

OMEGA RESEARCH



VOLUME 8

Information in this document is subject to change without notice.

THE TRADING SYSTEMS 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 SYSTEM, AS THE USE OF ANY SUCH TRADING SYSTEM DOES NOT GUARANTEE THAT YOU WILL MAKE PROFITS, INCREASE PROFITS, OR MINIMIZE LOSSES. THE SOLE INTENDED USES OF THE TRADING SYSTEMS INCLUDED IN THIS BOOK ARE TO DEMONSTRATE THE WAYS IN WHICH EASYLANGUAGE CAN BE USED TO DESIGN PERSONAL TRADING SYSTEMS AND TO SHOW SOME EXAMPLES OF HOW CERTAIN POPULAR, WELL-KNOWN TRADING STRATEGIES MAY BE INCORPORATED INTO PERSONAL TRADING SYSTEMS. 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.

Copyright © 1999 Omega Research Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of Omega Research, Inc. Printed in the United States of America.

TradeStation® and SuperCharts® are registered trademarks of Omega Research, Inc. EasyLanguage, Portfolio Maximizer, PaintBar, ShowMe and SystemBuilder are trademarks of Omega Research, Inc. Microsoft is a registered trademark of Microsoft Corporation and MS-DOS, Windows, and Excel are trademarks of Microsoft Corporation. DBC Signal and BMI are trademarks of Data Broadcasting Corp. Price data supplied courtesy of Global Market Information, Inc.

Contents

INTRODUCTION	
Welcome to Volume 8.....	5
Chapter 1:	
SystemBuilder™: A New Approach to System Development and Testing.....	11
Chapter 2	
Escalator Trading System.....	19
Chapter 3	
New Forecasting Possibilities in TradeStation®2000i.....	33
Chapter 4	
Jack-in-the-Box System.....	45
Chapter 5	
ActivityBar™ Price Distribution System.....	61
Chapter 6	
Fixed Ratio Money Management by Ryan Jones.....	69
Chapter 7	
Ryan's Hope Trading System.....	77
Chapter 8	
Stampede Trading System.....	89
Chapter 9	
Common Exits.....	101
Appendix A	
Volume in Review.....	109
Index.....	112

INTRODUCTION

Welcome to Volume 8

Welcome to Volume 8 of the Omega Research System Trading and Development Club. This volume showcases three of the revolutionary new features of TradeStation 2000i: ActivityBars, ProbabilityMaps, and SystemBuilder. In addition, we're pleased to present a chapter on Fixed Ratio Money Management by guest author Ryan Jones, the creator of this new and already popular strategy.

STAD 8 also features five new trading systems. The Price Distribution System is based on ActivityBars, Omega Research's exciting new charting and analysis tool that enables you to "look inside the bar" to see important information that is invisible to your competition. We think you'll also enjoy and profit from four other systems presented in this volume: Escalator, Jack-in-the-Box, Ryan's Hope, and Stampede. These systems incorporate several traditional technical tools including channel breakouts, chart patterns, moving averages, pyramiding, and volatility.

Please note, we designed and tested all the systems in this volume using TradeStation 2000i, our new generation of software for investors and traders. However, with the exception of the Price Distribution ActivityBar System (which requires TradeStation 2000i), the other systems can be used in earlier versions when transferred from the included ELA file. The STAD 8 CD includes the systems' EasyLanguage in ELS format for use by TradeStation 2000i and in ELA format for TradeStation 4.0.

As you'll soon discover, STAD 8 presents both brand-new, cutting-edge trading technology and classic tools that have withstood the test of time. We recommend that you study the EasyLanguage code that makes it possible for us to test and optimize these systems. Also, please pay special attention to the many new performance statistics and graphs that are now available in TradeStation 2000i. Finally, don't regard any of our STAD Club trading systems as "carved in stone." We encourage you to combine, revise, add, delete, and tinker with these systems as much as you can. Use our examples as starting points to help you develop your own unique trading systems. Good luck and good trading!

IMPORTANT NOTICE: The trading systems in this book are examples only, and they have been included solely for educational purposes. Omega Research does not recommend that you use any such trading system, as the use of any such trading systems does not guarantee that you will make profits, increase profits, or minimize losses. The sole intended use of the trading systems included in this book are to demonstrate the ways in which EasyLanguage can be used to design personal trading systems and to show some examples of how certain popular, well-known trading strategies may be incorporated into personal trading systems.

Contents at a Glance

- Chapter 1: SystemBuilder: A New Approach to System Development and Testing
- Chapter 2: Escalator Trading System
- Chapter 3: New Forecasting Possibilities in TradeStation 2000i
- Chapter 4: Jack-in-the-Box Trading System
- Chapter 5: ActivityBar Price Distribution System
- Chapter 6: Fixed Ratio Money Management, by Ryan Jones
- Chapter 7: Ryan's Hope Trading System
- Chapter 8: Stampede Trading System
- Chapter 9: Common Exits
- Appendix A: Volume in Review
- Index

Additional Educational Services

Omega Research is committed to enhancing individual trading potential through quality education. To learn more about system 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 the products and technical analysis. Workshops are an excellent way to learn how to use the products, learn about technical analysis and system trading and/or 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 systems — 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 systems, transfer the analysis techniques into your TradeStation® library and then apply the system you want to review to a chart. Use the System Report to view the system results and take a look at the EasyLanguage instructions by opening the system in the PowerEditor™.

To transfer the analysis techniques into TradeStation:

1. Place the System Trading 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 **STAD8.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 systems in this volume all begin with STAD8 (although the signals will not have this prefix). You can now open the systems 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 systems and work with them, refer to this book for detailed explanations of the systems and the EasyLanguage used to create them. For instructions on applying systems and viewing the System Report, please refer to your *TradeStation User's Manual*.

***Note to SuperCharts® Users:** To transfer the systems into SuperCharts, use the Tools - QuickEditor menu sequence and select Transfer. Keep in mind, however, that although you can apply the systems in SuperCharts, you will not be able to view the EasyLanguage instructions in the QuickEditor. This is because the systems were designed in the PowerEditor. Also, if you are using SuperCharts End of Day, some of the systems will not apply as they are designed for intraday trading. Since the purpose of the 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 systems 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 systems 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 fax and is designed to help you troubleshoot an analysis technique or trading system you are currently working on. For example, if you are incorporating a trading system 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 system in the Club, why a system was developed a certain way, or why the system 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:

- 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 systems 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 System Trading

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

1. You'll have a system that is compatible with your own personality and trading style — a system 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 system 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 system and yourself, thus strengthening your ability to follow your system and to trade in a highly disciplined manner.

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

Getting Ideas For Systems

We can easily think of at least five great ways to get ideas for trading systems. 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 systems
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 (May, 1999, Caesars Palace, Las Vegas)
5. And, of course, Omega System Trading & 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 systems trading is more likely to generate consistent profits than discretionary trading, and once we have an idea for a trading system, how do we progress from an idea to a complete system?

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 system based on your idea. For example, you could write a system that enters a position based on your idea and exits the position automatically after n-days. Alternatively, you could write a stop-and-reverse system that uses your idea to enter, exit, and reverse positions.
3. Design a setup for your system. 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 system. 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 system. 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 system'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 system. For example, you might choose to test your system 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 system 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 system.
- 8.** Test and optimize your system 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 system 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 system to be tradable, the backward and forward tests should yield favorable results. Your system is unlikely to perform better in the future than it did on the out-of-sample data. Check your system'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 system with consistency, confidence, and courage.

CHAPTER 1:

SystemBuilder[™]: A New Approach to System Development and Testing

One of the most significant advances in ease of use of system design and testing in TradeStation 2000i is SystemBuilder. This new module of TradeStation not only lets you build strategies through a point and click interface, but also gives the previously 'hard-coded' stops all the power and flexibility behind EasyLanguage.

These are the two main goals that the TradeStation team had when designing SystemBuilder:

- First, to allow anyone to point and click through building any combination of entry and exit strategies — named Signals — through a point and click interface. Creating trading strategies is no longer full of repetitive and tedious groundwork, as SystemBuilder will let you reuse your favorite entry and exit signals written in EasyLanguage.
- Second, to enhance with the power of EasyLanguage all the stops that formerly were provided by TradeStation. There have been many innovative strategies to enter and exit positions, and there have been many requests for TradeStation to include them as built-in signals and stops. Thanks to SystemBuilder, TradeStation not only included many new exit strategies, but also has given all users the ability to modify the stops we provide and create their own custom stops. So, if you wanted that percentage-based profit target instead of the point-based, or a modified trailing stop, or any idea you need to try out, you will be able to add it using the power and flexibility of EasyLanguage.

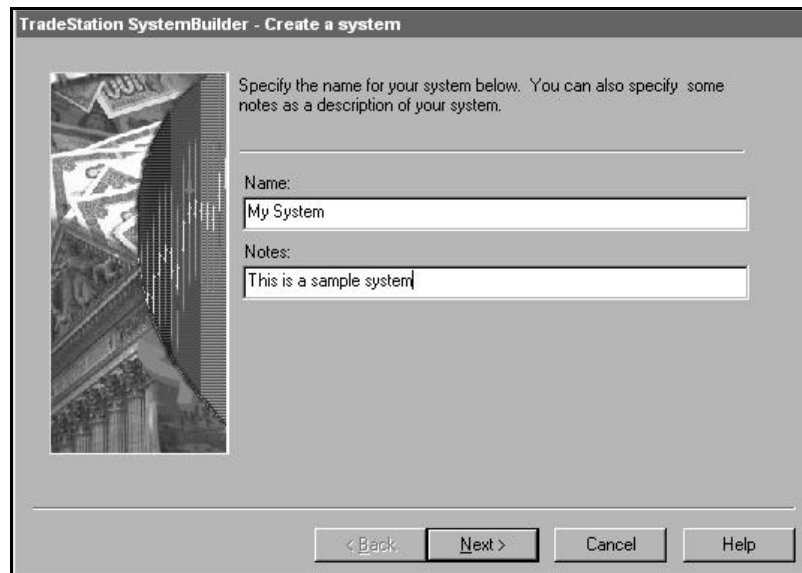
If you are used to or if you absolutely need to have your entry and exit criteria built in one document, TradeStation will also allow you to do this, and through SystemBuilder, will allow you to incorporate other different rules to enhance your trading strategies with minimal effort. In this chapter, we will see how SystemBuilder works.

Creating New Systems

Creating a new system is much easier than ever before. All you need to do is follow a six-step wizard that will guide you to create a new system. Here are the steps:

1. Create a system.

In this first step, you simply give SystemBuilder the name of the system you are creating, and you have space for notes in which you can type any information you deem relevant. These notes will be shown when you highlight the system in the main SystemBuilder window and when you edit the system.



TradeStation SystemBuilder - Create a system

Specify the name for your system below. You can also specify some notes as a description of your system.

Name:
My System

Notes:
This is a sample system

< Back Next > Cancel Help

2. Select signals.

In this step you will select all the signals that will constitute your system. The first thing you need to do is click on the ADD button.



TradeStation SystemBuilder - Select signals

Select the signals which make up your system from the list below. Use the Move Up and Move Down buttons to specify the order in which the signals will be evaluated.

