probably wouldn't have been tradeable in the real world.

***Open Position P/L:*** Include the "LastBarExit" signal (which you'll find in StrategyBuilder) so performance numbers aren't skewed by a large open profit or loss. This signal will automatically exit positions on the last bar of your test data.

***Total # of Trades:*** Every trade incurs a commission, and most trades (except for limit orders) suffer slippage (the difference between the price you want and the price you get when your order is filled). Many strategies are not powerful enough to overcome the costs of making a large number of trades. Emphasize the quality of trades, not the quantity of trades.

***Percent Profitable:*** This number is closely linked to the Ratio Avg Win/Avg Loss. All traders would like to have 90% winners with a 10-to-1 reward-to-risk ratio, but that is not attainable (yet). A trader can achieve a high winning percentage with a low reward-to-risk ratio, a low winning percentage with a high reward-to-risk ratio, or an average winning percentage with an average reward-to-risk ratio. A good, middle-of-the-road strategy would have 40 - 50% winners with a ratio of average win to average loss of 3 or 3.5 to 1.


TradeStation Strategy Performance Report - C:\Osa\ MSFT-Daily

**Performance Summary: All Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $4,473.40 | Open position P/L | $0.00 |
| Gross Profit | $7,332.60 | Gross Loss | ($2,859.40) |
| Total # of trades | 35 | Percent profitable | 42.86% |
| Number winning trades | 15 | Number losing trades | 20 |
| Largest winning trade | $2,772.50 | Largest losing trade | ($1,059.70) |
| Average winning trade | $488.85 | Average losing trade | ($142.97) |
| Ratio avg win/avg loss | 3.42 | Avg trade (win & loss) | $127.81 |
| Max consec. Winners | 6 | Max consec. losers | 8 |
| Avg # bars in winners | 54 | Avg # bars in losers | 14 |
| Max intraday drawdown | ($1,348.70) | | |
| Profit Factor | 2.56 | Max # contracts held | 100 |
| Account size required | $1,348.70 | Return on account | 331.69% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $4,473.40 | Open position P/L | $0.00 |
| Gross Profit | $7,332.60 | Gross Loss | ($2,859.40) |
| Total # of trades | 35 | Percent profitable | 42.86% |
| Number winning trades | 15 | Number losing trades | 20 |
| Largest winning trade | $2,772.50 | Largest losing trade | ($1,059.70) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win/Loss | Time | Graphs | Settings

*Figure 1*

***Largest Winning Trade:*** This result is more important in a long-term trendfollowing strategy than in a countertrend or event-based strategy (trendfollowing strategies buy high and try to sell higher, countertrend strategies buy low and try to sell high, and event-based strategies buy or sell a condition, pattern, etc. without regard to its location within a trend). Trendfollowing depends on a few "outlier" winning trades (trades more than three standard deviations greater than the average trade) to compensate for a large number of small losses, while the other strategies strive for consistency (lots of small profits vs. fewer losses). Even in a trendfollowing strategy, the largest winning trade shouldn't account for too large a percentage of a strategy's Total Net Profit (more than 40 - 45%): what if that one incredible trade hadn't occurred, or if (for whatever reason) the trader missed that trade? There's also no guarantee that a similar opportunity for a windfall profit will occur again in the reasonably near future for the same strategy in the same market. A good strategy maintains acceptable numbers even when the largest winning trade is deleted from the Performance Summary.

***Largest Losing Trade:*** For professionals, the Largest Losing Trade lost so much because of a gap, a limit move, excessive slippage, or another event beyond the trader's immediate control. Skilled traders do not accept large losing trades as normal occurrences in their strategies. In many profitable strategies, the small wins and small losses approximately cancel each other out, there are no (or very few) big losses, and the Total Net Profit approximates the dollars gained on the big wins.

***Ratio Avg Win/Avg Loss:*** The desired ratio of the average win to the average loss depends on the type of strategy followed. A profitable trendfollowing strategy generally achieves a ratio of at least 3 or 4 to 1, while countertrend and event-based strategies can often make money with ratios of 1 to 1 or even less (with a very high winning percentage).

***Avg Trade (win & loss):*** The dollar amount won or lost on the average trade may be the most important of the performance measurements. What's the expectation in dollars when a trader makes a trade? It must, of course, be a positive number after deductions are made for slippage and commissions. In addition, traders should consider the other costs of trading (hardware, software, data, books, seminars, etc). The average trade must also earn enough money to compensate the trader adequately for the time he or she devotes to trading, the stress that the trader endures, and the financial risks the trader assumes.

***Max Consecutive Winners/Max Consecutive Losers:*** These numbers are not meaningful in isolation from other performance measurements. For example, a strategy that won eight times in a row over the course of 100 trades may not be as profitable as a strategy that only won three times in a row but that had a higher winning percent and/or a higher reward-to-risk ratio. A trendfollowing strategy generally has a greater number of consecutive losers than consecutive winners, while countertrend and event-based strategies usually have a greater number of consecutive winners than consecutive losers. It's not uncommon for a profitable trendfollowing system to suffer ten consecutive small losses, and it's not too unusual for a countertrend or event-based strategy to post ten consecutive small wins.

***Avg # Bars in Winners/Avg # Bars in Losers:*** "Let profits run and cut losses short" is one of the doctrines practiced by successful traders. Most profitable trendfollowing strategies hold winning trades at least three times as long as losing trades; most profitable countertrend and event-based strategies keep winners at least as long as they keep losers.

***Max Intraday Drawdown:*** This statistic represents the largest peak-to-valley decline in equity experienced over the course of testing and/or trading a strategy. Drawdown is one of the most important factors in evaluating a strategy. Most professional traders would prefer a strategy that earned a 35% annual return with a 15% drawdown to a strategy that earned 45% with a 25% drawdown. Although the amount of drawdown that can be tolerated varies among individuals, a strategy that suffers a drawdown greater than 30% in back-testing or actual trading should be re-evaluated and modified to reduce the drawdown. Keep in mind that a 50% drawdown requires a 100% return on the remaining equity just to get back to breakeven!

***Profit Factor:*** Profit Factor, which is calculated by dividing Gross Profit by Gross Loss, represents the number of dollars won for each dollar lost. Obviously, the minimum requirement for a strategy's Profit Factor is that it's a positive number. A guideline to consider is that a robust strategy should generate a Profit Factor of at least 2.0 ($2 won for each $1 lost).

***Account Size Required:*** Account Size Required applies only to strategies that trade futures. In the Performance Report, Account Size Required is calculated by multiplying the margin required to trade one futures contract by the maximum number of contracts held, and adding the result to the Max Intraday Drawdown. Account Size Required is not a recommendation that a strategy be traded with only the required amount of capital behind it. Ideally, an account should be funded with at least two or three times the amount of money specified in Account Size Required to protect against worse-than-anticipated drawdowns.

***Return on Account:*** This number is calculated by dividing Total Net Profit by Account Size Required. The percent return must be interpreted correctly, or the trader will have an inflated idea of the return he or she should expect. Remember that Account Size Required is the minimum amount necessary for trading a strategy, not an ideal amount. Also, the Return on Account represents the percent return over the entire test period-not the percent return on an annual basis.

## In Conclusion

TradeStation's most valuable feature is that it allows you to back-test your trading ideas to discover exactly how your idea performed historically before you risk any money in the markets. We hope these brief notes and guidelines will help you to make the best possible use of your Strategy Performance Reports.

CHAPTER 8

# TrendScore

The TrendScore strategy is based on an idea by Tushar Chande, the author of the well-respected book *Beyond Technical Analysis*. For STAD Club, we added our ATRProtectiveStop, ATRBreakevenStop, and ATRTrailingStop. We also optimized the lookback period and the moving-average length. The result is a promising trendfollowing strategy that we tested on both stocks and futures.

To calculate the TrendScore, Chande compares today's close to a series of closes in the past. If today's close is greater than or equal to the close n bars ago, score 1; if today's close is less than the close n bars ago, score -1. Continue this process through 10 comparisons. If today's close is greater than or equal to all 10 of the previous closes, TrendScore will be +10; if today's close is less than all 10 previous closes, TrendScore will be -10. Begin the comparisons 11 bars back from today's close and proceed through the bar that occurred 20 bars ago. In other words, compare today's close to the closes 11 through 20 bars ago.

Next, calculate an 18-day simple moving average of the TrendScore and an 18-day simple moving average of the closes. When the TrendScore is greater than the moving average of the TrendScore, and the close is greater than the moving average of the closes, buy on the next open. Exit your long position when either the TrendScore is less than its moving average, or the close is less than its moving average. Similarly, when the TrendScore is less than the moving average of the TrendScore, and the close is less than the moving average of closes, sell short on the next open. Exit your short position when either the TrendScore is greater than its moving average, or the close is greater than its moving average. Other exits will be signaled by the ATRProtectiveStop, the ATRBreakevenStop, and the ATRTrailingStop.

## Defining Our Trading Rules

For the TrendScore strategy, we defined both long and short setups, orders, and exits. We did not define long or short triggers; instead, the strategy currently enters at the market on the next open after the setup is in place. We calculated TrendScore and its simple moving average, a simple moving average of closes, and a 10-bar ATR (Average True Range). The setups, orders, and exits are described next.

### Long and Short Setups

a) When TrendScore is above its moving average, and the close is above its moving average, you have a setup to buy.

b) When TrendScore is below its moving averge, and the close is below its moving average, you have a setup to sell short.

### Long and Short Orders

a) Buy at the market on the open after a long setup has occurred.

b) Sell short at the market on the next open after a short setup has occurred.

### Long and Short Exits

a) Exit a long position when TrendScore is below its moving average or the close is below its moving average. Also exit a long position at the ATRProtectiveStop, the ATRBreakevenStop, or the ATRTrailingStop.

b) Exit a short position when TrendScore is above its moving average or the close is above its moving average. Also exit a short position at the ATRProtectiveStop, the ATRBreakevenStop, or the ATRTrailingStop.

## Designing & Formatting

This section presents the EasyLanguage instructions and formatting for the strategy, with the EasyLanguage instructions broken down and explained line by line.

## EasyLanguage Strategy Components: TrendScore (STAD11: TrendScore)

### Strategy Inputs (STAD11: TrendScore)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| BreakevenATRs | 4 | The floor value, the number of Average True Ranges above/below the Entry Price at which the Stop becomes active for the position |
| ATRLength | 10 | Length, expressed in bars, used to calculate the Average True Range |
| LookBack | 10 | The number of historical bars to skip before calculating the TrendScore |
| Length | 18 | The Length used to calculate the average of both the Close price and the TrendScore |
| TrailingATRs | 4 | The number of Average True Ranges that are risked from the highest/lowest price of the position |
| ProtectiveATRs | 3 | The number of Average True Ranges that are risked in the position |

### Signal Components:

1. TrendScore
2. ATR Breakeven Stop
3. ATR Protective Stop
4. ATR Trailing Stop

## EasyLanguage Signal: TrendScore:

```
Inputs: LookBack(10), Length(18);
Variables: TrendScore(0), TSA(0), SMA(0);

TrendScore = 0;

For Value1 = 1 to 10 Begin
        If Momentum(Close, LookBack + Value1) >= 0 Then
                TrendScore = TrendScore + 1
        Else
                TrendScore = TrendScore - 1;
End;
```

```
TSA = Average(TrendScore, Length);
SMA = Average(Close, Length);

If TrendScore > TSA AND Close > SMA Then
        Buy Next Bar at Market
Else If MarketPosition = 1 Then
        ExitLong Next Bar at Market;

If TrendScore < TSA AND Close < SMA Then
        Sell Next Bar at Market
Else If MarketPosition = -1 Then
        ExitShort Next Bar at Market;
```

## Signal Inputs (TrendScore)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| LookBack | 10 | The number of historical bars to skip before calculating the TrendScore |
| Length | 18 | The Length used to calculate the average of both the Close price and the TrendScore |

## Signal Variables (TrendScore)

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| TrendScore | 0 | [Numeric] Holds the value of the TrendScore calculated on the current bar |
| TSA | 0 | [Numeric] Holds the value of the average TrendScore |
| SMA | 0 | [Numeric] Holds the value of the average Close price |

## Setup

By the Strategy definition, TrendScore is the difference between the number of closes starting "LookBack" bars ago that are less than the current close, minus the number of closes that are greater than the current close, evaluated over a range of ten bars. This number can be calculated by using a For loop and the Momentum function. The Momentum function tests whether the Close price of "x" bars ago is greater or less than the current Close. If greater, we add one to the TrendScore, if less we subtract one. The use of the For loop allows us to check each of the bars within the specified range, and tally the TrendScore. The TrendScore variable is reset to 0 at the beginning of each bar, permitting a new calculation to be generated from scratch.

```
TrendScore = 0;

For Value1 = 1 to 10 Begin
        If Momentum(Close, LookBack + Value1) >= 0 Then
                TrendScore = TrendScore + 1
        Else
                TrendScore = TrendScore - 1;
End;
```

The average of TrendScore and of the Close price is calculated and stored in the variables TSA (TrendScore Average) and SMA (Simple Moving Average) respectively, for use in the Long and Short Entry criteria.

```
TSA = Average(TrendScore, Length);
SMA = Average(Close, Length);
```

### Long Entry and Exit

If both the TrendScore and Close values are above their respective moving averages, a Buy order for the next bar is generated. If either value falls below its moving average and a Long position currently exists, the ExitLong order is triggered for the next bar. The function MarketPosition returns 1 to indicate a Long position and is used to generate the exit order.

```
If TrendScore > TSA AND Close > SMA Then
        Buy Next Bar at Market
Else If MarketPosition = 1 Then
        ExitLong Next Bar at Market;
```

### Short Entry and Exit

If both the TrendScore and Close values are below their respective moving averages, a Sell order for the next bar is generated. If either value rises above its moving average and a Short position currently exists, the ExitShort order is triggered for the next bar. The function MarketPosition returns -1 to indicate a Short position and is used to generate the exit order.

```
If TrendScore < TSA AND Close < SMA Then
        Sell Next Bar at Market
Else If MarketPosition = -1 Then
        ExitShort Next Bar at Market;
```

## EasyLanguage Signal: ATR Breakeven Stop

** See Common Stops Appendix

## EasyLanguage Signal: ATR Protective Stop

** See Common Stops Appendix

## EasyLanguage Signal: ATR Trailing Stop

** See Common Stops Appendix

## Testing & Improving

We tested the TrendScore strategy on daily bars of Adobe (ADBE), American Express (AXP), and Japanese Yen futures (JY) from 2/92 - 10/99. For ADBE and AXP, we deducted $.10 per share for slippage and $.05 per share for commission. For JY. We deducted $40 per contract for slippage and $10 per contract for commission.

Let's begin our survey of TrendScore's test results with ADBE. The optimized values are as follows:

ATRProtectiveStop = 3 (We used a 10-bar ATR for all the stops)

ATRBreakevenStop = 5

ATRTrailingStop = 6

TrendScore Lookback Length = 30

TrendScore Length = 40

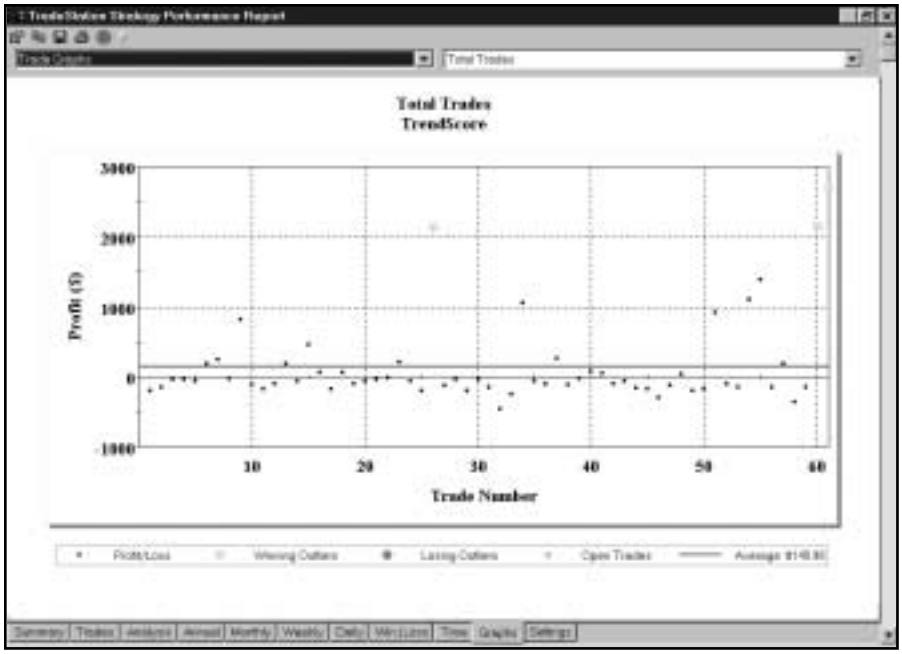
TrendScore's two most recent trades in ADBE are shown in Figure 1 [Figure 1, ADBE chart]. Both the entries and the exits were very timely. Applied to ADBE, TrendScore produced a net profit of $9,148 on 61 trades. Only 32.8% of the trades were profitable, but the average winner was 5.7 times as large as the average loser, resulting in an average trade of $149. The strategy earned $2.78 for each $1.00 it lost [Figure 2, ADBE Performance Summary.


*Figure 1. ADBE chart*


*Figure 2. ADBE Performance Summary*

Our strategy's Monthly Rolling Net Profit shows profitable results after the first few trades and a dramatic increase in profitability in the last few years [Figure 3, ADBE MRNP]. The graph of Total Trades indicates that TrendScore achieved three positive outliers-trades more than three standard deviations greater than the average trade-but suffered no negative outliers [Figure 4, ADBE Total Trades].


*Figure 3. ADBE MNRP*




*Figyre 4. ADBE Total Trades*

Next, let's take a look at TrendScore and AXP. The optimized values are as follows:

ATRProtectiveStop = 3 (We used a 10-bar ATR for all the stops)

ATRBreakevenStop = 3

ATRTrailingStop = 5

TrendScore Lookback Length = 30

TrendScore Length = 30

Applied to AXP, TrendScore earned $5,924 per 100 shares traded [Figure 5, AXP Performance Summary]. As was the case with ADBE, the winning percent was low (32.9%), but the ratio of average win to average loss was high (4.6), resulting in positive results. The strategy let profits run by holding winning trades for an average of 21 bars, while it cut losses short by exiting losers in an average of only five bars. TrendScore won $2.25 for each $1.00 it lost. The Annual Trading Summary shows that the strategy made money in seven of the eight years tested, losing only in 1994 [Figure 6, AXP Annual Trading Summary].

Finally, let's see how well TrendScore performed on JY. The optimized values are as follows:

ATRProtectiveStop = 3 (We used a 10-bar ATR for all the stops)

ATRBreakevenStop = 4

ATRTrailingStop = 5

TrendScore Lookback Length = 60

TrendScore Length = 80


TradeStation Strategy Performance Report – TrendScore AXP.Daily

**Performance Summary: All Trades**

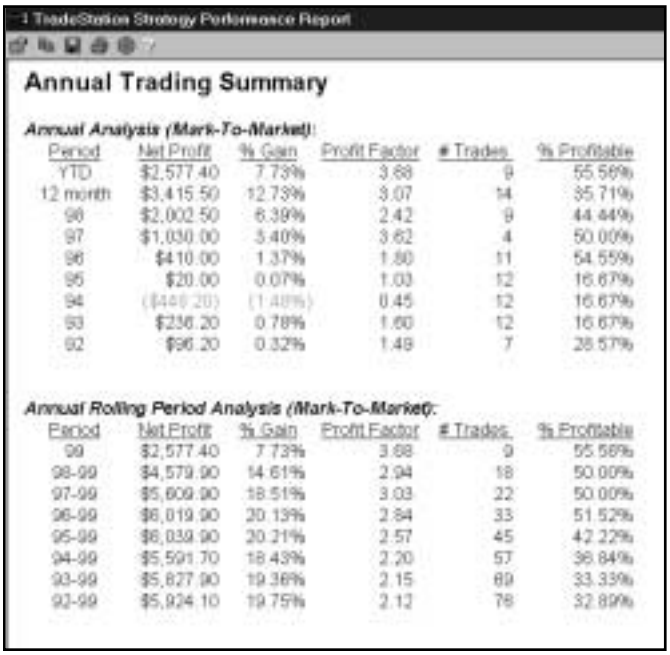
| | | | |
|---|---|---|---|
| Total Net Profit | $5,924.10 | Open position P/L | $0.00 |
| Gross Profit | $10,650.90 | Gross Loss | ($4,726.70) |
| Total # of trades | 73 | Percent profitable | 32.88% |
| Number winning trades | 24 | Number losing trades | 49 |
| Largest winning trade | $2,191.20 | Largest losing trade | ($471.20) |
| Average winning trade | $443.79 | Average losing trade | ($96.46) |
| Ratio avg win/avg loss | 4.60 | Avg trade (win & loss) | $81.15 |
| Max consec. Winners | 4 | Max consec. losers | 9 |
| Avg # bars in winners | 21 | Avg # bars in losers | 5 |
| Max intraday drawdown | ($1,055.90) | | |
| Profit Factor | 2.25 | Max # contracts held | 100 |
| Account size required | $1,055.90 | Return on account | 573.04% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $5,924.10 | Open position P/L | $0.00 |
| Gross Profit | $10,650.90 | Gross Loss | ($4,726.70) |
| Total # of trades | 73 | Percent profitable | 32.88% |
| Number winning trades | 24 | Number losing trades | 49 |
| Largest winning trade | $2,191.20 | Largest losing trade | ($471.20) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win | Loss | Time | Graphs | Settings

*Figure 5. AXP Performance Summary*


TradeStation Strategy Performance Report

**Annual Trading Summary**

***Annual Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| YTD | $2,577.40 | 7.73% | 3.68 | 9 | 55.56% |
| 12 month | $3,415.50 | 12.73% | 3.07 | 14 | 35.71% |
| 98 | $2,002.50 | 6.39% | 2.42 | 9 | 44.44% |
| 97 | $1,030.00 | 3.40% | 3.62 | 4 | 50.00% |
| 96 | $410.00 | 1.37% | 1.80 | 11 | 54.55% |
| 95 | $20.00 | 0.07% | 1.03 | 12 | 16.67% |
| 94 | ($440.20) | (1.48%) | 0.45 | 12 | 16.67% |
| 93 | $236.20 | 0.78% | 1.60 | 12 | 16.67% |
| 92 | $96.20 | 0.32% | 1.49 | 7 | 28.57% |

***Annual Rolling Period Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| 99 | $2,577.40 | 7.73% | 3.68 | 9 | 55.56% |
| 98-99 | $4,579.90 | 14.61% | 2.94 | 18 | 50.00% |
| 97-99 | $5,609.90 | 18.51% | 3.03 | 22 | 50.00% |
| 96-99 | $6,019.90 | 20.13% | 2.84 | 33 | 51.52% |
| 95-99 | $6,039.90 | 20.21% | 2.57 | 45 | 42.22% |
| 94-99 | $5,591.70 | 18.43% | 2.20 | 57 | 36.84% |
| 93-99 | $5,827.90 | 19.36% | 2.15 | 69 | 33.33% |
| 92-99 | $5,924.10 | 19.75% | 2.12 | 76 | 32.89% |

*Figure 6. AXP Annual Trading Summary*

Applied to JY, TrendScore generated a net profit of $72,000 on 110 trades. Only 31.8% of the trades were profitable, but the average winner ($4,399) was 4.02 times as large as the average loser (1,093), resulting in an average trade of $654. The strategy won $1.88 for each $1.00 it lost [Figure 7, JY Performance Summary]. The graph of Monthly Rolling Net Profit shows approximately breakeven results for the first two years (1993-1994) but strong results through the rest of the test period, with a new equity high on the most recent trade [Figure 8, JY MRNP]. When monthly returns were averaged over the seven-year test period, the strategy lost money only in February and November, trading profitably over the other 10 months [Figure 9, JY APBM].

## Suggestions For Improvement

As we mentioned in the section Defining Our Trading Rules, we did not specify a trigger for this strategy, opting to just buy or sell at the market on the next open after a setup. It has been our experience that most strategies can be improved by requiring both a setup and a trigger to enter trades. You might want to try adding a trigger to TrendScore. Consider, for example, buying at a tick above the high of the setup bar, buying at the close of the setup bar plus an Average True Range, or buying at the open plus 50% of the setup bar's range.


TradeStation Strategy Performance Report – TrendScore JY-CONT-Daily

**Performance Summary: All Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $72,000.00 | Open position P/L | $0.00 |
| Gross Profit | $153,975.00 | Gross Loss | ($81,975.00) |
| Total # of trades | 110 | Percent profitable | 31.82% |
| Number winning trades | 35 | Number losing trades | 75 |
| Largest winning trade | $21,025.00 | Largest losing trade | ($5,350.00) |
| Average winning trade | $4,399.29 | Average losing trade | ($1,093.00) |
| Ratio avg win/avg loss | 4.02 | Avg trade (win & loss) | $654.55 |
| Max consec. Winners | 4 | Max consec. losers | 10 |
| Avg # bars in winners | 28 | Avg # bars in losers | 7 |
| Max intraday drawdown | ($18,212.50) | | |
| Profit Factor | 1.88 | Max # contracts held | 1 |
| Account size required | $18,212.50 | Return on account | 395.33% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $90,837.50 | Open position P/L | $0.00 |
| Gross Profit | $97,887.50 | Gross Loss | ($47,350.00) |
| Total # of trades | 54 | Percent profitable | 29.63% |
| Number winning trades | 16 | Number losing trades | 38 |
| Largest winning trade | $21,025.00 | Largest losing trade | ($5,350.00) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | W / Loss | Time | Graphs | Settings

*Figure 7. JY Performance Summary*


*Figure 8. JY MRNP*


*Figure 9. JY APBM*

CHAPTER 9

# Triple Play

Our Triple Play strategy is a variation of Charles Lindsay's well-respected Trident Trading Strategy. The most significant change we made to this strategy for STAD Club is the addition of Bollinger Bands to mechanically locate the crucial chart points. Triple Play is based on three swing highs and lows (a swing high has at least one lower high immediately before and after it; a swing low has at least one higher low immediately before and after it). Since the strategy's rules for long and short positions are analogous, we'll just introduce the long side here. For a long position, Triple Play first identifies a swing low (point 1), a swing high (point 2), and a swing low (point 3) between points 1 and 2 [Figure 1, Triple Play long setup]. Bollinger Bands are overlaid on the price chart to help locate points 1 and 2. For both the long and the short setups, points 1 and 2 must be outside the bands; point 3 must be inside the bands. Also, point 3 must retrace between 38.2% and 61.8% (Fibonacci retracement levels) of the price swing between point 1 and point 2 [Figure 1, Triple Play Chart].