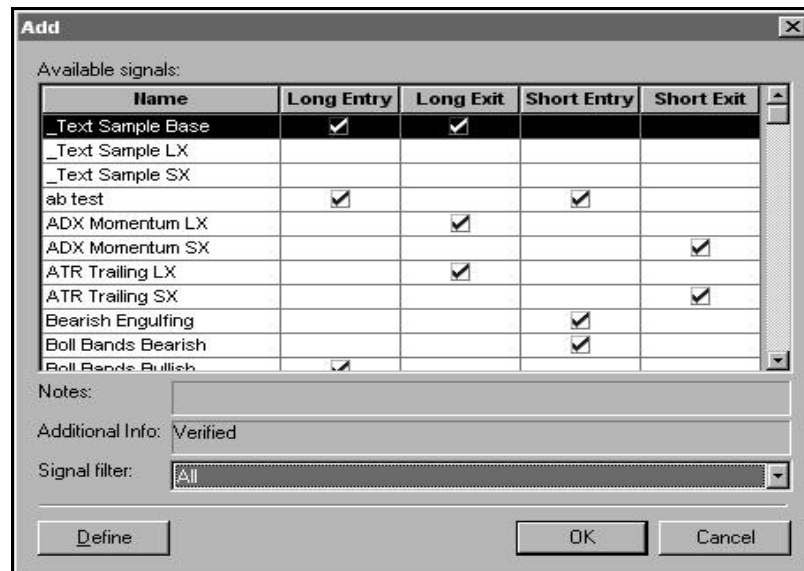
Signals included:

Name	Long Entry	Long Exit	Short Entry	Short Exit
------	------------	-----------	-------------	------------

Move Up Move Down Add Delete

< Back Next > Cancel Help

Then you will see a list of all the available signals. In this dialog you will see all the signals available in TradeStation, with four columns that will indicate if the signal has instructions to buy, sell short, or exit from long or short positions. At the bottom of this dialog you will have the filters, through which you will be able to only show entry orders, only show exit orders, or only show signals that have both entries and exits.



Additionally, by double clicking on any of the headers you will be able to sort the signals by that particular header. Once you highlight any signals you wish to include in the system, you can click on OK to include these signals into the system, and then you can continue on with the wizard.

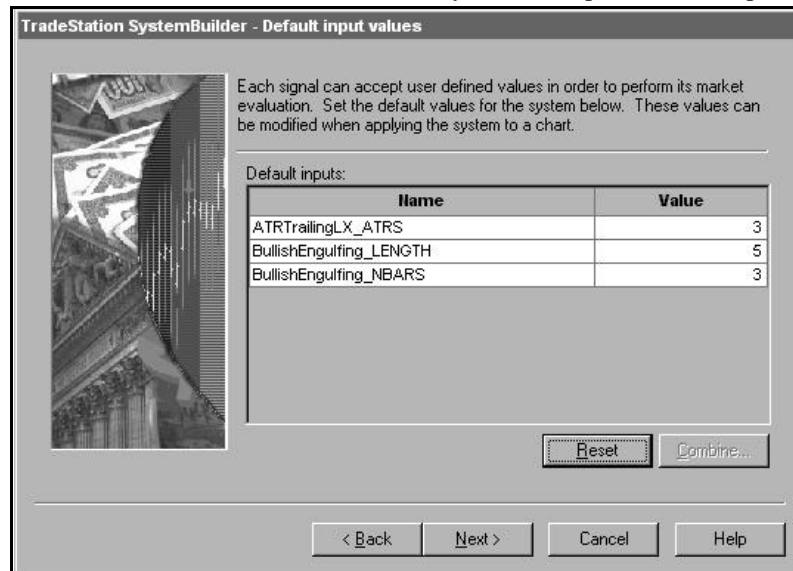
For this example, we will choose the long entry signal **Bullish Engulfing** and the long exit **ATR Trailing LX**. Once this is done your window should look like this:



3. Default input values.

In step 3, we will define the default values of the inputs for the system. Note that the inputs shown will correspond to the inputs of all the different signals you included in step 2. Because signals can have inputs that have the same name, the default names chosen by SystemBuilder will have the name of the signal, followed by an underscore (_) and then the name of the input. If you want, you can modify the name of the input by clicking and typing on the name.

In the current example you will have three inputs: **ATRTrailingLX_ATRS**, **BullishEngulfing_Length** and **BullishEngulfing_NBars**. As we mentioned before, you can simply click and rewrite the name or value of any of these inputs on this step.



TradeStation SystemBuilder - Default input values

Each signal can accept user defined values in order to perform its market evaluation. Set the default values for the system below. These values can be modified when applying the system to a chart.

Default inputs:

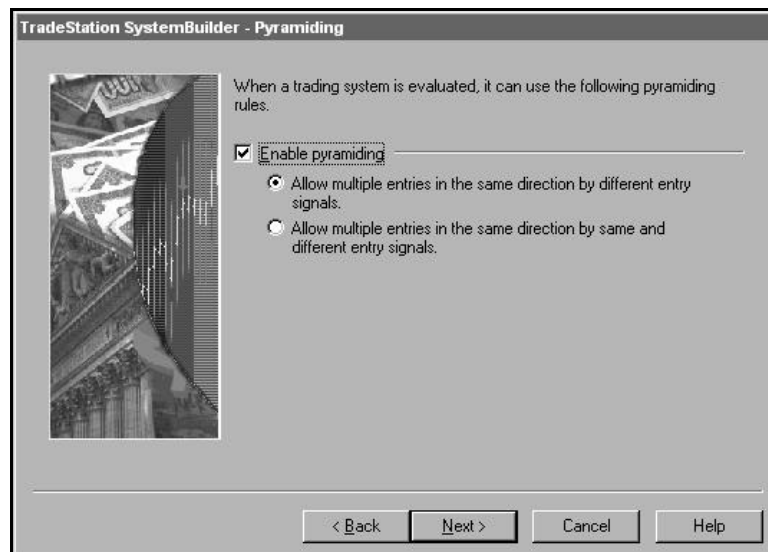
Name	Value
ATRTrailingLX_ATRS	3
BullishEngulfing_LENGTH	5
BullishEngulfing_NBars	3

Buttons: Reset, Combine, < Back, Next >, Cancel, Help

Note: If you have two inputs that should be defined as one (for example two inputs coming from different signals that represent the number of bars to be used for a moving average) then you can highlight both inputs and click on the combine button. This will effectively merge these two inputs into one. If you want to undo ALL your changes to the inputs (renames, default values and merging) you can click on the reset button.

4. Pyramiding.

In this step you will choose your system's default setting for pyramiding. Note that this is only the default value, and you will be able to change this setting when applying the system to a chart. For the current example we will leave the default settings.



TradeStation SystemBuilder - Pyramiding

When a trading system is evaluated, it can use the following pyramiding rules:

☒ Enable pyramiding

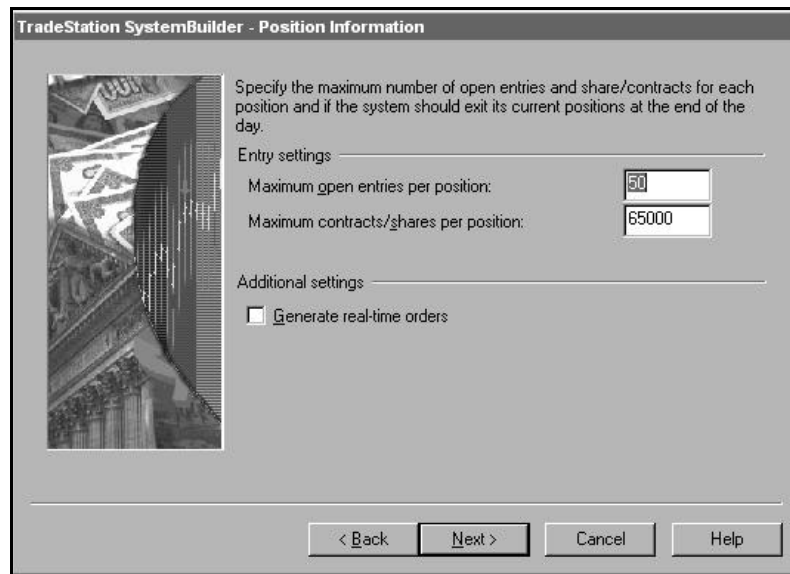
☒ Allow multiple entries in the same direction by different entry signals.

☐ Allow multiple entries in the same direction by same and different entry signals.

Buttons: < Back, Next >, Cancel, Help

5. Position information.

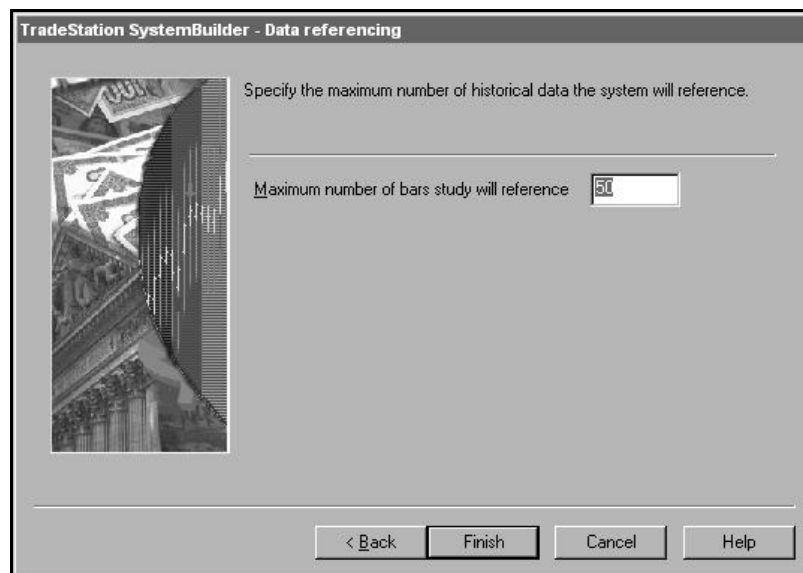
In this step you will select the details settings for the positions your system will take. For the current example we will leave the default settings.



The dialog box is titled "TradeStation SystemBuilder - Position Information". It features a background image of a classical building facade on the left. The main text area contains the instruction: "Specify the maximum number of open entries and share/contracts for each position and if the system should exit its current positions at the end of the day." Below this, there are two sections: "Entry settings" and "Additional settings". Under "Entry settings", there are two input fields: "Maximum open entries per position:" with a value of 50, and "Maximum contracts/shares per position:" with a value of 65000. Under "Additional settings", there is a checkbox labeled "Generate real-time orders" which is currently unchecked. At the bottom of the dialog, there are four buttons: "< Back", "Next >", "Cancel", and "Help".

6. Data referencing.

In this final step, you will specify the maximum number of bars your system will reference. For the current example we will leave the default settings.



The dialog box is titled "TradeStation SystemBuilder - Data referencing". It features a background image of a classical building facade on the left. The main text area contains the instruction: "Specify the maximum number of historical data the system will reference." Below this, there is a single input field labeled "Maximum number of bars study will reference" with a value of 50. At the bottom of the dialog, there are four buttons: "< Back", "Finish", "Cancel", and "Help".

And you are done! As you can see, you can create your systems very easily and with minimum effort. Additionally, these signals that we used (as well as all of the signals provided with TradeStation 2000i) can be used as additions to your own system. Now you can see how your favorite entry technique works with a variety of different exits.

This is another great benefit of SystemBuilder: it will allow you to test the effectiveness of entry and exit techniques separately with minimal work. You can very easily switch exits to see if your entries perform consistently, or try an exit with a variety of entries to see how it handles drawdown and how well it can capture profits. SystemBuilder will truly let you test the nuts and bolts of your strategies with incredible ease.

The 'old' style

As mentioned before, TradeStation will still allow you to create your systems the way you are used to in all prior versions of TradeStation. The PowerEditor will not limit you in any way, so you will be able to write your system as one signal and apply it to a chart after creating a system through SystemBuilder.

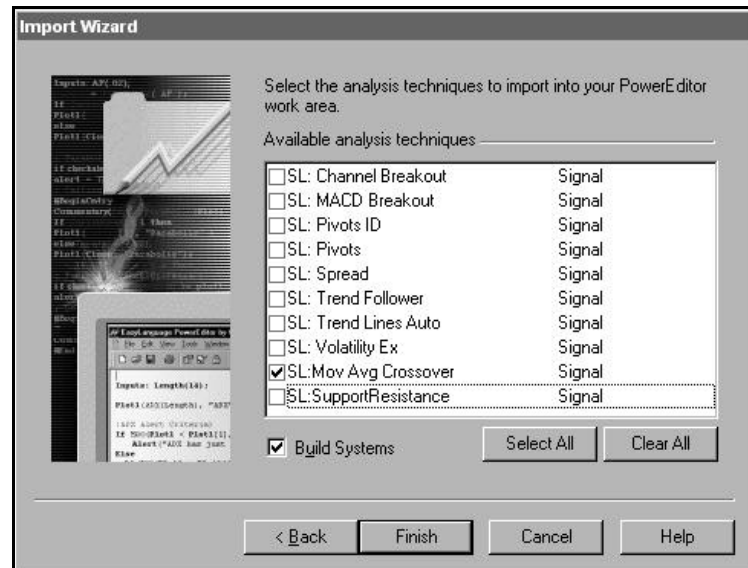
Once you have your trading rules in a signal in the PowerEditor, the additional step of creating a system through SystemBuilder will allow you to incorporate any other signals into this trading strategy through the SystemBuilder. Just as explained before, you will be able to add additional rules to your trading strategy through a point and click interface.

This can be demonstrated best through an example. Following we will show how to import the Moving Average Crossover system from STAD Club Volume 1.

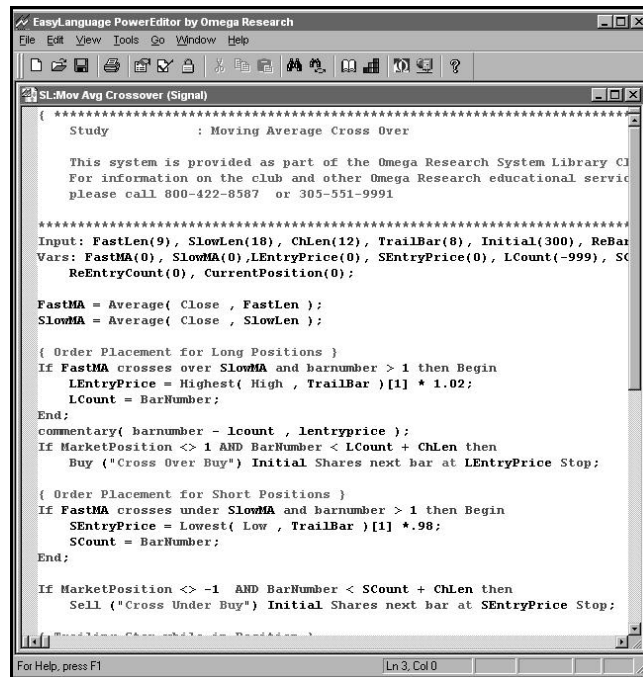
Importing your work from older versions of TradeStation

When you import systems from older versions of TradeStation, the Import/Export wizard of the PowerEditor will import all of your trading rules written in EasyLanguage into signals. You will have an option (on by default) to create the system automatically. As mentioned before, all the built-in stops that were available in prior versions are provided as signals, so any stops that were set in the systems will be added to the system automatically so you will have the exact same system ready to go.

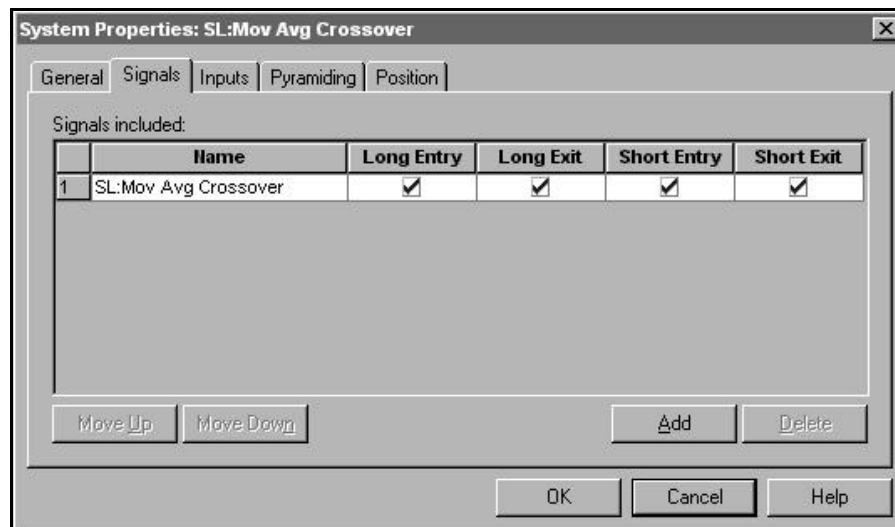
When importing your work created in prior versions of TradeStation, you will use the import/export wizard as usual, but when the dialog displaying the content of the .ela file is shown you will see that there is an option to create systems. This is shown in the following bitmap:



When the system is imported, the PowerEditor will create a signal using the same name your system had, and you will be able to review your EasyLanguage instructions by opening the signal on your PowerEditor. When we edit the Moving Average CrossOver signal we imported, we should see a window like this:



When you open SystemBuilder, you will see that you also have the system under the same name. When you choose to edit this system, you will see that it contains only one signal, which is the one we just edited. The fact that we have the system created and available through SystemBuilder will allow us to add any number of signals through a point and click interface. This is something you could only do before by retyping the instructions for the orders into the system itself, often running the risk of mistakenly altering the original EasyLanguage instructions.



If you want to add a signal, all you need to do is click on the add button of this dialog, highlight an additional entry or exit and click on OK. In the ADD dialog, let's choose a signal we used before. Choose the **Bullish Engulfing** entry and click on OK. It's that simple - you have added more instructions to your system. Adding different entry signals is equivalent to creating one signal that has the combined instructions of both signals, one immediately after the other. In other words, this system will now enter a position if either a moving average crossover occurs, OR if there is a Bullish Engulfing pattern.

THIS PAGE LEFT BLANK INTENTIONALLY

CHAPTER 2

Escalator Trading System

One of the most commonly used strategies for designing a trading system is to first identify the trend and then to look for a chart pattern that provides timely entries into the market in the direction of the trend. Escalator is an example of this kind of system. It uses two simple moving averages to define the trend and a two-bar pattern to time its buys and sells. We named this system Escalator because it's based on an up close/down close or down close/up close pattern, resembling side-by-side escalators with one moving up and the other moving down.

Let's begin with the entry signals. To buy, the close of the current bar must be above both a short-term and a long-term moving average. To sell short, the close must be below both moving averages. We'll use 8 as the default value for the short-term moving average and 40 as the default value for the long-term moving average (these values will be optimized, and the new values specified as we apply the system to a few markets). The chart pattern is simple: to buy, we look for the close of one bar ago to be in the bottom 25% of that bar's range and the close of the current bar to be in the top 25% of that bar's range. In other words, we're focusing on a weak-close/strong-close pattern as our setup to buy in an uptrend. To sell short, we look for the close of one bar ago to be in the top 25% of that bar's range and the close of the current bar to be in the bottom 25% of that bar's range. We focus on a strong-close/weak-close pattern to set up a sell in a downtrend. The two Escalator bars constitute our setup for an entry. The actual entry, in the case of a buy, is one point above the high of the two-bar pattern; the actual entry to sell short is one point below the low of the two-bar pattern. The setup for either a long entry or a short entry is in effect for only the bar immediately following the Escalator pattern. In other words, if the entry requirement isn't met on the bar after the setup, the setup is cancelled.

Now, let's add our initial protective stop. When we enter a long position, we'll set our protective stop, (our stop-loss order), at the low of the two-bar setup pattern minus one point. For a short position, our stop will be set at the high of the pattern plus one point. Therefore, our risk (on either a long or a short position) will be two points greater than the range of the Escalator setup pattern (plus, of course, any slippage we incur on the entry or exit).

Finally, let's add our exit strategy. For Escalator, the exit will be very simple. Rather than trailing a stop in an attempt to capture an occasional huge trade, we'll just set a reasonable profit target and strive for many consistent, "business-like" profits. Our objective will be a profit equal to two times the risk on the trade. In other words, if the risk from our entry price to the initial protective stop is \$500, we'll exit on a limit order when our open profit reaches \$1,000. For this system, we are going to limit the result of each trade to one of two possibilities: either we get stopped out at our initial protective stop (barring a gap or excessive slippage), or we take profits at our objective of two times the trade's initial risk. Of course, Escalator can easily be modified to include a breakeven stop and/or a trailing stop should you wish to do so.

Defining Our Trading Rules

In this system, we defined both long and short entries and exits. We also did some setup work to calculate the two moving averages and the range of each bar. The setup, entries, and exits are described next.

Setup

- a) Calculate an 8-bar and a 40-bar simple moving average.
- b) Calculate the range of each bar.

Long Entries

- a) The setup to buy is a close one bar ago in the bottom 25% of the range of that bar, followed by the close of the current bar in the top 25% of the current bar's range.
- b) The current bar's close must be above both the 8-bar and the 40-bar simple moving average of closes (remember that we'll optimize the moving average values for specific markets later in this chapter).
- c) On the next bar only, buy at the two-bar high plus one point.

Short Entries

- a) The setup to sell is a close one bar ago in the top 25% of the range of that bar, followed by the close of the current bar in the bottom 25% of the current bar's range.
- b) The current bar's close must be below both the 8-bar and the 40-bar simple moving average of closes.
- c) On the next bar only, sell at the two-bar low minus one point.

Long Exits

- a) Set a protective stop at the two-bar low minus one point.
- b) Place a limit order to exit at the profit target of two times the risk.

Short Exits

- a) Set a protective stop at the two-bar high plus one point.
- b) Place a limit order to exit at the profit target of two times the risk.