With the setup in place, our Triple Play strategy uses points 1, 2, and 3 to calculate an entry price, initial protective stop, trailing stop, and two price targets [Figure 1, Triple Play Chart]. The strategy's setup, trigger, orders, and exits are described next.


*Figure 1. Triple Play Chart*

## Defining Our Trading Rules

For the Triple Play strategy, we defined both long and short setups, triggers, orders, and exits. We also calculated the Bollinger Bands indicator and the Fibonacci retracement levels. The setups, triggers, orders, and exits are described next.

### Long and Short Setups

a) The setup to buy requires a price move from a swing low (point 1) to a swing high (point 2) and a retracement to a swing low (point 3).

b) The setup to sell requires a price move from a swing high (point 1) to a swing low (point 2) and a retracement to a swing high (point 3).

### Long and Short Triggers

a) The trigger for a long position is a resumption of the up move to the following level: subtract point 1 from point 2, divide the difference by 4, and add the result to point 3.

b) The trigger for a short position is a resumption of the down move to the following level: subtract point 2 from point 1, divide the difference by 4, and subtract the result from point 3.

### Long and Short Orders

a) To enter a long position, place an order to buy two units (200 shares of a stock or two contracts of a commodity) at the long entry trigger on a stop.

b) To enter a short position, place an order to sell short two units (200 shares of a stock or two contracts of a commodity) at the short entry trigger on a stop.

### Long and Short Exits

a) If long two units, exit both at the initial protective stop or exit one unit at target 1 and one unit at either the trailing stop or target 2, whichever is hit first. (The initial protective stop is at point 3 minus one point. The trailing stop, which is activated after a close at point 2 or higher, is calculated as follows: Find the highest close since entry and subtract 25% of the difference between point 1 and point 2. Target 1 is calculated as follows: add point 3 to target 2 and divide the total by 4. Target 2 is calculated as follows: add point 2 to point 3 and subtract point 1 from the total.)

b) If short two units, exit both at the initial protective stop or exit one unit at target 1 and one unit at either the trailing stop or target 2, whichever is hit first. (The calculations for the initial protective stop, the trailing stop, target 1, and target 2 are analogous to the calculations for the long side.)

## Designing & Formatting

This section presents the EasyLanguage instructions and formatting for the strategy, with the EasyLanguage instructions broken down and explained line by line.

### EasyLanguage Strategy Components: Triple Play (STAD11: Triple Play)

### Strategy Inputs (STAD11: Triple Play)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| PatternLength | 25 | The maximum number of bars that can be included in the formation of the Triple Play |
| Strength | 7 | Determines the strength of the swing points |
| BarsToEnter | 10 | Specifies the number of bars after a Triple Play formation occurs that an entry signal can last |
| Length | 11 | Determines the length used by the Bollinger Bands |
| Deviations | 2 | The number of standard deviations used by the Bollinger Bands |

## Signal Components:

Triple Play

## EasyLanguage Signal: Triple Play:

```
Inputs:
        PatternLength(25), Strength(7), BarsToEnter(10), Length(11), Deviations(2);
Variables:
        StrongLow(0), StrongHigh(0), WeakLow(0), WeakHigh(0),
        StrongLowBar(0), StrongHighBar(0), WeakLowBar(99999), WeakHighBar(99999),
        LongEntryPrice(99999), LongExitPrice(0), LongTarget1(0), LongTarget2(0),
        ShortEntryPrice(0), ShortExitPrice(0), ShortTarget1(0), ShortTarget2(0),
        RecentLow(0), RecentHigh(0), FoundAWeakLow(False), FoundAWeakHigh(False),
        Offset(Strength), TrailStopDrop(0), TrailStopRise(0),
        LongTrailingTrigger(0), ShortTrailingTrigger(0), HighClose(0), LowClose(99999),
        BuyCounter(0), SellCounter(0), MakeLongTrade(False), MakeShortTrade(False),
        LongTrailStop(False), ShortTrailStop(False),
        UpperBand(0), LowerBand(0);


UpperBand = BollingerBand(Close, Length, Deviations);
LowerBand = BollingerBand(Close, Length, -Deviations);

StrongHighBar = IFF(StrongHighBar <= PatternLength + Strength, StrongHighBar + 1, StrongHighBar);
StrongLowBar = IFF(StrongLowBar <= PatternLength + Strength, StrongLowBar + 1, StrongLowBar);

If SwingHigh(1, High, Strength, Strength + 1) <> -1 AND High[Strength] > UpperBand[Strength] Then Begin
        StrongHigh = SwingHigh(1, High, Strength, Strength + 1);
        StrongHighBar = Strength;
        RecentLow = MRO(SwingLow(1, Low, Strength, Strength + 1) <> -1, StrongLowBar - StrongHighBar,
1);
        If RecentLow <> -1 Then Begin
                WeakLow = SwingLow(1, Low, Strength, Strength + 1)[Value1];
                WeakLowBar = Value1 + Strength;
                FoundAWeakLow = True;
        End;
End;

If SwingLow(1, Low, Strength, Strength + 1) <> -1 AND Low[Strength] < LowerBand[Strength] Then Begin
        StrongLow = SwingLow(1, Low, Strength, Strength + 1);
        StrongLowBar = Strength;
        RecentHigh = MRO(SwingHigh(1, High, Strength, Strength + 1) <> -1, StrongHighBar - StrongLowBar,
1);
        If RecentHigh <> -1 Then Begin
                WeakHigh = SwingHigh(1, High, Strength, Strength + 1)[Value1];
                WeakHighBar = Value1 + Strength;
                FoundAWeakHigh = True;
        End;
```

```
End;

Condition1 = StrongLowBar < PatternLength + Strength;
Condition2 = StrongHighBar < PatternLength + Strength;

If FoundAWeakLow AND Condition1 Then Begin
        LongEntryPrice = ( (StrongHigh - StrongLow) / 4 ) + WeakLow;
        LongExitPrice = WeakLow - 1 point;
        LongTarget2 = WeakLow + (StrongHigh - StrongLow);
        LongTarget1 = (WeakLow + LongTarget2) / 2;
        LongTrailingTrigger = StrongHigh;
        TrailStopDrop = (StrongHigh - WeakLow) / 4;
        BuyCounter = 0;
        MakeLongTrade = True;
        MakeShortTrade = False;
End;

If FoundAWeakHigh AND Condition2 Then Begin
        ShortEntryPrice = WeakHigh - ( (StrongHigh - StrongLow) / 4 ) ;
        ShortExitPrice = WeakHigh + 1 point;
        ShortTarget2 = WeakHigh - (StrongHigh - StrongLow);
        ShortTarget1 = (WeakHigh + ShortTarget2) / 2;
        ShortTrailingTrigger = StrongLow;
        TrailStopRise = (WeakHigh - StrongLow) / 4;
        SellCounter = 0;
        MakeShortTrade = True;
        MakeLongTrade = False;
End;

FoundAWeakLow = False;
FoundAWeakHigh = False;

If MarketPosition = 0 Then Begin
        If BuyCounter < BarsToEnter Then
                BuyCounter = BuyCounter + 1
        Else
                MakeLongTrade = False;
        If SellCounter < BarsToEnter Then
                SellCounter = SellCounter + 1
        Else
                MakeShortTrade = False;
        If MakeLongTrade Then
                Buy ("Buy 2") 2 Contracts Next Bar at LongEntryPrice Stop;
        If MakeShortTrade Then
                Sell ("Sell 2") 2 Contracts Next Bar at ShortEntryPrice Stop;
End;

If MarketPosition = 1 Then Begin
        MakeLongTrade = False;
        If CurrentContracts = 2 Then Begin
```

```
                LongTrailStop = False;
                HighClose = 0;
                ExitLong ("LDump 1") Next Bar at LongExitPrice Stop;
                ExitLong ("LTarget 1") 1 Contracts Total Next Bar at LongTarget1 Limit;
        End
        Else Begin
                If Close > HighClose Then
                        HighClose = Close;
                ExitLong ("LProtect 2") Next Bar at EntryPrice Stop;
                If HighClose > LongTrailingTrigger Then
                        LongTrailStop = True;
                If LongTrailStop Then
                        ExitLong ("LTrail") Next Bar at HighClose - TrailStopDrop Stop;
                ExitLong ("LTarget 2") Next Bar at LongTarget2 Limit;
        End;
End;

If MarketPosition = -1 Then Begin
        MakeShortTrade = False;
        If CurrentContracts = 2 Then Begin
                ShortTrailStop = False;
                LowClose = 99999;
                ExitShort ("SDump 1") Next Bar at ShortExitPrice Stop;
                ExitShort ("STarget 1") 1 Contracts Total Next Bar at ShortTarget1 Limit;
        End
        Else Begin
                If Close < LowClose Then
                        LowClose = Close;
                ExitShort ("SProtect 2") Next Bar at EntryPrice Stop;
                If LowClose < ShortTrailingTrigger Then
                        ShortTrailStop = True;
                If ShortTrailStop Then
                        ExitShort ("STrail") Next Bar at LowClose + TrailStopRise Stop;
                ExitShort ("STarget 2") Next bar at ShortTarget2 Limit;
        End;
End;
```

## Signal Inputs (Triple Play)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| PatternLength | 25 | The maximum number of bars that can be included in the formation of the Triple Play |
| Strength | 7 | Determines the strength of the swing points |
| BarsToEnter | 10 | Specifies the number of bars after a Triple Play formation occurs that an entry signal will last |
| Length | 11 | Used to determine the Length of the average calculation that the Bollinger Bands are based on |
| Deviations | 2 | The number of standard deviations used by the Bollinger Bands |

## Signal Variables (Triple Play)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| StrongLow | 0 | [Numeric] Stores the value of the nearest Low swing point that was lower than the bottom Bollinger Band |
| StrongHigh | 0 | [Numeric] Stores the value of the nearest High swing point that was higher than the upper Bollinger Band |
| WeakLow | 0 | [Numeric] Stores the value of the nearest Low swing point that was within the Bollinger Bands |
| WeakHigh | 0 | [Numeric] Stores the value of the nearest High swing point that was within the Bollinger Bands |
| StrongLowBar | 0 | [Numeric] Stores the number of bars ago that a Low swing point that was lower than the bottom Bollinger Band occurred |
| StrongHighBar | 0 | [Numeric] Stores the number of bars ago that a High swing point that was higher than the upper Bollinger Band occurred |
| WeakLowBar | 99999 | [Numeric] Stores the number of bars ago that a Low swing point that was not lower than the bottom Bollinger Band occurred |
| WeakHighBar | 99999 | [Numeric] Stores the number of bars ago that a Low swing point that was not higher than the upper Bollinger Band occurred |
| LongEntryPrice | 99999 | [Numeric] Stores the value calculated for a Long Entry when the setup formation is completed |
| LongExitPrice | 0 | [Numeric] Stores the value calculated for a Long Entry when the setup formation is completed |
| LongTarget1 | 0 | [Numeric] Used to store the value for the exiting the first portion of a Long trade at a profit |
| LongTarget2 | 0 | [Numeric] Used to store the value for the exiting the final portion of a Long trade at a profit |
| ShortEntryPrice | 0 | [Numeric] Stores the value calculated for a Short Entry when the setup formation is completed |
| ShortExitPrice | 0 | [Numeric] Stores the value calculated for a Short Entry when the setup formation is completed |
| ShortTarget1 | 0 | [Numeric] Used to store the value for the exiting the first portion of a Short trade at a profit |
| ShortTarget2 | 0 | [Numeric] Used to store the value for the exiting the final portion of a Short trade at a profit |
| RecentLow | 0 | [Numeric] Stores the value of a Low swing point as the middle swing point of a Long side formation |
| RecentHigh | 0 | [Numeric] Stores the value of a High swing point as the middle swing point of a Short side formation |
| FoundAWeakLow | False | [True/False] Used to signal that a "weak" swing point has occurred between the high and low swing points to complete the formation |

## Signal Variables (Triple Play) continued from previous page

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| FoundAWeakHigh | False | [True/False] Used to signal that a "weak" swing point has occurred between the high and low swing points to complete the formation |
| TrailStopDrop | 0 | [Numeric] Stores the value to subtract from the highest Close since entry for use in calculating a Long trailing stop |
| TrailStopRise | 0 | [Numeric] Stores the value to add to the lowest Close since entry for use in calculating a Short trailing stop |
| LongTrailingTrigger | 0 | [Numeric] Stores the value to reach that will trigger the exit orders for the trailing stop, Long side only |
| ShortTrailingTrigger | 0 | [Numeric] Stores the value to reach that will trigger the exit orders for the trailing stop, Short side only |
| HighClose | 0 | [Numeric] Used to keep track of the highest Close since entry of the position |
| LowClose | 99999 | [Numeric] Used to keep track of the lowest Close since entry of the position |
| BuyCounter | 0 | [Numeric] Used to count the number of bars that pass since the completion of the Long formation, until the position is entered |
| SellCounter | 0 | [Numeric] Used to count the number of bars that pass since the completion of the Short formation, until the position is entered |
| MakeLongTrade | False | [True/False] Used to generate the Buy order |
| MakeShortTrade | False | [True/False] Used to generate the Sell order |
| LongTrailStop | False | [True/False] Used to indicate that the trailing stop for the Long side has been triggered |
| ShortTrailStop | False | [True/False] Used to indicate that the trailing stop for the Short side has been triggered |
| UpperBand | 0 | [Numeric] Holds the value calculated for the upper Bollinger Band |
| LowerBand | 0 | [Numeric] Holds the value calculated for the lower Bollinger Band |

## Setup

The major or "strong" swing points that are basis for the formation are those that break the upper and lower Bollinger Bands. We calculate the upper and lower band values with the first two statements.

```
UpperBand = BollingerBand(Close, Length, Deviations);
LowerBand = BollingerBand(Close, Length, -Deviations);
```

The number of bars ago that a strong swing point occurred is incremented using the IFF() function. IFF evaluates a condition and if true, returns the second parameter entered in the function. If false, the third parameter will be returned. Both the second and third parameters must be a numeric value. In this case, if StrongHighBar or StrongLowBar has not yet reached the limit of the PatternLength plus the Strength of the swing point, they will be incremented by one.

```
StrongHighBar = IFF(StrongHighBar <= PatternLength + Strength, StrongHighBar + 1, StrongHighBar);
StrongLowBar = IFF(StrongLowBar <= PatternLength + Strength, StrongLowBar + 1, StrongLowBar);
```

If a swing high is found that is also above the upper Bollinger Band, we store the value of the swing point into StrongHigh and the number of bars ago it occurred into StrongHighBar. We then run a check to see if there was a "weak" swing low between this strong high point and the most recent strong low point by using the MRO() function. The MRO() function will return the number of bars ago a condition occurred, and -1 if the condition was not true within the specified range. If a weak swing low is detected with the MRO() function, we use RecentLow to store the value of that low point into WeakLow and the number of bars ago that it occurred into WeakLowBar. The FoundAWeakLow condition is set to True to be used as a trigger for the Triple Play formation.

```
If SwingHigh(1, High, Strength, Strength + 1) <> -1 AND High[Strength] > UpperBand[Strength] Then Begin
        StrongHigh = SwingHigh(1, High, Strength, Strength + 1);
        StrongHighBar = Strength;
        RecentLow = MRO(SwingLow(1, Low, Strength, Strength + 1) <> -1, StrongLowBar - StrongHighBar,
1);
        If RecentLow <> -1 Then Begin
                WeakLow = SwingLow(1, Low, Strength, Strength + 1)[Value1];
                WeakLowBar = Value1 + Strength;
                FoundAWeakLow = True;
        End;
End;
```

On the opposite side, if a swing low is found that is below the lower Bollinger Band, we store the value of the swing point into StrongLow and the number of bars ago it occurred into StrongLowBar. We then run a check to see if there was a "weak" swing high between this strong low point and the most recent strong high point, again using the MRO() function. If a weak swing high is detected with the MRO() function, RecentHigh is used to store the value of the high point into WeakHigh and the number of bars ago that it occurred into WeakHighBar. The FoundAWeakHigh condition is set to True to be used as a trigger for our formation.

```
If SwingLow(1, Low, Strength, Strength + 1) <> -1 AND Low[Strength] < LowerBand[Strength] Then Begin
        StrongLow = SwingLow(1, Low, Strength, Strength + 1);
        StrongLowBar = Strength;
        RecentHigh = MRO(SwingHigh(1, High, Strength, Strength + 1) <> -1, StrongHighBar - StrongLowBar,
1);
        If RecentHigh <> -1 Then Begin
                WeakHigh = SwingHigh(1, High, Strength, Strength + 1)[Value1];
                WeakHighBar = Value1 + Strength;
                FoundAWeakHigh = True;
        End;
End;
```

In addition to the formation of the swing points, we check that the further strong point has not occurred too far back in the past. This is done by checking the number of bars ago the swing point occurred to the value of PatternLength plus Strength.

```
Condition1 = StrongLowBar < PatternLength + Strength;
Condition2 = StrongHighBar < PatternLength + Strength;
```

By the manner in which the setup is evaluated, FoundAWeakLow will only be True on the bar that completes the formation of the Triple Play. If on that bar the range of the formation is within the PatternLength, we then set up all of the values to be used for the Long Entry and its Exits. This includes the calculation of LongEntryPrice, LongExitPrice and the profit targets labeled LongTarget 1 and 2. These are respectively, the value of the weak low plus one-fourth the difference between the strong points, one point below the weak low, and the weak low plus the difference between the strong points and the average of the weak low with that value. The trigger for the trailing stop is when the price reaches the current strong high value and the value used to determine the trailing stop is stored in TrailStopDrop, one-fourth the difference between the strong high and weak low. BuyCounter is set to zero in order to count the number of bars that pass after the formation of our pattern, and MakeLongTrade is set to True and MakeShortTrade set to False.

```
If FoundAWeakLow AND Condition1 Then Begin
        LongEntryPrice = ( (StrongHigh - StrongLow) / 4 ) + WeakLow;
        LongExitPrice = WeakLow - 1 point;
        LongTarget2 = WeakLow + (StrongHigh - StrongLow);
        LongTarget1 = (WeakLow + LongTarget2) / 2;
        LongTrailingTrigger = StrongHigh;
        TrailStopDrop = (StrongHigh - WeakLow) / 4;
        BuyCounter = 0;
        MakeLongTrade = True;
        MakeShortTrade = False;
End;
```

Respectively, FoundAWeakHigh will only be True on the bar that completes the formation of the short side. If on that bar the range of the formation is within the PatternLength, we then set up all of the values to be used for the Short Entry and its Exits. This includes the calculation of ShortEntryPrice, ShortExitPrice and the profit targets labeled ShortTarget 1 and 2. These are respectively, the value of the weak high minus one-fourth the difference between the strong points, one point above the weak high, and the weak high minus the difference between the strong points and the average of the weak high with that value. The trigger for the trailing stop is when the price reaches the current strong low value and the value used to determine the trailing stop is stored in TrailStopRise, one-fourth the difference between the weak high and strong low. SellCounter is set to zero in order to count the number of bars that pass after the formation of our pattern, and MakeShortTrade is set to True and MakeLongTrade set to False.

```
If FoundAWeakHigh AND Condition2 Then Begin
        ShortEntryPrice = WeakHigh - ( (StrongHigh - StrongLow) / 4 ) ;
        ShortExitPrice = WeakHigh + 1 point;
        ShortTarget2 = WeakHigh - (StrongHigh - StrongLow);
        ShortTarget1 = (WeakHigh + ShortTarget2) / 2;
        ShortTrailingTrigger = StrongLow;
        TrailStopRise = (WeakHigh - StrongLow) / 4;
        SellCounter = 0;
        MakeShortTrade = True;
        MakeLongTrade = False;
End;
```

After the Long or Short values are set the FoundAWeakLow and FoundAWeakHigh values are no longer needed. On every bar they are always reset to false.

```
FoundAWeakLow = False;
FoundAWeakHigh = False;
```

## Long and Short Entry

If there is currently no position entered, we check first the Buy and Sell counters. If either has been reset to zero and is currently less than BarsToEnter (Input), then they are incremented. If they are equal to BarsToEnter, then the respective "Make Trade" trigger variables are set to False. If either of the "Make Trade" triggers are True, the respective Buy or Sell order will be placed for 2 contracts on the next bar, at the entry price determined when the formation of the Triple Play was complete.

```
If MarketPosition = 0 Then Begin
        If BuyCounter < BarsToEnter Then
                BuyCounter = BuyCounter + 1
        Else
                MakeLongTrade = False;
        If SellCounter < BarsToEnter Then
                SellCounter = SellCounter + 1
        Else
                MakeShortTrade = False;
        If MakeLongTrade Then
                Buy ("Buy 2") 2 Contracts Next Bar at LongEntryPrice Stop;
        If MakeShortTrade Then
                Sell ("Sell 2") 2 Contracts Next Bar at ShortEntryPrice Stop;
End;
```

## Long Exits

When a Long position is taken, the first thing done is to turn off the trigger for MakeLongTrade. Then the number of current contracts is tested to determine what types of actions to take. If both contracts are still entered, LongTrailStop is set to False and HighClose is set to 0, they are to be used if the position is closed out to only one contract. With two exits, we generate a protective exit that closes the entire position and a target exit which scales out one half of the position.

```
If MarketPosition = 1 Then Begin
        MakeLongTrade = False;
        If CurrentContracts = 2 Then Begin
                LongTrailStop = False;
                HighClose = 0;
                ExitLong ("LDump 1") Next Bar at LongExitPrice Stop;
                ExitLong ("LTarget 1") 1 Contracts Total Next Bar at LongTarget1 Limit;
        End
```

If the target exit was hit, we would only have one contract left. We keep track of the highest close in HighClose and generate a breakeven stop at the price of entry. The trailing stop trigger is set when the highest close (of the single contract position) crosses above LongTrailingTrigger, which was calculated as the strong high value in the Triple Play formation. If the LongTrailStop is in effect, it will trigger an exit at a value of the highest close of the single contract position minus one-fourth the difference between the strong high and weak low. In addition, a second target stop for a profit is generated if the price continues to rise.

```
        Else Begin
                If Close > HighClose Then
                        HighClose = Close;
                ExitLong ("LProtect 2") Next Bar at EntryPrice Stop;
                If HighClose > LongTrailingTrigger Then
                        LongTrailStop = True;
                If LongTrailStop Then
                        ExitLong ("LTrail") Next Bar at HighClose - TrailStopDrop Stop;
                ExitLong ("LTarget 2") Next Bar at LongTarget2 Limit;
        End;
End;
```

## Short Exits

When a Short position is taken, we turn off the trigger for MakeShortTrade. Again, the number of current contracts is tested to determine what types of actions to take. If both contracts are still entered, ShortTrailStop is set to False and LowClose is set to 99999, they are to be used if the position is closed out to only one contract. With two exits, we generate a protective exit that closes the entire position and a target exit which scales out one half of the position.

```
If MarketPosition = -1 Then Begin
        MakeShortTrade = False;
        If CurrentContracts = 2 Then Begin
                ShortTrailStop = False;
                LowClose = 99999;
                ExitShort ("SDump 1") Next Bar at ShortExitPrice Stop;
                ExitShort ("STarget 1") 1 Contracts Total Next Bar at ShortTarget1 Limit;
        End
```

If the target exit was hit, we would only have one contract left. We keep track of the lowest close in LowClose and generate a breakeven stop at the price of entry. The trailing stop trigger is set when the lowest close (of the single contract position) crosses below ShortTrailingTrigger, which was calculated as the strong low value in the Triple Play formation. If the ShortTrailStop is in effect, it will trigger an exit at a value of the lowest close of the single contract position plus one-fourth the difference between the weak high and strong low. In addition, a second target stop for a profit is generated if the price continues to fall.

```
        Else Begin
                If Close < LowClose Then
                        LowClose = Close;
                ExitShort ("SProtect 2") Next Bar at EntryPrice Stop;
                If LowClose < ShortTrailingTrigger Then
                        ShortTrailStop = True;
                If ShortTrailStop Then
                        ExitShort ("STrail") Next Bar at LowClose + TrailStopRise Stop;
                ExitShort ("STarget 2") Next bar at ShortTarget2 Limit;
        End;
End;
```

## Testing & Improving

We tested Triple Play on Microsoft (MSFT), Wal-Mart (WMT), and Japanese Yen futures (JY). For MSFT and WMT, we tested the long side only on daily bars from 11/95 - 11/99, with maxbarsback set to 70. We deducted $.10 per share for slippage and $.05 per share for commission. For JY, we tested both the long and short sides on daily bars from 5/84 to 11/99, with maxbarsback set to 65. We deducted $40 per contract for slippage and $10 per contract for commission.

First, let's see how Triple Play performed on MSFT. Here are the optimized values:

PatternLength = 70

Strength = 4

BarsToEnter = 30

Length = 30

Deviations = 2

Our strategy gained $5,205 on 12 trades, with 75% profitable and a 4.17 ratio of average win to average loss [Figure 2, MSFT Performance Summary]. The largest winner ($1,632) and the average winner ($628) far eclipsed the largest loser (-$414) and the average loser (-$150). The average trade (wins & losses) earned $433. Perhaps the strongest statistic is the Profit Factor: Triple Play won $12.51 for each $1.00 it lost.