Designing & Formatting

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

Easy Language System Components:
Escalator (2-Bar Reversal) System (STAD8: Escalator)

System Inputs (Escalator):

INPUT	DEFAULT	DESCRIPTION
FastLength	8	Length, expressed in bars, used to calculate the short-term Moving Average.
SlowLength	40	Length, expressed in bars, used to calculate the long-term Moving Average
RiskLength	2	Length, expressed in bars, used to calculate the lowest Low upon which the risk is based

Signal Components:

1. 2-Bar Reversal Long

2. 2-Bar Reversal Short

EasyLanguage Signal: 2-Bar Reversal Long

Inputs: FastLength(8), SlowLength(40), RiskLength(2);

Variables: LongRisk(0);

Condition1 = Close[1] <= Low[1] + (.25 * Range[1]);

Condition2 = Close >= High - (.25 * Range);

Condition3 = Close > Average(Close, FastLength) AND Close > Average(Close, SlowLength);

If Condition1 AND Condition2 AND Condition3 AND MarketPosition <> 1 Then Begin

Buy Next Bar at Highest(High, 2) + 1 Point Stop;

LongRisk = Lowest(Low, RiskLength) - 1 Point;

End;

ExitLong ("LX1") Next Bar at LongRisk Stop;

If MarketPosition = 1 Then

ExitLong ("LX2") Next Bar at EntryPrice + (2 * LongRisk) Limit;

Signal Inputs (2-Bar Reversal Long):

INPUT	DEFAULT	DESCRIPTION
FastLength	8	Length, expressed in bars, used to calculate the short-term Moving Average.
SlowLength	40	Length, expressed in bars, used to calculate the long-term Moving Average
RiskLength	2	Length, expressed in bars, used to calculate the lowest Low upon which the Long side risk is based

In addition to the Input above, we define the following variables:

Signal Variables (2-Bar Reversal Long):

VARIABLE	DEFAULT	DESCRIPTION
LongRisk	0	Maintains the value of the low price upon which the risk is based.

Setup

Condition1 is used to see if the Close of one bar ago is in the bottom 25% of the range of one bar ago. Condition2 checks to see if the current close is in the top 25% of the current bar's range. Condition3 checks that the current Close is above both the fast and the slow moving averages.

```
Condition1 = Close[1] <= Low[1] + (.25 * Range[1]);
Condition2 = Close >= High - (.25 * Range);
Condition3 = Close > Average(Close, FastLength) AND Close > Average(Close, SlowLength);
```

Long Entry

If the three conditions described above are true, and we're not already in a long position (MarketPosition is not equal to 1), we'll place a Buy order one point above the two-bar high. We'll also establish a risk level based on one point below the lowest Low of RiskLength bars.

```
If Condition1 AND Condition2 AND Condition3 AND MarketPosition <> 1 Then Begin
    Buy Next Bar at Highest(High, 2) + 1 Point Stop;
    LongRisk = Lowest(Low, RiskLength) - 1 Point;
```

```
End;
```

Long Exit

An Exit order (LX1) is placed on each bar at the LongRisk value in the event that the entry order is filled. Once the order has been filled and the position is Long, a profit target (LX2) is placed at the entry price plus twice the LongRisk value.

```
ExitLong ("LX1") Next Bar at LongRisk Stop;
If MarketPosition = 1 Then
    ExitLong ("LX2") Next Bar at EntryPrice + (2 * LongRisk) Limit;
```

EasyLanguage Signal: 2-Bar Reversal Short

```
Inputs: FastLength(8), SlowLength(40), RiskLength(2);
Variables: ShortRisk(0);
```

```
Condition1 = Close[1] >= High[1] - (.25 * Range[1]);
Condition2 = Close <= Low + (.25 * Range);
Condition3 = Close < Average(Close, FastLength) AND Close < Average(Close, SlowLength);
```

```
If Condition1 AND Condition2 AND Condition3 AND MarketPosition <> -1 Then Begin
    Sell Next Bar at Lowest(Low, 2) - 1 Point Stop;
    ShortRisk = Highest(High, RiskLength) + 1 Point;
End;
```

```
ExitShort ("SX1") Next Bar at ShortRisk Stop;
If MarketPosition = -1 Then
    ExitShort ("SX2") Next Bar at EntryPrice - (2 * ShortRisk) Limit;
```

Signal Inputs (2-Bar Reversal Short):

INPUT	DEFAULT	DESCRIPTION
FastLength	8	Length, expressed in bars, used to calculate the short-term Moving Average.
SlowLength	40	Length, expressed in bars, used to calculate the long-term Moving Average
RiskLength	2	Length, expressed in bars, used to calculate highest High upon which the Short side risk is based

In addition to the Input above, we define the following variables:

Signal Variables (2-Bar Reversal Short):

VARIABLE	DEFAULT	DESCRIPTION
ShortRisk	0	Maintains the value of the high price upon which the risk is based.

Setup

Condition1 is used to see if the Close of one bar ago is in the top 25% of the range of one bar ago. Condition2 checks to see if the current close is in the bottom 25% of the current bar's range. Condition3 checks that the current Close is below both the fast and the slow moving averages.

```
Condition1 = Close[1] >= High[1] - (.25 * Range[1]);
```

```
Condition2 = Close <= Low + (.25 * Range);
```

```
Condition3 = Close < Average(Close, FastLength) AND Close < Average(Close, SlowLength);
```

Short Entry

If the three conditions described above are true, and we're not already in a short position (MarketPosition is not equal to -1), we'll place a Buy order one point below the two-bar low. We'll also establish a risk level based on one point above the highest High of RiskLength bars.

```
If Condition1 AND Condition2 AND Condition3 AND MarketPosition <> -1 Then Begin
```

```
    Sell Next Bar at Lowest(Low, 2) - 1 Point Stop;
```

```
    ShortRisk = Highest(High, RiskLength) + 1 Point;
```

```
End;
```

Short Exit

An Exit order (SX1) is placed on each bar at the ShortRisk value in the event that the entry order is filled. Once the order has been filled and the position is Short, a profit target (SX2) is placed at the entry price plus twice the LongRisk value..

```
ExitShort ("SX1") Next Bar at ShortRisk Stop;
```

```
If MarketPosition = 1 Then
```

```
    ExitShort ("SX2") Next Bar at EntryPrice - (2 * ShortRisk) Limit;
```

Testing & Improving

We tested this system on U.S. Treasury Bonds (US) and General Motors (GM). Following are the optimized values for both markets:

US:

Long

FastLength = 12

SlowLength = 20

RiskLength = 4

Short

FastLength = 4

SlowLength = 20

RiskLength = 5

GM:

Long (not tested on the short side)

FastLength = 12

SlowLength = 30

RiskLength = 5

First, let's look at Escalator's performance on Bonds. We tested our system on daily data from January, 1984, to March, 1999, and included a deduction of \$40 for slippage and \$10 for commission. The TradeStation System Report [Figure 1, TradeStation System Report] shows a total net profit of \$61,472 on 98 trades. 45% of the trades were profitable, and the average trade earned \$627. The average winner was 2.08 times as large as the average loser (ratio avg win/avg loss), and the system earned \$1.69 for every dollar it lost (profit factor). The Analysis section of the System Report [Figure 2, Analysis] reveals that although the Sharpe Ratio, Return Retracement Ratio, and K-Ratio were not up to standard, the Rina Index, the Net Profit/Largest Loss ratio, and the Net Profit/Maximum Drawdown ratio were fine. Note that Escalator produced no positive or negative outliers. We're not surprised, because the system includes a relatively conservative initial protective stop and exits at a fixed multiple of the initial risk. As we planned, there were no huge losing trades or huge winning trades in this system.

TradeStation System Report			
TradeStation System Report - ESCALATOR US.LNG-Daily			
Performance Summary: All Trades			
Total Net Profit	\$61,472.00	Open position P/L	\$0.00
Gross Profit	\$150,014.00	Gross Loss	(\$88,542.00)
Total # of trades	98	Percent profitable	45.00%
Number winning trades	44	Number losing trades	54
Largest winning trade	\$7,356.00	Largest losing trade	(\$5,114.00)
Average winning trade	\$3,409.41	Average losing trade	(\$1,639.67)
Ratio avg win/avg loss	2.08	Avg trade (win & loss)	\$627.27
Max consec. Winners	4	Max consec. losers	11
Avg # bars in winners	23	Avg # bars in losers	12
Max intraday drawdown	(\$21,560.00)		
Profit Factor	1.69	Max # contracts held	1
Account size required	\$24,260.00	Return on account	253.39%
Performance Summary: Long Trades			
Total Net Profit	\$58,156.00	Open position P/L	\$0.00
Gross Profit	\$93,858.00	Gross Loss	(\$35,702.00)
Total # of trades	53	Percent profitable	53.00%
Number winning trades	28	Number losing trades	25
Largest winning trade	\$7,356.00	Largest losing trade	(\$2,957.00)
Average winning trade	\$3,409.41	Average losing trade	(\$1,430.88)
Ratio avg win/avg loss	2.38		
Max consec. Winners	4	Max consec. losers	11
Avg # bars in winners	23	Avg # bars in losers	12
Max intraday drawdown	(\$21,560.00)		
Profit Factor	1.69	Max # contracts held	1
Account size required	\$24,260.00	Return on account	253.39%
Summary	Trades	Analysis	Annual
Monthly	Weekly	Daily	Win/Loss
Time	Graphs	Settings	

Figure 1. TradeStation System Report

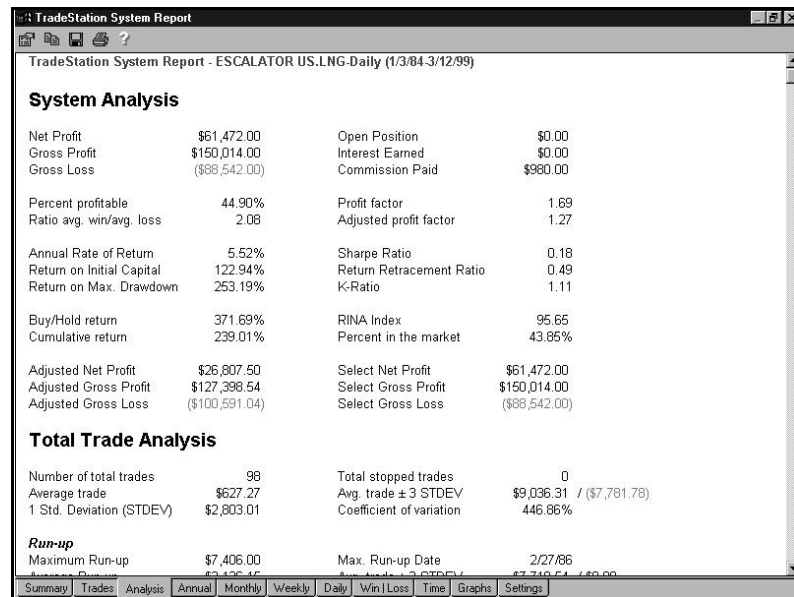


Figure 2. Analysis

The next section of the System Report that we'll focus on is the Annual Trading Summary [Figure 3, *Annual*]. This report includes the system's annual performance on both a year-by-year mark-to-market basis and on an annual rolling period mark-to-market basis. The former displays the trading results one year at a time, while the latter displays the results from the beginning of each year of the test period to the end of the test period (for example, 98 to 99, 97 to 99, 96 to 99, and so on, back to 84 to 99). The important information here is that although six individual years resulted in a loss, all of the rolling periods were profitable. For example, 1996 showed a loss of \$1,251, but 1996 to 1999 showed a profit of \$21,113.