TradeStation Strategy Performance Report - "Triple Play 88and96 MSFT-Daily

**Performance Summary: All Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $5,205.90 | Open position P/L | $0.00 |
| Gross Profit | $5,656.10 | Gross Loss | ($472.30) |
| Total # of trades | 12 | Percent profitable | 75.00% |
| Number winning trades | 9 | Number losing trades | 3 |
| Largest winning trade | $1,632.50 | Largest losing trade | ($414.20) |
| Average winning trade | $628.99 | Average losing trade | ($150.77) |
| Ratio avg win/avg loss | 4.17 | Avg trade (win & loss) | $433.82 |
| Max consec. Winners | 6 | Max consec. losers | 1 |
| Avg # bars in winners | 5 | Avg # bars in losers | 3 |
| Max intraday drawdown | ($732.90) | | |
| Profit Factor | 12.51 | Max # contracts held | 200 |
| Account size required | $732.90 | Return on account | 710.40% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $5,205.90 | Open position P/L | $0.00 |
| Gross Profit | $5,656.10 | Gross Loss | ($472.30) |
| Total # of trades | 12 | Percent profitable | 75.00% |
| Number winning trades | 9 | Number losing trades | 3 |
| Largest winning trade | $1,632.50 | Largest losing trade | ($414.20) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win / Loss | Time | Graphs | Settings

*Figure 2. MSFT Performance Summary*

The strategy made money in each of the four years of our test period [Figure 3, MSFT Annual Trading Summary]. The Equity Curve shows approximately breakeven performance on the first seven trades but a very strong performance on trades eight to 23 [Figure 4, MSFT Equity Curve].

TradeStation Strategy Performance Report

## Annual Trading Summary

***Annual Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| YTD | $3,062.50 | 9.53% | 103.08 | 10 | 80.00% |
| 12 month | $3,320.00 | 12.43% | 15.28 | 6 | 50.00% |
| 98 | $1,853.80 | 6.12% | N/A | 6 | 100.00% |
| 97 | $257.10 | 0.86% | 1.60 | 5 | 60.00% |
| 96 | $32.40 | 0.11% | N/A | 2 | 100.00% |

***Annual Rolling Period Analysis (Mark-To-Market):***

| Period | Net Profit | % Gain | Profit Factor | # Trades | % Profitable |
|---|---|---|---|---|---|
| 99 | $3,062.50 | 9.53% | 103.08 | 10 | 80.00% |
| 98-99 | $4,916.30 | 16.23% | 164.88 | 16 | 87.50% |
| 97-99 | $5,173.40 | 17.23% | 12.27 | 21 | 80.95% |
| 96-99 | $5,205.80 | 17.35% | 12.34 | 23 | 82.61% |

*Figure 3. MSFT Annual Trading Summary*

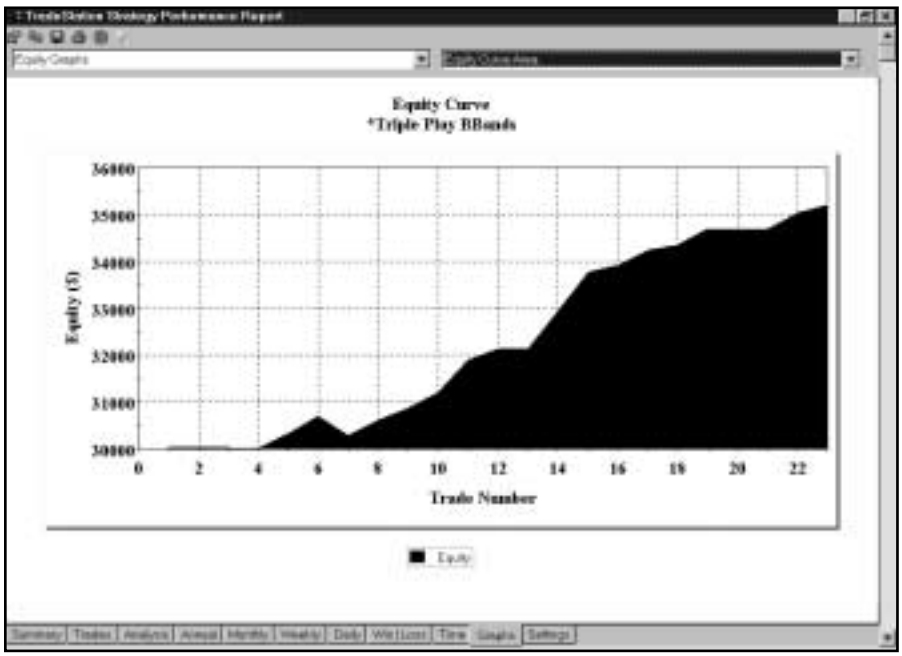



*Figure 4. MSFT Equity Curve*

Next, let's take a look at Triple Play's results for WMT. The optimized values are as follows:

PatternLength = 60

Strength = 4

BarsToEnter = 20

Length = 20

Deviations = 1

Applied to WMT, Triple Play earned $4,729 on 21 trades [Figure 5, WMT Performance Summary]. Seventy-six percent of the trades were profitable, and the average winner ($339) was 2.41 times as large as the average loser ($141). The average trade (wins & losses) made $225. Our strategy posted seven consecutive winners versus only two consecutive losers. The Profit Factor indicates that the strategy won $7.70 for each $1.00 it lost.


TradeStation Strategy Performance Report - "Triple Play BBands WMT-Daily

**Performance Summary: All Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $4,729.40 | Open position P/L | $0.00 |
| Gross Profit | $5,435.60 | Gross Loss | ($706.20) |
| Total # of trades | 21 | Percent profitable | 76.19% |
| Number winning trades | 16 | Number losing trades | 5 |
| Largest winning trade | $935.60 | Largest losing trade | ($300.00) |
| Average winning trade | $339.72 | Average losing trade | ($141.24) |
| Ratio avg winning loss | 2.41 | Avg trade (win & loss) | $225.21 |
| Max consec. Winners | 7 | Max consec. losers | 2 |
| Avg # bars in winners | 10 | Avg # bars in losers | 4 |
| Max intraday drawdown | ($625.00) | | |
| Profit Factor | 7.70 | Max # contracts held | 200 |
| Account size required | $625.00 | Return on account | 756.70% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $4,729.40 | Open position P/L | $0.00 |
| Gross Profit | $5,435.60 | Gross Loss | ($706.20) |
| Total # of trades | 21 | Percent profitable | 76.19% |
| Number winning trades | 16 | Number losing trades | 5 |
| Largest winning trade | $935.60 | Largest losing trade | ($300.00) |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win/Loss | Time | Graphs | Settings

*Figure 5. WMT Performance Summary*

The Equity Curve gained ground slowly during the first 18 trades but advanced strongly throughout the rest of the test period [Figure 6, WMT Equity Curve]. The Underwater Equity Curve shows minimal drawdowns-the largest was about 1.25 percent-and 17 new equity highs [Figure 7, WMT Underwater Equity Curve]. Triple Play's Average Profit by Month was perfect: the strategy earned money in every month when monthly returns were averaged over the length of the test period [Figure 8, WMT Average Profit by Month]. The graph of Maximum Favorable Excursion indicates that our strategy did a good job of locking in profits; note that all the trades that posted open profits of $200 or more were eventually closed out as winners [Figure 9, WMT Maximum Favorable Excursion].

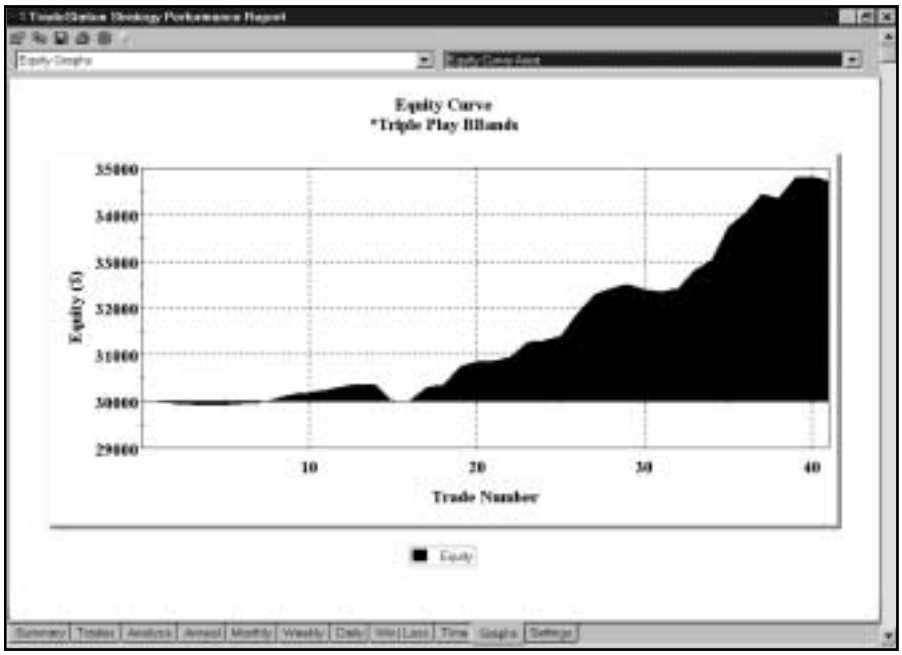



*Figure 6. WMT Equity Curve*

Finally, let's consider Triple Play's handling of JY. Here are the optimized values:

PatternLength = 30

Strength = 6

BarsToEnter = 11

Length = 20

Deviations = 2

Applied to the Yen, Triple Play produced a net profit of $31,220 per contract [Figure 10, JY Performance Summary]. The strategy won on 53% of its trades, with an average winner ($3,053) 3.18 times as large as an average loser ($960) and an average trade (wins & losses) of $1,200. The Equity Curve [Figure 11, JY Equity Curve] moved into positive territory after the first few trades and stayed positive for the length of the test period-climbing to a new equity high on the test's final trade.

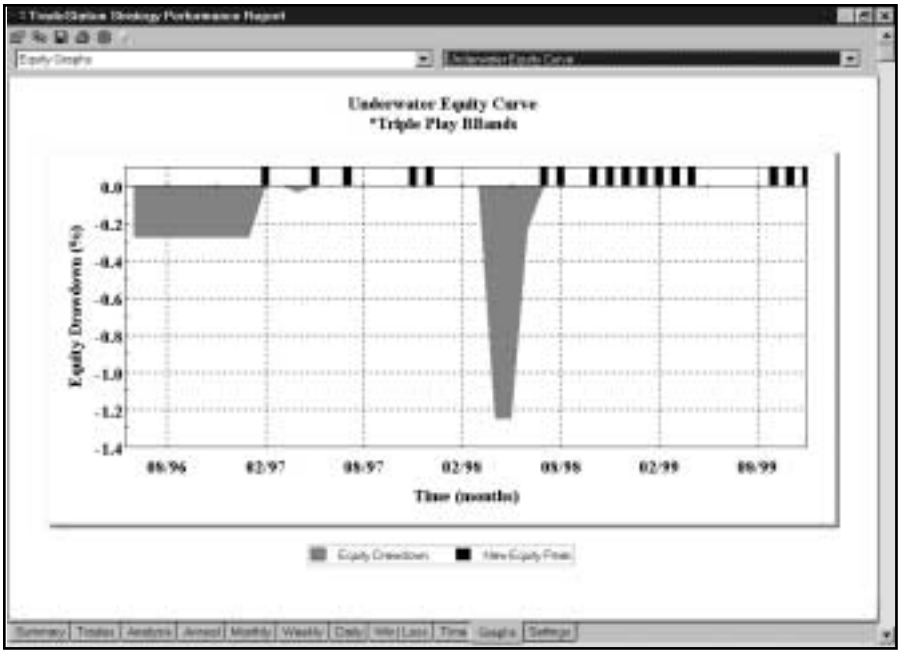



*Figure 7. WMT Underwater Equity Curve*

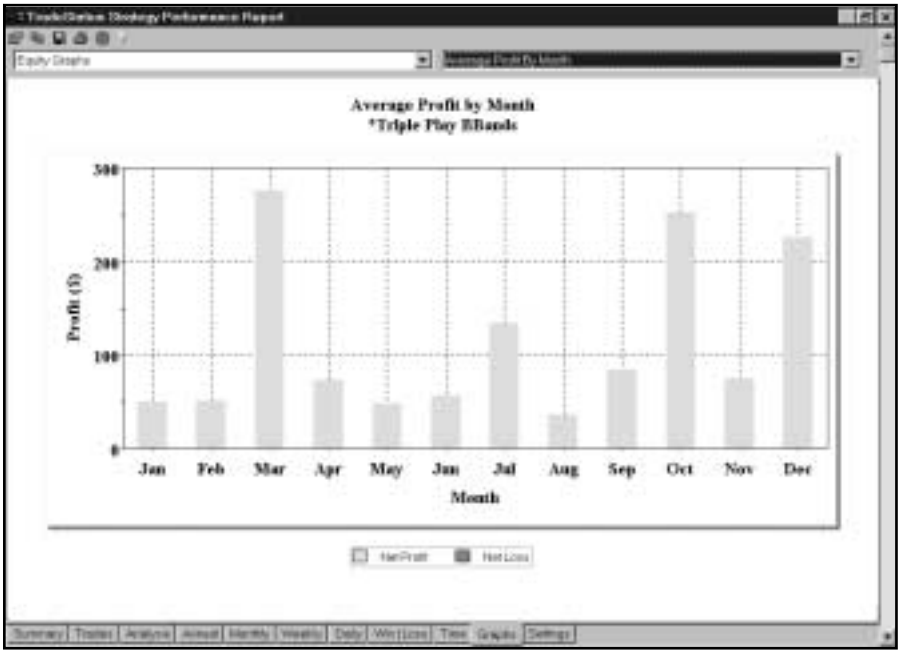



*Figure 8. WMT Average Profit by Month*

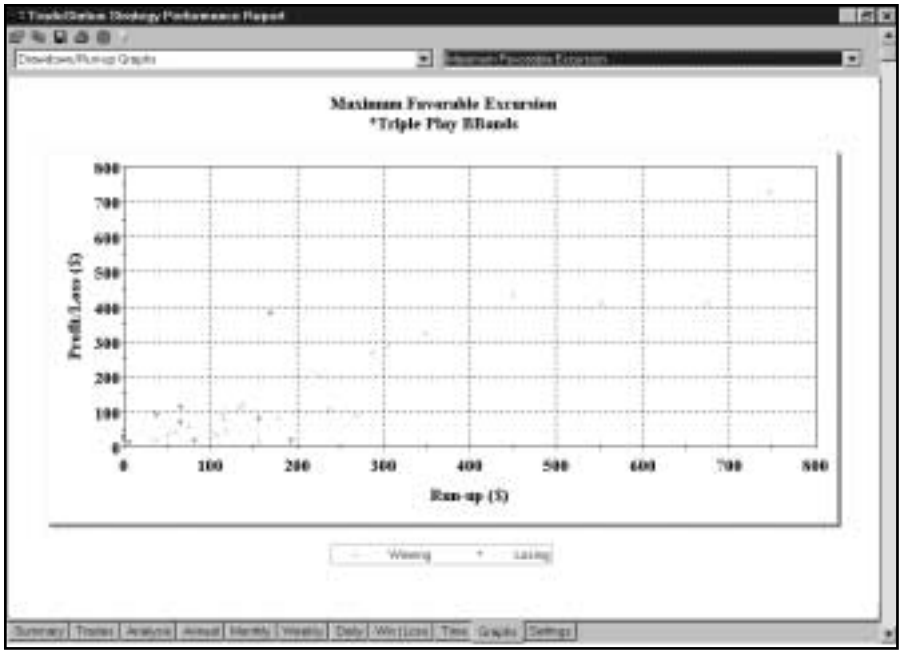



*Figure 9. WMT Maximum Favorable Excursion*


*Figure 10. JY Performance Summary*

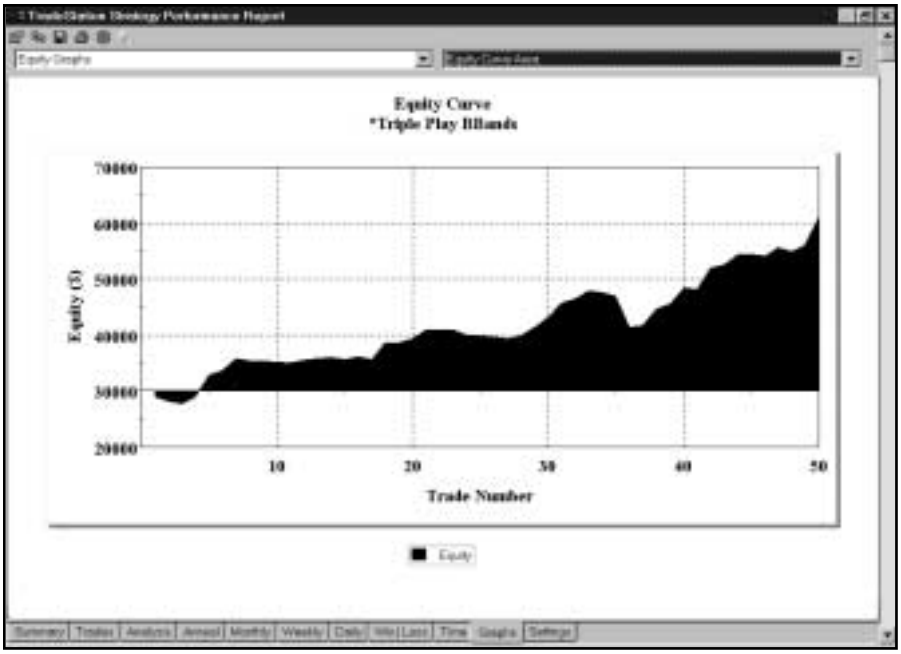



*Figure 11. JY Equity Curve*

## Suggestions For Improvement

Although we didn't perform any optimizations for this strategy, three of its values could probably be improved through optimization. The divisor in the formula for calculating the entry price has a default value of four; you may want to test divisors of three and five. The formula for determining target 1 includes a divisor of two; perhaps a divisor of 1.5 or 2.5 would work better. Finally, the formula for calculating the trailing stop includes a divisor of four; why not test the strategy with divisors of three or five?

CHAPTER 10

# Win One for the Gapper

Our Win One for the Gapper strategy begins with an indicator that measures the amount of "trendiness" (directional movement) in a market, then adds a pattern designed by Larry Williams, and finally adds four stops from StrategyBuilder. First, the strategy employs the Directional Movement Index to determine the direction of the market and whether the market is in a choppy or trending mode. Second, the strategy looks for an opening gap against the direction of the trend. Third, the strategy enters a position when the gap is closed (the opening gap setup and trigger are based on Larry Williams' "Oops" component). Fourth, the strategy exits at the protective stop, the breakeven stop, the trailing stop, or the big profit stop.

# Defining Our Trading Rules

For the Win One for the Gapper strategy, we defined long and short setups, triggers, orders, and exits. We also calculated the Directional Movement Index (DMI) and the Average True Range (ATR). The setups, triggers, orders, and exits are described next.

## Long and Short Setups

a) If DMIPlus is greater than DMIMinus, ADX is greater than it was on the previous bar, and the open is less than the previous bar's low, you have a setup to buy

b) If DMIPlus is less than DMIMinus, ADX is greater than it was on the previous bar, and the open is greater than the previous bar's high, you have a setup to sell short.

## Long and Short Triggers

a) A long position is triggered if prices rise (after the gap-lower open) to the previous bar's low.

b) A short position is triggered if prices fall (after the gap-higher open) to the previous bar's high.

## Long and Short Orders

a) With a setup to buy in effect, place an order to buy at the previous bar's low on a stop.

b) With a setup to sell short in effect, place an order to sell at the previous bar's high on a stop.

## Long and Short Exits

a) Exit a long position at the protective stop, breakeven stop, trailing stop, or big profit stop.

b) Exit a short position at the protective stop, breakeven stop, trailing stop, or big profit stop.

# Designing & Formatting

This section presents the EasyLanguage instructions and formatting for the system, with the EasyLanguage instructions broken down and explained line by line.

## EasyLanguage Strategy Components: Win One for the Gapper (STAD11: Win Gapper)

### Strategy Inputs (STAD11: Win Gapper)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| BreakevenATRs | 4 | The floor value, the number of Average True Ranges above/below the Entry Price at which the Stop becomes active for the position |
| ATRLength | 10 | The number of bars used in the calculation of the Average True Range (ATR) |
| GapperLength | 14 | The number of bars to use to calculate the ADX, DMI+ and DMI- values |
| TrailingATRs | 4 | The number of Average True Ranges (ATR) used as the trailing stop |
| ProtectiveATRs | 3 | The number of Average True Ranges (ATR) used as the protective stop |
| BigProfitATRs | 4 | Number of Average True Ranges used to determine the "Big Profit" level |
| ExitBars | 3 | Length, expressed in bars, used to determine the number of bars used in the trailing stop after the "Big Profit" level has been achieved |

### Signal Components:

1. Win One 4 the Gapper
2. ATR Big Profit Stop
3. ATR Breakeven Stop
4. ATR Protective Stop
5. ATR Trailing Stop
6. Last Bar Exit

## EasyLanguage Signal: Win One 4 the Gapper:

```
Inputs: Length(14);

If ADX(Length) > 25 AND ADX(Length) > ADX(Length)[1] Then Begin
        If DMIPlus(Length) > DMIMinus(Length) AND Open of Next Bar < Low Then
                Buy Next Bar at Low Stop;
        If DMIPlus(Length) < DMIMinus(Length) AND Open of Next Bar > High Then
                Sell Next Bar at High Stop;
End;
```

## Signal Inputs (Win One 4 the Gapper)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| Length | 14 | The number of bars to use to calculate the ADX, DMI+ and DMI- values |

## Long and Short Entry

The common requirement for the Long and Short entry is that the ADX must be greater than 25 and rising. By using this in the conditional portion of an If-Then-Begin statement, we can evaluate the dependent conditions separately. The additional condition for the Long entry is the DMI+ greater than the DMI- and the next bar opening lower than the current Low. The Short entry looks for the opposite condition. By using a simple greater than / less than comparison for the DMI values and Open of Tomorrow for the first tick of the next bar, we can generate the following Buy and Sell statements.

```
If ADX(Length) > 25 AND ADX(Length) > ADX(Length)[1] Then Begin
        If DMIPlus(Length) > DMIMinus(Length) AND Open of Next Bar < Low Then
                Buy Next Bar at Low Stop;
        If DMIPlus(Length) < DMIMinus(Length) AND Open of Next Bar > High Then
                Sell Next Bar at High Stop;
End;
```

## EasyLanguage Signal: ATR Big Profit Stop:

** See Common Stops Appendix

## EasyLanguage Signal: ATR Breakeven Stop:

** See Common Stops Appendix

## EasyLanguage Signal: ATR Protective Stop:

** See Common Stops Appendix

## EasyLanguage Signal: ATR Trailing Stop:

** See Common Stops Appendix

## EasyLanguage Signal: Last Bar Exit:

** See Common Stops Appendix

# Testing & Improving

We tested the Win One for the Gapper strategy on daily bars for IBM, the British Pound (BP), and Coffee (KC). For IBM, we tested from 2/92 to 11/99 and deducted $.10 per share for slippage and $.05 per share for commission. For the British Pound, we tested from 6/84 to 11/99, deducting $40 for slippage and $10 for commission. For Coffee, we tested from 2/92 to 11/99, deducting $40 for slippage and $10 for commission.

Let's begin our survey of Gapper's performance with IBM. The optimized values are as follows:

ATR Protective Stop = 3

ATR Breakeven Stop = 6

ATR Trailing Stop = 4

ATR Big Profit Stop = 10

Exit Bars = 1

Gapper Length = 14

Figure 1 shows a very profitable trade in IBM. ADX was above 25 and rising, and DMIPlus was above DMIMinus. The stock gapped lower on the open, and Gapper bought on a rally to the previous bar's low. The exit was about as good as it gets-just one bar after a high that wasn't exceeded for more than a month [Figure 1, IBM chart].

Applied to IBM, Gapper earned a net profit of $3,310 on only 12 trades [Figure 2, IBM Performance Summary]. Sixty-six percent of the trades were profitable, with the average win a powerful 7.19 times as large as the average loss. The largest winning trade ($1,235) far surpassed the largest losing trade ($115), and Gapper's average trade made $275. The Profit Factor indicates that our strategy earned $14.38 for each $1.00 it lost.

Figure 3 lists each Gapper trade in IBM. It shows that the strategy lost money on its first three trades but rebounded-winning on eight of the next nine trades [Figure 3, IBM Trades]. The final graph for IBM [Figure 4, IBM Average Profit by Month] indicates that Gapper traded profitably in 10 of 12 months when monthly returns are averaged over the eight-year test period.

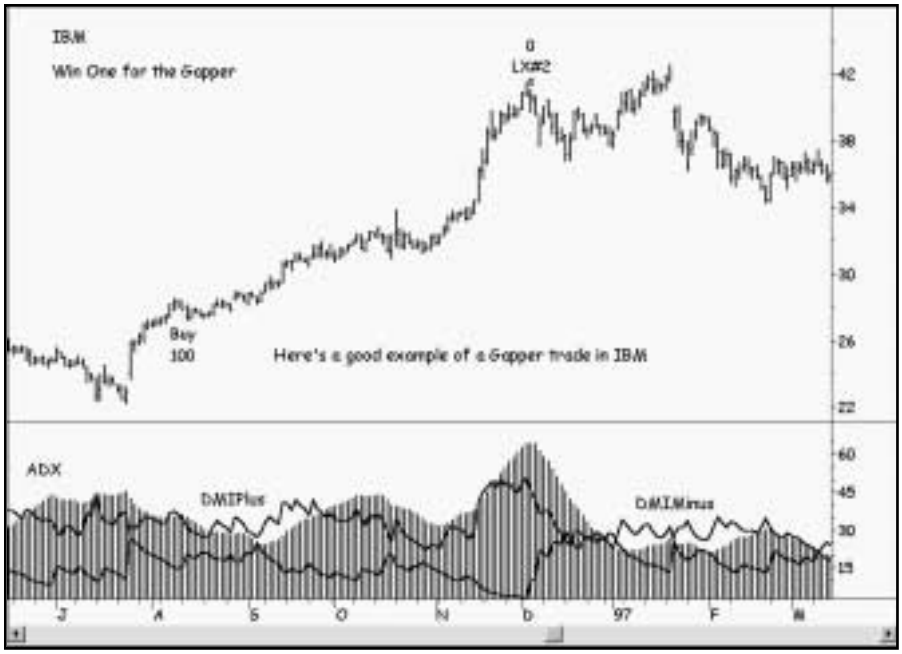
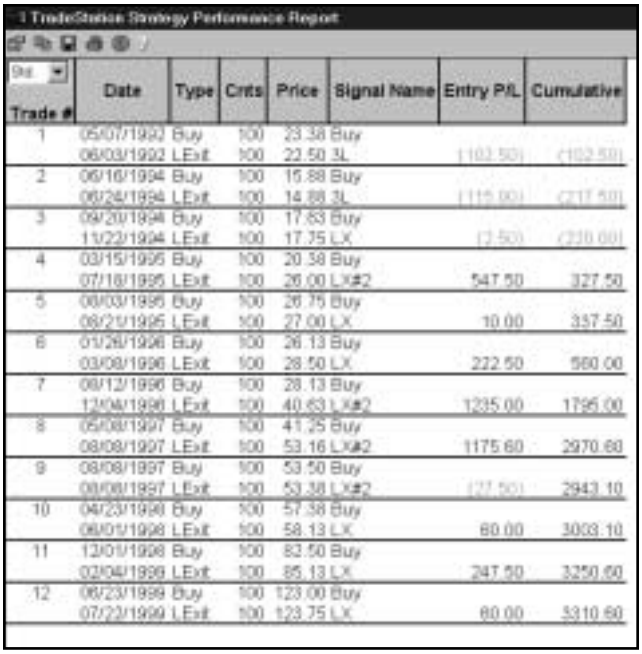