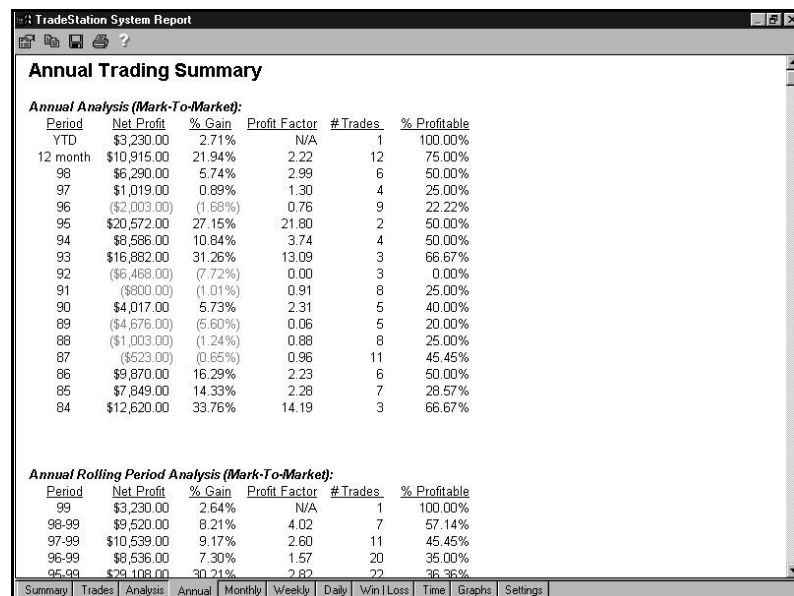


Figure 3. Annual

The Time Analysis section of the system report provides some good news and some bad news [Figure 4, Time]. The good news is that the system let profits run on winners for an average of 32 bars, while it cut losses short on losers by exiting in half as many bars. The bad news is that 395 days is a long time to wait between equity peaks. To improve that statistic, we'd want to trade more than one system on this market or apply this system to more than just one market. Diversification in both systems and markets really does tend to smooth out the equity curve.

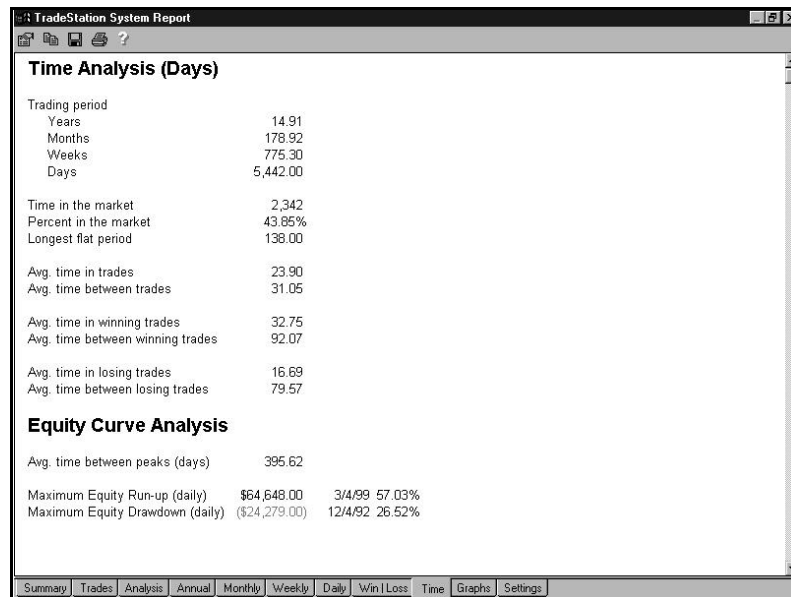


Figure 4. Time

Let's see if we can learn more about our system by studying some of the System Report's graphs. A profitable but not very smooth equity curve is the subject of our first graph [Figure 5, Equity Curve]. The system performed well in the first third of the trades, performed poorly in the middle third, and performed extremely well in the final third. The Underwater Equity Curve [Figure 6, Underwater Equity Curve] shows dramatically how poorly the system did in the middle period of our test, making no new equity peaks from 1989 to 1995, and experiencing a 25% drawdown on the initial account size that we set at \$50,000. The graph of Average Profit by Month [Figure 7, Average Profit by Month] also tells us that the system was not very consistent. Trading results for the average May, July, and October were negative, and the average December was only marginally profitable.

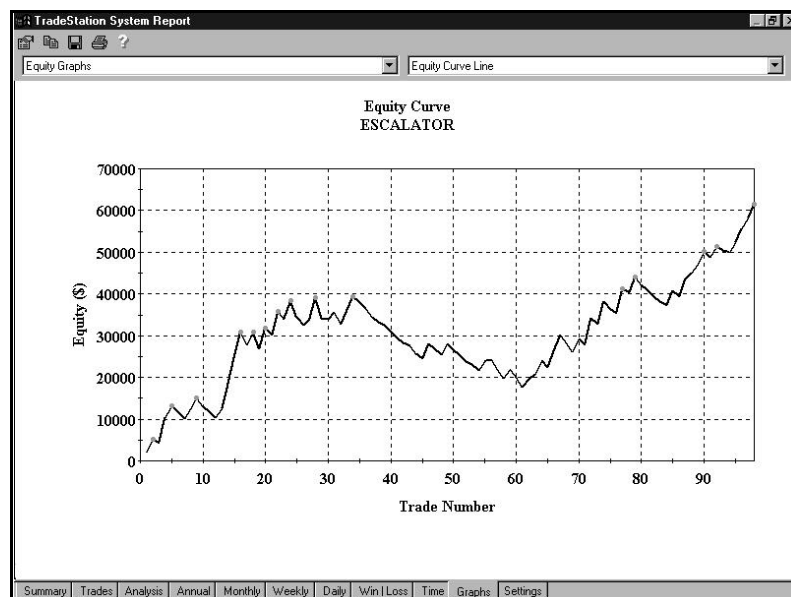


Figure 5. Equity Curve

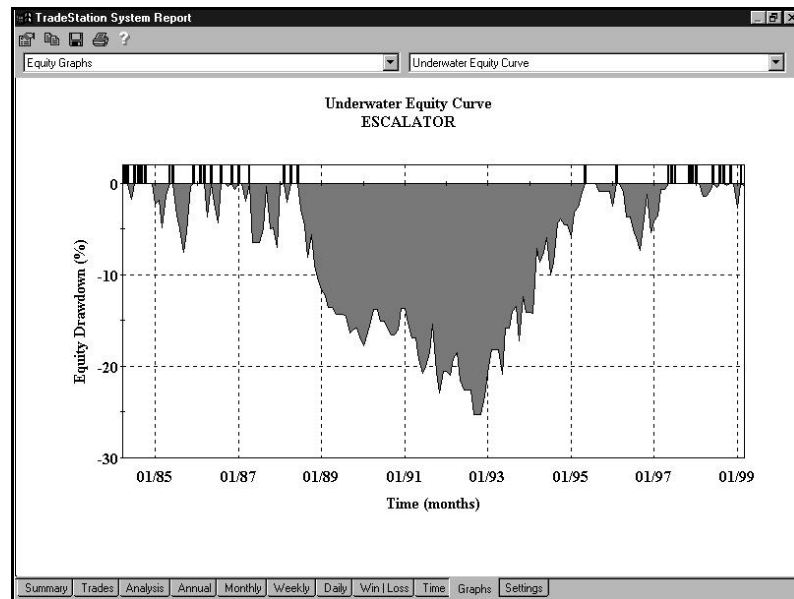


Figure 6. Underwater Equity Curve

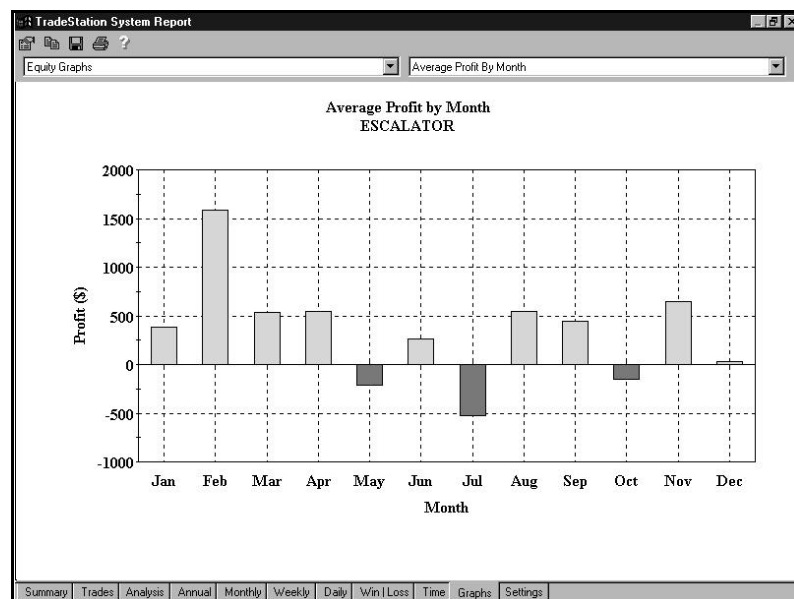


Figure 7. Average Profit by Month

Maximum Adverse Excursion [Figure 8, MAE] and Maximum Favorable Excursion [Figure 9, MFE] are the last two graphs we'll look at for our test on Bonds. MAE shows us that only one winning trade experienced a drawdown of more than \$3,000, so we wouldn't want to risk more than \$3,000 on a trade. MFE tells us that once a trade is profitable by about \$3,200, it not only stays a winner but also earns even more money. Adding to your position when open profits reach \$3,200 might be a good investment.