*Figure 1. IBM chart*


*Figure 2. IBM Performance Summary*



TradeStation Strategy Performance Report

| Trade # | Date | Type | Cnts | Price | Signal Name | Entry P/L | Cumulative |
|---|---|---|---|---|---|---|---|
| 1 | 05/07/1992 | Buy | 100 | 23.38 | Buy | | |
| | 06/03/1992 | LExit | 100 | 22.50 | 3L | (102.50) | (102.50) |
| 2 | 06/16/1994 | Buy | 100 | 15.88 | Buy | | |
| | 06/24/1994 | LExit | 100 | 14.88 | 3L | (115.00) | (217.50) |
| 3 | 09/20/1994 | Buy | 100 | 17.63 | Buy | | |
| | 11/22/1994 | LExit | 100 | 17.75 | LX | (2.50) | (220.00) |
| 4 | 03/15/1995 | Buy | 100 | 20.38 | Buy | | |
| | 07/18/1995 | LExit | 100 | 26.00 | LX#2 | 547.50 | 327.50 |
| 5 | 08/03/1995 | Buy | 100 | 26.75 | Buy | | |
| | 08/21/1995 | LExit | 100 | 27.00 | LX | 10.00 | 337.50 |
| 6 | 01/26/1996 | Buy | 100 | 26.13 | Buy | | |
| | 03/08/1996 | LExit | 100 | 28.50 | LX | 222.50 | 560.00 |
| 7 | 09/12/1996 | Buy | 100 | 28.13 | Buy | | |
| | 12/04/1996 | LExit | 100 | 40.63 | LX#2 | 1235.00 | 1795.00 |
| 8 | 05/08/1997 | Buy | 100 | 41.25 | Buy | | |
| | 08/08/1997 | LExit | 100 | 53.16 | LX#2 | 1175.60 | 2970.60 |
| 9 | 08/08/1997 | Buy | 100 | 53.50 | Buy | | |
| | 08/08/1997 | LExit | 100 | 53.38 | LX#2 | (27.50) | 2943.10 |
| 10 | 04/23/1998 | Buy | 100 | 57.38 | Buy | | |
| | 06/01/1998 | LExit | 100 | 58.13 | LX | 60.00 | 3003.10 |
| 11 | 12/01/1998 | Buy | 100 | 82.50 | Buy | | |
| | 02/04/1999 | LExit | 100 | 85.13 | LX | 247.50 | 3250.60 |
| 12 | 06/23/1999 | Buy | 100 | 123.00 | Buy | | |
| | 07/23/1999 | LExit | 100 | 123.75 | LX | 60.00 | 3310.60 |

*Figure 3. IBM Trades*

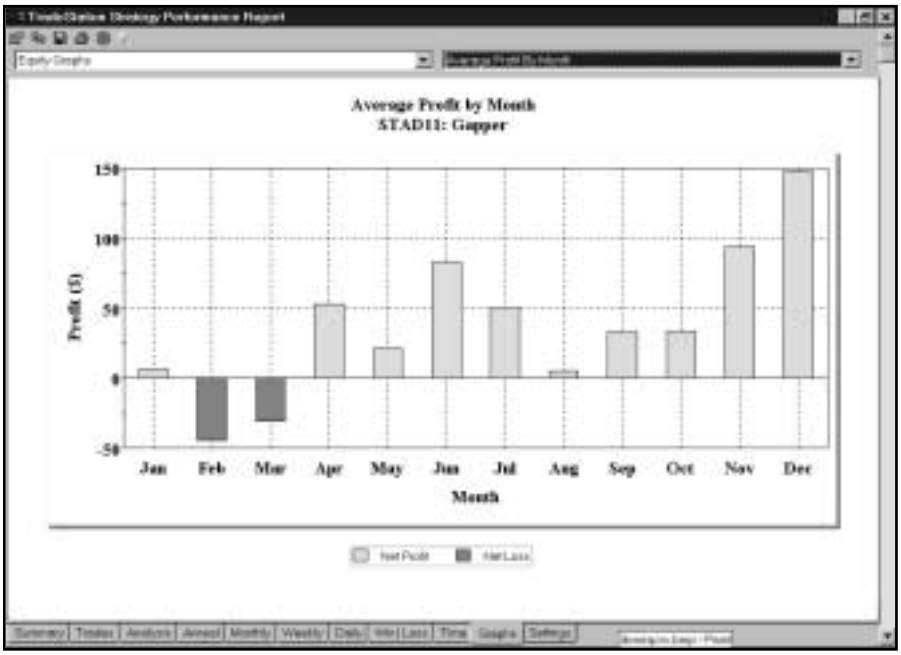



*Figure 4. IBM Average Profit by Month*

Next, let's see how Gapper fared on the British Pound. The optimized values are as follows:

ATR Protective Stop = 4

ATR Breakeven Stop = 4

ATR Trailing Stop = 5

ATR Big Profit Stop = 10

Exit Bars = 4

Gapper Length = 14

Figure 5 displays a fine trade on the short side of BP [Figure 5, BP chart]. The Performance Summary tells us that Gapper produced profits of $55,600 on 38 trades, with 47% profitable and an average winner 3.27 times the size of an average loser [Figure 6, BP Performance Summary]. The average trade (wins & losses) earned $1,463, and our strategy earned $2.94 for each $1.00 it lost.

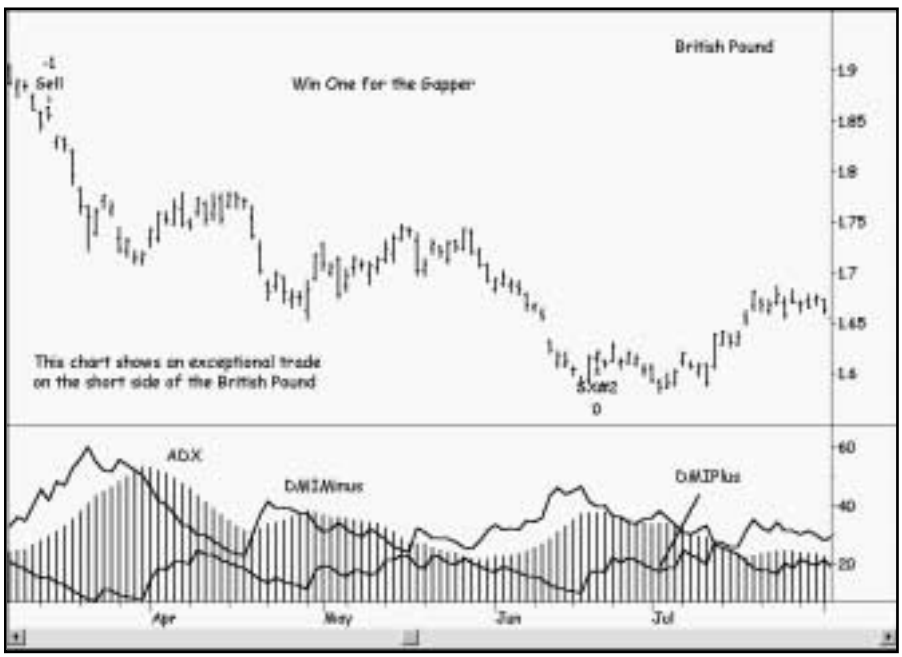



*Figure 5. BP chart*


TradeStation Strategy Performance Report - BTAD11: Gapper BP-CONT.Daily

**Performance Summary: All Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $55,600.00 | Open position P/L | $0.00 |
| Gross Profit | $94,250.00 | Gross Loss | |
| Total # of trades | 39 | Percent profitable | 47.37% |
| Number winning trades | 18 | Number losing trades | 20 |
| Largest winning trade | $14,812.50 | Largest losing trade | |
| Average winning trade | $4,680.56 | Average losing trade | |
| Ratio avg win/avg loss | 3.27 | Avg trade (win & loss) | $1,485.16 |
| Max consec. Winners | 4 | Max consec. losers | 5 |
| Avg # bars in winners | 48 | Avg # bars in losers | 27 |
| Max intraday drawdown | | | |
| Profit Factor | 2.94 | Max # contracts held | 1 |
| Account size required | $15,887.50 | Return on account | 348.99% |

**Performance Summary: Long Trades**

| | | | |
|---|---|---|---|
| Total Net Profit | $29,418.75 | Open position P/L | $0.00 |
| Gross Profit | $43,237.50 | Gross Loss | |
| Total # of trades | 22 | Percent profitable | 45.45% |
| Number winning trades | 10 | Number losing trades | 12 |
| Largest winning trade | $12,325.00 | Largest losing trade | |

Summary | Trades | Analysis | Annual | Monthly | Weekly | Daily | Win/Loss | Time | Graphs | Settings

*Figure 6. BP Performance Summary*

Finally, let's turn to Gapper and the Coffee market. The optimized values are as follows:

ATR Protective Stop = 3

ATR Breakeven Stop = 6

ATR Trailing Stop = 5

ATR Big Profit Stop = 14

Exit Bars = 1

Gapper Length = 18

Figure 7 shows an amazing Gapper trade in Coffee that earned $77,912 per contract in about five months [Figure 7, KC chart]. We'd have been very willing to pocket that profit, but there's something we need to check before we get too excited. In the Performance Summary [Figure 8, KC Performance Summary] note that the one huge winner accounted for $77,912 of the strategy's $89,712 total profit (87%). Although we want to see some big winners-especially in a trendfollowing strategy like this one-we don't want the strategy's ultimate success to be based on one grandslam homerun. Also, our strategy actually lost money ($15,850) in trading the short side of this market. Gapper needs some more work before we'd trade it with real money in the Coffee market!

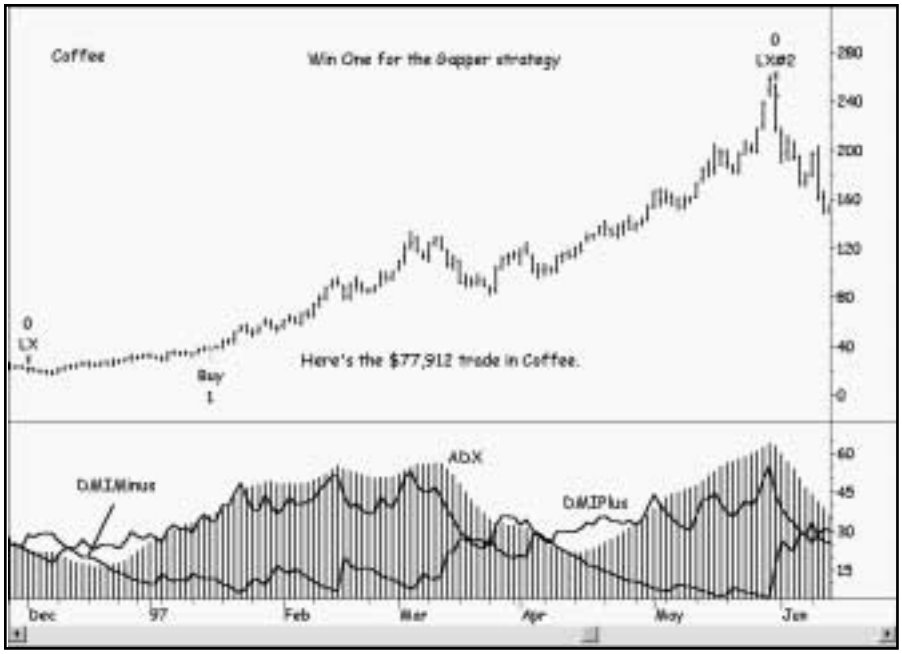



*Figure 7. KC chart*


*Figure 8. KC Performance Summary*

## Suggestions For Improvement

We can easily come up with five ideas to test that might improve Win One for the Gapper's results: first, optimize the length of the DMI indicator (we used the default value of 14); second, optimize ADX's minimum level for a setup (we chose 25 arbitrarily); third, optimize the number of bars in the statement ADX > ADX[1] (we only tested 1 bar ago); fourth, optimize a value that states how many points DMIPlus must be above DMIMinus (we just specified that it must be above); fifth, optimize a value for the number of points that prices have to rise above the previous bar's low for a trigger to buy and the number of points that prices have to fall below the previous bar's high for a trigger to sell short (we just bought at the previous low and sold at the previous high). With TradeStation 2000i, testing and optimizing the suggested values is quick and easy. Good luck as you work on improving the Gapper strategy!