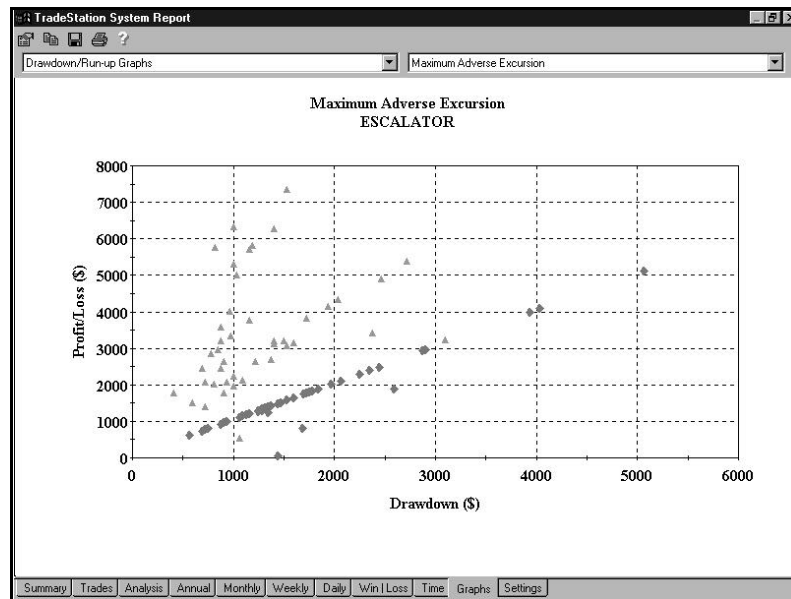


Figure 8. Maximum Adverse Excursion

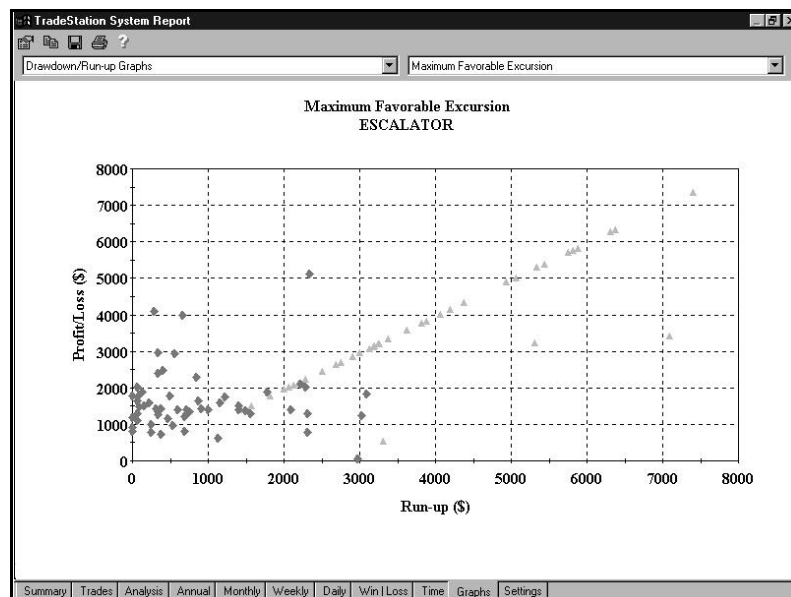


Figure 9. Maximum Favorable Excursion

In addition to Bonds, we tested Escalator on General Motors (GM) from January, 1984, to March, 1999, deducting \$.13 per share for slippage and \$.10 per share for commission. We based our test on a fixed purchase of 100 shares per trade and limited our testing to the long side, since few investors take long-term short positions in US stocks. The bar chart [Figure 10, (GM) General Mtrs Corp] is a good example of Escalator working as we intended. In the uptrend that began in late September, we found a weak-close/strong-close pattern and bought the next bar on the open, which gapped above the two-bar high. Since the input for RiskLength for GM is 5, we set our protective stop one point below the lowest low of the five bars before our entry. Eventually, GM reached our profit target of two times the risk, and we exited just three bars before the high. Notice that if we had been setting a reasonable trailing stop, the market would have stopped us out in December (if not sooner) with a much smaller profit.



Figure 10. (GM) General Motors Corp.

The TradeStation System Report [Figure 11, Summary] tells us that Escalator generated profits of \$3,662 (per 100 shares) on 47 trades, with 45% of the trades profitable, and an average trade of \$77.93 (after deducting for slippage and commission). The average win was 1.93 times as large as the average loss, and the system earned \$1.56 for each dollar it lost.

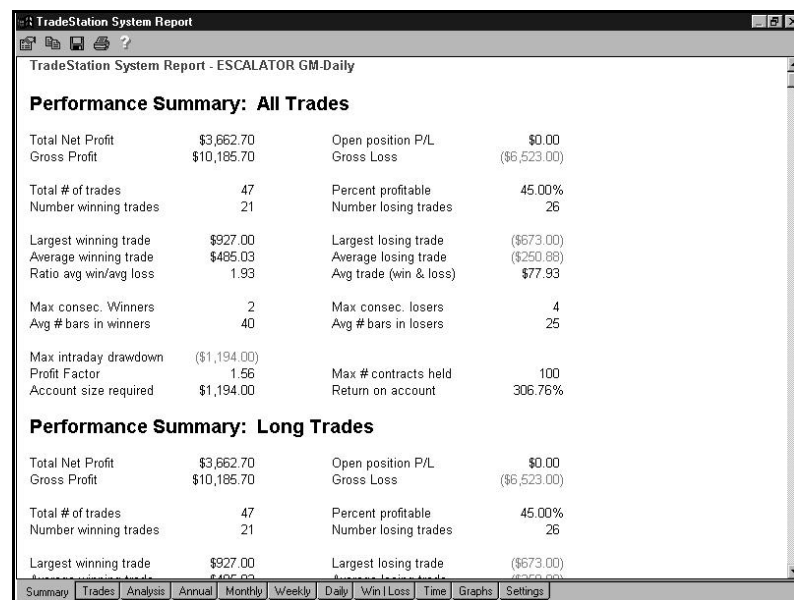


Figure 11. Summary

The Equity Curve [Figure 12, Equity Curve] shows that the system survived several early losses, climbed back to even at trade 19, and really took off around trade 29. Monthly Rolling Net Profit [Figure 13, MRNP] can be compared to a "snapshot" of the system's performance taken at the end of each month. When GM was trading sideways to down, the system (which is "buy only") struggled to stay even, but when the stock began to perform well, our system did likewise.

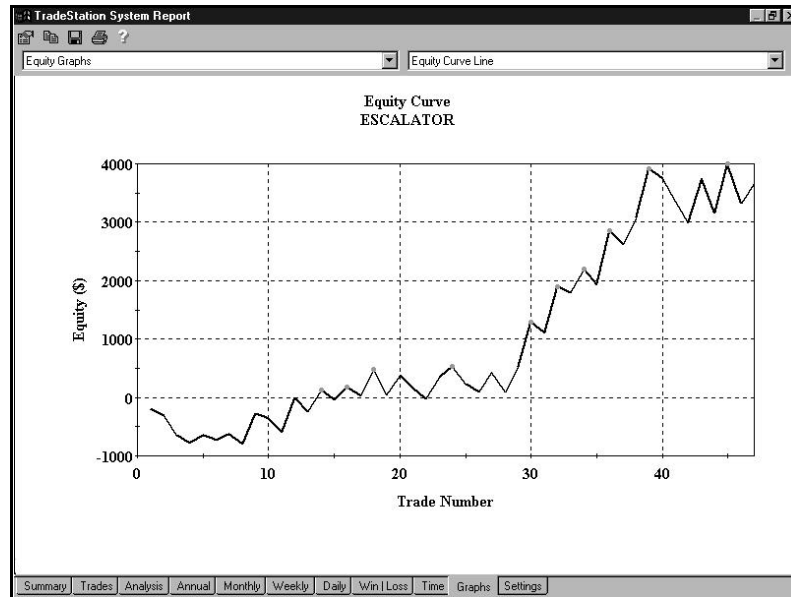


Figure 12. Equity Curve

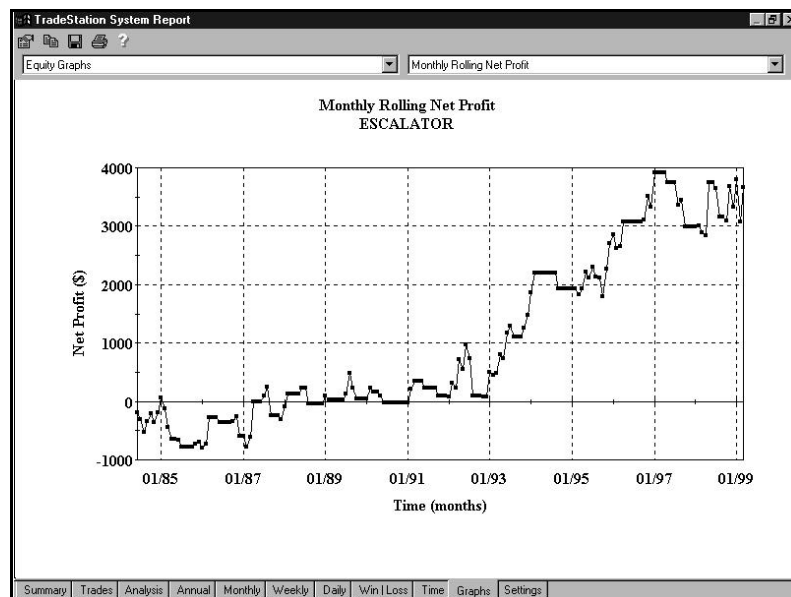


Figure 13. Monthly Rolling Net Profit

Maximum Adverse Excursion [Figure 14, MAE] reveals that only one trade (an eventual \$300 winner) was closed out with a profit after having suffered a drawdown of \$300 or more. This suggests that a \$300 money-management stop might be useful when the initial risk from entry to a point below the five-bar low is greater than \$300 (per 100 shares). Finally, Maximum Favorable Excursion [Figure 15, MFE] shows that only four trades turned into losers after posting open profits of \$225 or more, versus 13 trades that accrued more profits after reaching the \$225 threshold.

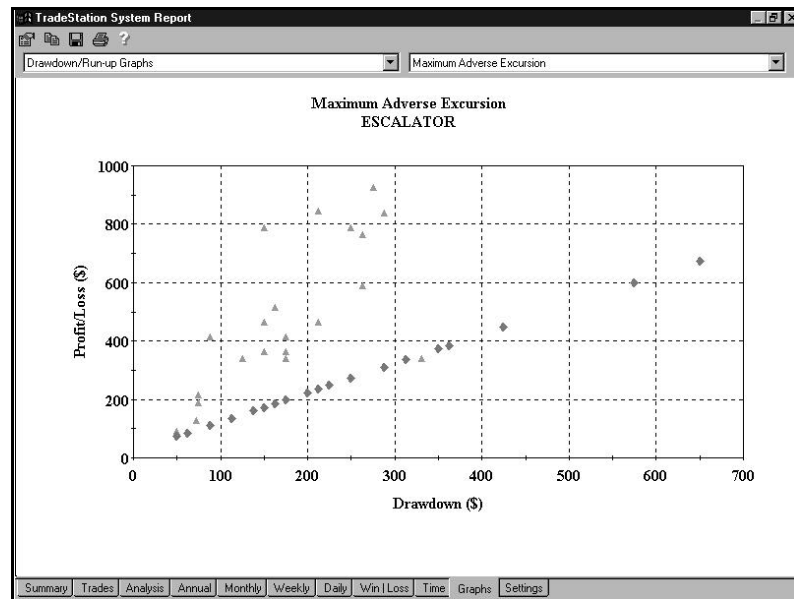


Figure 14. Maximum Adverse Excursion

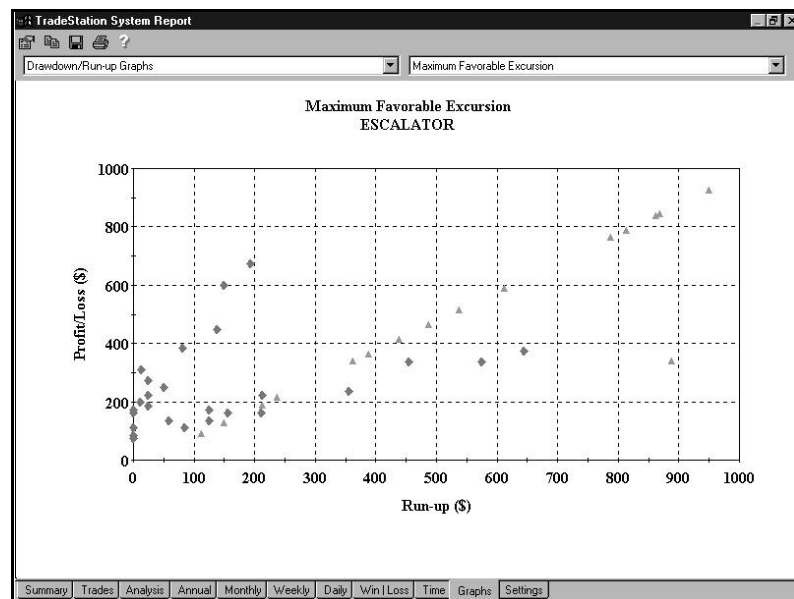


Figure 15. Maximum Favorable Excursion

Suggestions for Improvement

We can think of three possible ways to improve this system. First, consider adding the requirement that the strong-close bar (in the case of an uptrend) have heavier volume and/or a greater range than the weak-close bar that preceded it. Second, consider adding a breakeven stop to minimize risk once a trade is profitable by a reasonable amount (e.g., the entry price +1 times the initial risk). Third, consider adding a very loose trailing stop until prices reach the profit objective and then tightening the stop dramatically to lock in profits while still allowing the trade to continue its trend.

THIS PAGE LEFT BLANK INTENTIONALLY

CHAPTER 3

New Forecasting Possibilities in TradeStation® 2000i

In TradeStation 2000i we have provided a whole new arena for the development of indicators and analysis techniques by allowing you to not only have charts with space to the right of the most current price, but by allowing you to create indicators and ProbabilityMaps through EasyLanguage that will use this "space to the right" to show any type of information.

Working with Indicators

The easiest point from which to begin our exploration of this new "space to the right" feature, is by stating the most obvious use of this feature when using indicators. Those of you who are familiar with displacing plot statements will know you can displace plots of any indicator in both directions, backwards and forward. In order to plot a moving average of the closes, you can write the following indicator:

```
Inputs: Price(Close), Length(10);  
Plot1(Average(Price, Length) , "Plot1");
```

Now if you want to displace this indicator to the right (plot today the value of the indicator 10 days ago), you would have needed to write

```
Inputs: Price(Close), Length(10), Displace(10);  
Plot1(Average(Price, Length)[Displace], "Plot1");
```

This would have calculated a 10 day moving average of 10 bars ago, and displayed the result on the current bar. Doing this on every bar would have the effect of displacing the average 10 bars to the right. The limitation we had previously was that we were not able to finish the plot, so we could only show values up to the most recent bar.

In TradeStation 2000i we now have the ability to plot on a future bar. You can do this by displacing that plot statement using a negative value in between brackets:

```
Inputs: Price(Close), Length(10), Displace(-10);  
Plot1[Displace](Average(Price, Length), "Plot1");
```

While both indicators shown are valid and return very similar results, you will see that when these indicators are plotted on a chart, the second version (displacing the plot) will continue to draw the indicator into the space to the right of the chart. Please note that in order to displace the plot to the right, you must have the "Bars to Right" option in TradeStation set to a value that is equivalent or greater than the displacement value.

What you are telling TradeStation is to plot 10 bars forward (into the future) the value that you have in the plot statement. Note that — 10 would mean to plot 10 bars into the future, while 10 would mean to plot 10 bars back.

Forecasting

In addition to displacing plot statements, we can also find the last bar of the chart, and extend a regular indicator so it shows forecasting figures. This is probably a more interesting use of this new feature.

In order to do this, you first need to find the last bar of the chart. You will be able to do this by using the LastBarOnChart function provided with EasyLanguage. Then you will have to tell TradeStation what to plot on every bar thereafter.

Let's start with a simple example. Let's calculate the average net change of the last five bars, and add it to the last plot value on each consecutive bar thereafter. This will produce a line that will show us where the indicator would go if it were to continue moving as it has on the last five bars. To create this example, we could use the following EasyLanguage:

```
Inputs: Price(Close), Length(10), BarsToRight(10);
Variables: AvgNetChange(0), Counter(0);
```

```
Plot1(Average(Price, Length), "Plot1");
```

```
AvgNetChange = Average(Plot1 - Plot1[1], 10);
```

```
If LastBarOnChart Then Begin
    value1 = Plot1 + AvgNetChange;
    For Counter = -1 DownTo -BarsToRight Begin
        Plot1[counter](value1, "Plot1");
        value1 = value1 + AvgNetChange;
    End;
End;
```

This indicator calculates the average net change of the moving average, then adds this value to the last value of the indicator for each bar for which we want to extend the plot to the right. The resulting forecast line will be a straight line extending from the Indicator showing the direction it had on the last few bars.

Next, we can take this forecasting example a step further by measuring the acceleration of the moving average net change. Thus, we need to calculate how much the net change was increasing or decreasing on a bar by bar basis. This is calculated by taking the average net change of the average net change. In EasyLanguage it would be written as follows:

```
AvgNetChange = Average(Plot1 - Plot1[1], 10);
Acceleration = Average( AvgNetChange - AvgNetChange[1], 5);
```

Because we want the acceleration to be more sensitive, we use a shorter term average. Next, on every iteration of the loop where we draw the forecasting values of the indicator, we will add the acceleration to the average net change of each bar. This will give us an idea of how much the indicator is accelerating (or decelerating) on every bar. The revised EasyLanguage would be written as follows:

```

Inputs: Price(Close), Length(10), BarsToRight(10);
Variables: AvgNetChange(0), Counter(0);
Variables: Acceleration(0), tPlotValue(0), tAvgNetChng(0);

Plot1(Average(Price, Length), "Plot1");

AvgNetChange = Average(Plot1 - Plot1[1], 10);
Acceleration = Average(AvgNetChange - AvgNetChange[1], 5);

If LastBarOnChart Then Begin
    tAvgNetChng = AvgNetChange;
    tPlotValue = Plot1 + tAvgNetChng;
    For Counter = -1 DownTo -BarsToRight Begin
        Plot1[Counter](tPlotValue, "Plot1");
        tPlotValue = tPlotValue + tAvgNetChng;
        tAvgNetChng = tAvgNetChng + Acceleration;
    End;
End;

```

This technique can be used with any indicator by substituting the value plotted (in the fourth line of the indicator) with the particular formula for the indicator you wish to project into the future.

ProbabilityMaps

ProbabilityMaps, now available in TradeStation 2000i, are a completely new breed of indicators specifically designed to be used as forecasting tools. This new type of indicator will plot information, derived from EasyLanguage, to the right of the chart in new ways not previously possible using standard indicators.

Technically speaking, what are ProbabilityMaps?

A ProbabilityMap is a new type of indicator that will provide you with a rectangular area, that we will call the ProbabilityMap area, divided into smaller areas called cells. You can color each one of these cells a specific color by assigning a numerical value to each cell, where 0 is black and 100 is white. The ProbabilityMap area is defined by bars (x axis) and points (y axis).

When applied to a chart, the ProbabilityMap will initially be plotted to the right of the last bar on the chart. Using the ProbabilityMap drawing tool, you can select the bar from which the ProbabilityMap will be calculated and drawn. This area will be shown to the right of that specific bar.

Lets get started

When a document is created with the ProbabilityMap template provided with TradeStation's PowerEditor, the following text is inserted into the document:

```
{ *****
Written by:
Description:
***** }
Input: Columns(100);
Vars: bottom(0), top(0), counter(0), RowHt(0), price(0);

{ The ProbabilityMap will paint a rectangular area to the right of the selected bar with colors
that are mapped to numbers from 0 to 100, usually 0 will be black and 100 will be white. This
rectangular area is defined by a number of columns or bars, and upper and lower bounds }
{ Set the number of columns with the input Columns }

PM_SetNumColumns(Columns);

{ TODO 1 of 4: Replace <value> with the calculations for the upper bounds of the
ProbabilityMap }
bottom = value;

{ TODO 2 of 4: Replace <value> with the calculation for the lower bounds of the
ProbabilityMap }
top = value;

PM_SetHigh(bottom);
PM_SetLow(top);

{ TODO 3 of 4: Set the height of each position -in points-. This is equivalent to the resolution
of each point of the activitybar }
RowHt = value;
PM_SetRowHeight(RowHt);

{ Filling in the ProbabilityMap values }
For counter = 1 to Columns Begin
    price = bottom;
    while price < top Begin

        { TODO 4 of 4: Substitute <value> with the Probability value for position
(counter, price), this value should be a number between 1 and 100 }
        value1 = value;

        PM_SetCellValue(counter, price, value1) ;
        price = price + RowHt ;
    End ;
End ;
```

The first question that needs to be addressed is how many bars you want to show in the ProbabilityMap. The ProbabilityMap study will draw starting from either the last bar on the chart or from the bar at which the mouse pointer was clicked on, and going X number of bars to the right (or into the future). We will tell the ProbabilityMap how many bars extend to the right using the command:

```
PM_SetNumColumns(num)
```

Where num is a positive integer number. In this example, we will let the user determine this by providing an input called Columns. This does always need to be set by an Input. The ProbabilityMap could be written in such a way so as to define the number of columns or bars into the future that it will use on a bar by bar basis. Using the Input, the command would be revised as follows:

```
PM_SetNumColumns(columns);
```

Second, we need to specify what the the upper and lower bounds of the ProbabilityMap area will be. These bounds can also be set on a bar by bar basis, depending on the needs of the ProbabilityMap. For this example we will use a 5 bar average true range above and below the Close of the specified bar. These values will be represented by the variables Bottom and Top, so we would write:

```
Bottom = Close - AvgTrueRange(5);
Top = Close + AvgTrueRange(5);
```

Then these values would be assigned to the ProbabilityMap using the reserved words PM_SetHigh and PM_SetLow as follows:

```
PM_SetHigh(top);
PM_SetLow(bottom);
```

We will continue to work with this later in this article.

Next, we need to specify the height of each row. As explained before, the ProbabilityMap area is similar to a grid that is divided into columns and rows. The columns represent the number of bars into 'the future' that the ProbabilityMap will plot. A row is defined as the number of divisions from the top to the bottom. The height of these cells can be defined using the keyword PM_SetRowHeight(RowHt). You can specify the height of the rows on a bar by bar basis, so the height can change depending on the price of the symbol being analyzed. If you are plotting a ProbabilityMap on a stock that trades around \$10/share, you will want a smaller row height than if you are plotting the same ProbabilityMap on the Dow, which is in the thousands.

One idea for defining the row height is to calculate the average true range of the bars, and divide this number by the approximate number of cells you want the grid to contain. Thus, if you want approximately 50 cells in the grid, you could use:

```
PM_SetRowHeight(AvgTrueRange(5) / 50);
```

The next step is to fill in the values for the ProbabilityMap. We will do this using two nested loops, a For loop to go through all the columns, and a While loop to go through the Y axis prices filling the ProbabilityMap values for each price. For this exercise we will use the Function TLValueEasy(). This Function returns the trend line regression value for a certain number of bars into the future.