APPENDIX A

# Common Exits

This section defines and explains the stops that are used more than once in the systems presented in this issue. We hope that you will find this single reference chapter to be more convenient than repeated descriptions of each stop throughout the volume.

## EasyLanguage Signal: ATR Big Profit Stop:

Applicable Systems in this issue:

- **STAD11: MISER**
- **STAD11: Win Gapper**

## Signal EasyLanguage:

```
Inputs: BigProfitATRs(7), ATRLength(10), ExitBarLen(3);
Variables: ATRVal(0), PosHL(0);

ATRVal = AvgTrueRange(ATRLength) * BigProfitATRs;

If BarsSinceEntry = 0 Then
        PosHL = Close;

If MarketPosition = 1 Then Begin
        If Close > PosHL Then
                PosHL = Close;
        If PosHL > EntryPrice + ATRVal Then
                ExitLong Next Bar at Lowest(Low, ExitBarLen) Stop;
End;


If MarketPosition = -1 Then Begin
        If Close < PosHL Then
                PosHL = Close;
        If PosHL < EntryPrice - ATRVal Then
                ExitShort Next Bar at Highest(High, ExitBarLen) Stop;
End;
```

## Signal Inputs (ATR Big Profit Stop)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| BigProfitATRs | 7 | Number of Average True Ranges used to determine the "Big Profit" level |
| ATRLength | 10 | Length, expressed in bars, used to calculate the Average True Range |
| ExitBarLen | 3 | Length, expressed in bars, used to determine the number of bars used in the trailing stop after the "Big Profit" level has been achieved |

## Signal Variables (ATR Big Profit Stop)

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| ATRVal | 0 | [Numeric] Holds the value of the Average True Range multiplied by the Big Profit amount |
| PosHL | 0 | [Numeric] Holds the value of the highest/lowest Close during the position |

### Setup

During the setup, the Average True Range is calculated and multiplied by the BigProfitATRs in order to determine the "Big Profit" level. On the bar of entry, when BarsSinceEntry is equal to 0, the position high/low variable is set to the Close of the entry bar.

```
ATRVal = AvgTrueRange(ATRLength) * BigProfitATRs;

If BarsSinceEntry = 0 Then
        PosHL = Close;
```

### Long Exit

When the market position is Long, the Long Exit becomes active. The PosHL variable is used to keep track of the highest Close of the position. If the PosHL (the highest Close of the position) exceeds the entry price plus the big profit amount (ATRVal), a Long Exit order is placed at the lowest Low of ExitBarLen bars.

```
If MarketPosition = 1 Then Begin
        If Close > PosHL Then
                PosHL = Close;
        If PosHL > EntryPrice + ATRVal Then
                ExitLong Next Bar at Lowest(Low, ExitBarLen) Stop;
End;
```

### Short Exit

When the market position is short, the Short Exit becomes active. The PosHL variable is used to keep track of the lowest Close of the position. If the PosHL (the lowest Close of the position) descends below the entry price minus the big profit amount (ATRVal), a Short Exit order is placed at the highest High of ExitBarLen bars.

```
If MarketPosition = -1 Then Begin
        If Close < PosHL Then
                PosHL = Close;
        If PosHL < EntryPrice - ATRVal Then
                ExitShort Next Bar at Highest(High, ExitBarLen) Stop;
End;
```

## EasyLanguage Signal: ATR Breakeven Stop:

Applicable Systems in this issue:

- **STAD11: High Five**
- **STAD11: MISER**
- **STAD11: TrendScore**
- **STAD11: Win Gapper**

## Signal EasyLanguage:

```
Inputs: ATRs(4), ATRLength(10);
Variable: ATRVal(0), PosHL(0);

ATRVal = AvgTrueRange(ATRLength) * ATRs;

If BarsSinceEntry = 0 Then
        PosHL = Close;

If MarketPosition = 1 Then Begin
        If Close > PosHL Then
                PosHL = Close;
        If PosHL > EntryPrice + ATRVal Then
                ExitLong ("1L") Next Bar at EntryPrice Stop;
End;

If MarketPosition = -1 Then Begin
        If Close < PosHL Then
                PosHL = Close;
        If PosHL < EntryPrice - ATRVal Then
                ExitShort ("1S") Next Bar at EntryPrice Stop;
End;
```

## Signal Inputs (ATR Breakeven Stop)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| ATRs | 4 | The floor value, the number of Average True Ranges above/below the Entry Price at which the Stop becomes active for the position |
| ATRLength | 10 | Length, expressed in bars, used to calculate the Average True Range |

## Signal Variables (ATR Breakeven Stop)

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| ATRVal | 0 | [Numeric] Holds the value of the Average True Range multiplied by the number of Trailing ATRs |
| PosHL | 0 | [Numeric] Holds the value of the highest/lowest Close of the position |

## Setup

In the Setup portion of the signal, the Average True Range is calculated and multiplied by the number of 'ATRs' specified in the Inputs.

```
ATRVal = AvgTrueRange(ATRLength) * ATRs;
```

On the first bar of the position, when the 'BarsSinceEntry' is equal to 0, the 'PosHL' variable is assigned the Close value. This resets the tracking of the position highest/lowest Close of the position, based on the direction of the position.

```
If BarsSinceEntry = 0 Then
        PosHL = Close;
```

## Long Exit

Once a Long position is taken, we must evaluate the highest closing price of the position and the Floor value established by the Average True Range. First, a comparison between the Close and the 'PosHL' Variable is made. During a Long position the 'PosHL' variable represents the highest Close of the position. Thus, if the Close is greater than the 'PosHL' value, the Close value is assigned to the 'PosHL' variable as the new highest Close. Next, If the highest Close of the position (PosHL) exceeds the sum of the 'EntryPrice' and the specified Average True Range (the Floor value), a Long Exit Stop order is placed at the entry price (breakeven price).

```
If MarketPosition = 1 Then Begin
        If Close > PosHL Then
                PosHL = Close;
        If PosHL > EntryPrice + ATRVal Then
                ExitLong ("1L") Next Bar at EntryPrice Stop;
End;
```

## Short Exit

Once a Short position is taken, we must evaluate the lowest closing price of the position and the Floor value established by the Average True Range. First, a comparison between the Close and the 'PosHL' Variable is made. During a Short position the 'PosHL' variable represents the lowest Close of the position. Thus, if the Close is less than the 'PosHL' value, the Close value is assigned to the 'PosHL' variable as the new lowest Close. Next, If the lowest Close of the position (PosHL) falls below the difference between the 'EntryPrice' and the specified Average True Range (the Floor value), a Short Exit Stop order is placed at the entry price (breakeven price).

```
If MarketPosition = -1 Then Begin
        If Close < PosHL Then
                PosHL = Close;
        If PosHL < EntryPrice - ATRVal Then
                ExitShort ("1S") Next Bar at EntryPrice Stop;
End;
```

## EasyLanguage Signal: ATR Protective Stop:

Applicable Systems in this issue:

- **STAD11: High Five**
- **STAD11: JK Night Train**
- **STAD11: MISER**
- **STAD11: The Bullseye**
- **STAD11: TrendScore**
- **STAD11: Win Gapper**

## Signal EasyLanguage:

```
Inputs: ProtectiveATRs(3), ATRLength(10);
Variable: ATRVal(0);

ATRVal = AvgTrueRange(ATRLength) * ProtectiveATRs;

If MarketPosition = 1 Then
        ExitLong Next Bar at EntryPrice - ATRVal Stop;

If MarketPosition = -1 Then
        ExitShort Next Bar at EntryPrice + ATRVal Stop;
```

## Signal Inputs (ATR Protective Stop):

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| ProtectiveATRs | 3 | The number of Average True Ranges that are risked in the position |
| ATRLength | 10 | Length, expressed in bars, used to calculate the Average True Range |

## Signal Variables (ATR Protective Stop):

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| ATRVal | 0 | [Numeric] Holds the value of the Average True Range, multiplied by the ProtectiveATRs |

## Setup

In the Setup portion of the signal, the Average True Range is calculated and multiplied by the number of ProtectiveATRs specified in the Inputs.

```
ATRVal = AvgTrueRange(ATRLength) * ProtectiveATRs;
```

## Long Exit

When the market position is Long, a Long Exit is placed at the entry price minus the Protective Volatility Average True Range calculation (ATRVal).

```
If MarketPosition = 1 Then
        ExitLong Next Bar at EntryPrice - ATRVal Stop;
```

## Short Exit

When the market position is Short, a Short Exit is placed at the entry price plus the Protective Volatility Average True Range calculation (ATRVal).

```
If MarketPosition = -1 Then
        ExitShort Next Bar at EntryPrice + ATRVal Stop;
```

## EasyLanguage Signal: ATR Trailing Stop:

Applicable Systems in this issue:

- **STAD11: TrendScore**
- **STAD11: Win Gapper**

## Signal EasyLanguage:

```
Inputs: TrailingATRs(4), ATRLength(10);
Variables: PosHigh(0), PosLow(0), ATRVal(0);

ATRVal = AvgTrueRange(ATRLength) * TrailingATRs;

If MarketPosition = 1 Then Begin
        If BarsSinceEntry = 0 Then
                PosHigh = High;
        If High > PosHigh Then
                PosHigh = High;
        ExitLong Next Bar at PosHigh - ATRVal Stop;
End;
```

```
If MarketPosition = -1 Then Begin
        If BarsSinceEntry = 0 Then
                PosLow = Low;
        If Low < PosLow Then
                PosLow = Low;
        ExitShort Next Bar at PosLow + ATRVal Stop;
End;
```

## Signal Inputs (ATR Trailing Stop)

| INPUT | DEFAULT | DESCRIPTION |
|---|---|---|
| TrailingATRs | 4 | The number of Average True Ranges that are risked from the highest/lowest price of the position |
| ATRLength | 10 | Length, expressed in bars, used to calculate the Average True Range |

## Signal Variables (ATR Trailing Stop)

| VARIABLE | DEFAULT | DESCRIPTION |
|---|---|---|
| PosHigh | 0 | [Numeric] Holds the value of the position High |
| PosLow | 0 | [Numeric] Holds the value of the position Low |
| ATRVal | 0 | [Numeric] Holds the value of the Average True Range multiplied by the number of TrailingATRs |

## Setup

In the Setup portion of the signal, the Average True Range is calculated and multiplied by the number of TrailingATRs specified in the Inputs.

```
ATRVal = AvgTrueRange(ATRLength) * TrailingATRs;
```

## Long Exit

When the market position is Long, the Long Exit becomes active. The PosHigh variable is used to keep track of the highest High of the position. The trailing Stop is placed at the PosHigh value minus the Trailing ATR calculation.

```
If MarketPosition = 1 Then Begin
        If BarsSinceEntry = 0 Then
                PosHigh = High;
        If High > PosHigh Then
                PosHigh = High;
        ExitLong Next Bar at PosHigh - ATRVal Stop;
End;
```

## Short Exit

When the market position is Short, the Short Exit becomes active. The PosLow variable is used to keep track of the lowest Low of the position. The trailing Stop is placed at the PosLow value plus the Trailing ATR calculation.

```
If MarketPosition = -1 Then Begin
        If BarsSinceEntry = 0 Then
                PosLow = Low;
        If Low < PosLow Then
                PosLow = Low;
        ExitShort Next Bar at PosLow + ATRVal Stop;
End;
```

## EasyLanguage Signal: Last Bar Exit:

Applicable Systems in this issue:

- **STAD11: High Five**
- **STAD11: JK Night Train**
- **STAD11: MISER**
- **STAD11: The Bullseye**
- **STAD11: Win Gapper**

## Signal EasyLanguage:

```
If LastBarOnChart Then Begin
ExitLong This Bar on Close;
        ExitShort This Bar on Close;
End;
```

***This Signal does not contain any Inputs or Variables.***

## Long/Short Exits

On the last bar of the chart, any Long or Short positions are closed out in order to insure that all trades are included in the System Report.

```
If LastBarOnChart Then Begin
        ExitLong This Bar on Close;
        ExitShort This Bar on Close;
End;
```

APPENDIX B

# Volume In Review

The EightBall strategy in STAD 10 contained an error. The ELS code on the CD uses the variation of the strategy that we described in Suggestions For Improvement — setting up a long entry when the fast moving average is declining and setting up a short entry when the fast moving average is rising. Since that variation is actually the one on the CD, the alternative you might want to test should be a buy setup when the fast moving average is rising and a sell setup when the fast moving average is declining.

**A STAD Club subscriber sent the following question:**

> Is it possible in StrategyBuilder to create a strategy whereby a trading signal is given only when all variables are in agreement?

**Our response:**

> It's not yet possible in StrategyBuilder to require three signals to be in agreement for an entry to be initiated. If you selected a moving-average crossover, a price-channel breakout, and a move to the parabolic stop as your entry signals in StrategyBuilder, TradeStation would signal an entry when just one of the three conditions was true. You can, however, create your strategy easily by writing three functions and requiring all three to be true for an entry to be signaled. Many of the functions you might wish to use are already written for you and can be accessed in TradeStation. Alternatively, you can just use the EasyLanguage PowerEditor to create your strategy the "old" way.
> Just include the word AND between the signals you write.

# INDEX