Lets start by simply drawing the values of the TLValueEasy() function on the chart. For this we would write:

```
For counter = 1 To Columns Begin
    value1 = TLValueEasy(Close, 1, 10, -Counter);
    PM_SetCellValue(Counter, value1, 100) ;
End ;
```

If you verify the lines we have so far and apply the ProbabilityMap to the chart, you will probably receive a Run Time Error stating that the value you assigned to the ProbabilityMap is too low for the range specified. What this means is that you tried to paint a cell that was out of the range of the ProbabilityMap area. If, for example, your ProbabilityMap had a high value of 100 and a low value of 80, and you write the command PM_SetCellValue(1, 75, 100), you would get this error because the command was trying to paint a cell located at the price 75, but the bottom bound was set to 80. In order to avoid this you would write:

```
For Counter = 1 To Columns Begin
    value1 = TLValueEasy(Close, 1, 10, -counter);
    If value1 > Bottom ANDDD value < Top Then
        PM_SetCellValue(Counter, value1, 100) ;
End ;
```

Once this line has been added, we should have enough instructions to verify this ProbabilityMap and successfully add it to a chart. In order to review the procedure thus far, the instructions for the ProbabilityMap should appear as shown below (excluding any comments):

```
Input: Columns(100);
Vars: Bottom(0), Top(0), Counter(0), RowHt(0), Price(0);

    PM_SetNumColumns(Columns);

Bottom = Close - AvgTrueRange(5);
Top = Close + AvgTrueRange(5);
PM_SetHigh(Top);
PM_SetLow(Bottom);

PM_SetRowHeight(AvgTrueRange(5) / 50);

For counter = 1 To Columns Begin
    value1 = TLValueEasy(Close, 1, 10, -Counter);
    If value1 > Bottom AND value1 < Top Then
        PM_SetCellValue(Counter, value1,100) ;
End;
```

From this point forward, feel free to verify the ProbabilityMap and apply it to a chart on any step.

The first thing you might notice is that the ProbabilityMap drawing of the trend line regression technique quickly disappears (before the 50 bars). If this happens, the reason is that the regression values went either over the upper or under the lower bounds of the ProbabilityMap. In order to improve this, you can adjust the bands by multiplying the average true range by a different factor (4, for instance). Thus, you would have:

```
Bottom = Close - 4 * AvgTrueRange(5);
Top = Close + 4 * AvgTrueRange(5);
PM_SetHigh(Top);
PM_SetLow(Bottom);
```

You will then see that the ProbabilityMap is much 'taller', thus allowing you to plot more values within the grid.

Let's now add additional colors to the ProbabilityMap. Let's say that we will paint one average true range above and below the regression value, transitioning in color from white (regression value) to black (average true range value). Remember that when painting ProbabilityMaps 100 represents white and 0 represents black. To perform the colorization, the EasyLanguage would be revised as follows:

```
For counter = 1 To Columns Begin
    value1 = TLValueEasy(Close, 1, 10, -Counter);
    If value1 > Bottom AND value1 < Top Then
        PM_SetCellValue(Counter, value1, 100) ;

    Price = value1;
    While Price < Top Begin
        Color = MaxList(100 - ((Price - value1) * 100 / AvgTrueRange(5)), 0);
        If Price < Top AND Price > Bottom Then
            PM_SetCellValue(Counter, Price, Color);
        Price = Price + PM_GetRowheight;
    End;
End;
```

Let's break-down these instructions to understand them better:

The While loop will start at the regression line (the value of the variable Price at the beginning of the loop) and increment from that value until it reaches the value of the Top boundary of the ProbabilityMap. The instructions that do this are:

```
Price = value1;
While Price < Top Begin

    Price = Price + PM_GetRowheight;
End;
```

At each iteration of this loop, the current space will be painted with the appropriate color. In order to do this we will need to calculate the color that should be used for each price. The following formula is used to calculate the color:

```
Color = MaxList(100 - ((Price - value1) * 100 / AvgTrueRange(5)), 0);
```

Let's break this formula down. We mentioned before that the probability will be drawn from the trend line regression value up to 1 average true range. To do this we have to calculate the distance between the close and the average true range as a percentage. This would be done using the following formula:

```
(Price - value1) * 100 / AvgTrueRange(5)
```

This will give us a value from 0 to 100. Zero (0) will represent when the variable price is the same as the linear regression value, and 100 will represent when the price equivalent to the Linear Regression value plus one average true range. This is a step in the right direction, but we actually want the opposite result: we want the linear regression value to be white (100) and the average true range to be black (0). In order to accomplish this, we will subtract the resulting value from 100. Thus, the formula will look like this:

```
100 - (Price - value1) * 100 / AvgTrueRange(5)
```

The price we were looking for may actually be greater than the average true range, so this formula may result in a negative value. To this problem, we use the MaxList() Function to return the higher value between the formula return and 0. The resulting formula would be:

```
Color = MaxList(100- ((Price - value1) * 100 / AvgTrueRange(5)), 0);
```

Let's see an example: suppose that the regression value (variable value1) is 100, the average true range is 10, and we were interested in finding the color we would use for a price of 1010. Let's substitute the values:

```
Color = MaxList(100 - ((101 - 100) * 100 / 10), 0)
```

Which is the same as:

```
Color = MaxList(100 - ( 100 / 10) , 0)
```

Resolving further to:

```
Color = MaxList( 10, 0)
```

The value that gets assigned to Color would be 10. Feel free to assign any number greater than 100 to this formula to see what color would be chosen for that particular price. Now we need to check that the calculation falls within the defined range.

The same will be done with the lower side of the regression values. Here is the resulting EasyLanguage statements for the ProbabilityMap:

```
value1 = TLValueEasy(Close, 0, 10, -Counter);
```

```
If value1 < Top AND value1 > Bottom Then
    PM_SetCellValue(Counter, value1, 100);
Price = value1;
```

```
While Price < Top Begin
    Color = MaxList(100- ((Price - value1) * 100 / AvgTrueRange(5)), 0);
    If Price < Top AND Price > Bottom Then
        PM_SetCellValue(Counter, Price, Color);
    Price = Price + PM_GetRowHeight;
End;
Price = value1;
```

```
While Price > Bottom Begin
    Color = MaxList(100- ((value1-Price) * 100 / AvgTrueRange(5)), 0);
    If Price < Top AND Price > Bottom Then
        PM_SetCellValue(Counter, Price, Color);
    Price = Price - PM_GetRowHeight;
End;
```


The prediction of this formula becomes less and less accurate with each bar that goes on. To compensate for this we want to decrease the probabilities used for each bar that occurs. To do this we will need to use a variable counter, which is the variable that keeps count of the bar that is being painted. Let's look at the following formula:

```
MaxList((100 - counter) / 100, 0)
```

This formula will subtract from 100 the bar we are at, and it will divide the resulting value by 100. So if we are at bar 0 (the current bar) we will perform the following calculation, which is equal to 1:

```
(100 - 0) / 100
```

If we are at bar 50, we will have $(100 - 50) / 100$, which will return 0.5. We will use either 0 or the result of this calculation, whichever is greater. So what are we going to do with this formula? We will multiply the color by this formula in order to have the effect of reducing the probability on every bar. Here are the resulting instructions:

```
value1 = TLValueEasy(Close, 0, 10, -Counter);
If value1 < Top AND value1 > Bottom Then
    PM_SetCellValue(Counter, value1, 100);
price = value1;

While price < Top begin
    color = MaxList(100- ((price - value1)*100 / AvgTrueRange(5)),0) *
    MaxList((100 - Counter) / 100,0);
    If price < Top AND Price > Bottom Then
        PM_SetCellValue(Counter, Price, Color);
    Price = Price + PM_GetRowHeight;
End;
Price = value1;

While Price > Bottom Begin
    Color = MaxList(100 - ((value1 - Price) * 100 / AvgTrueRange(5)), 0) *
    MaxList((100 - Counter) / 100, 0);
    If Price < Top AND Price > Bottom Then
        PM_SetCellValue(Counter, Price, Color);
    Price = Price - PM_GetRowHeight;
End;
```

So the final ProbabilityMap should read:

```
Input: Columns(100);
Vars: BotTom(0), Top(0), Counter(0), RowHt(0), Price(0), Color(0);

PM_SetNumColumns(columns);

BotTom = Close - 4 * AvgTrueRange(5);
Top = Close + 4 * AvgTrueRange(5);
PM_SetHigh(Top);
PM_SetLow(BotTom);

PM_SetRowHeight(AvgTrueRange(5) / 50);
```

```

For Counter = 1 To Columns Begin
    value1 = TLValueEasy(Close, 0, 10, -Counter);
    If value1 < Top AND value1 > BotTom Then
        PM_SetCellValue(Counter, value1, 100);
    Price = value1;
    While Price < Top Begin
        Color = MaxList(100- ((Price - value1) * 100 /
        AvgTrueRange(5)),0)*MaxList((100-Counter)/100,0);
        If Price < Top AND Price > BotTom Then
            PM_SetCellValue(Counter, Price, Color);
        Price = Price + PM_GetRowHeight;
    End;
    Price = value1;
    While Price > BotTom Begin
        Color = MaxList(100- ((value1-Price)*100 /
        AvgTrueRange(5)),0)*MaxList((100-Counter)/100,0);
        If Price < Top AND Price > BotTom Then
            PM_SetCellValue(Counter, Price, Color);
        Price = Price - PM_GetRowHeight;
    End;
End;

```

Additional notes: You may notice that the ProbabilityMap is very slow when applied to a chart. One of the things that can be done to improve speed is to replace all the references of the function AvgTrueRange(5) with a variable. This is true not only for ProbabilityMaps but also for any other EasyLanguage analysis technique you create. The variable we will use is AvgTR. The following is the modified EasyLanguage:

```

Input: Columns(100);
Vars: BotTom(0), Top(0), Counter(0), RowHt(0), Price(0), Color(0), AvgTR(0);

PM_SetNumColumns(Columns);
AvgTR = AvgTrueRange(5);

BotTom = Close - 4 * AvgTR;
Top = Close + 4 * AvgTR;
PM_SetHigh(Top);
PM_SetLow(BotTom);

PM_SetRowHeight(AvgTR / 50);

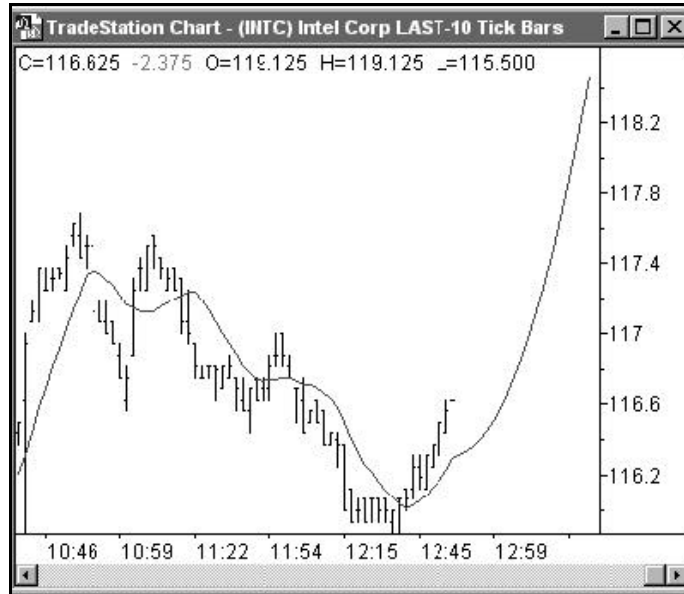
For Counter = 1 To Columns Begin
    value1 = TLValueEasy(Close, 0, 10, -Counter);
    If value1 < Top AND value1 > BotTom Then
        PM_SetCellValue(Counter, value1, 100);
    Price = value1;
    While Price < Top Begin
        Color = MaxList(100- ((Price - value1) * 100 /
        AvgTR),0)*MaxList((100-Counter)/100,0);
        If Price < Top AND Price > BotTom Then
            PM_SetCellValue(Counter, Price, Color);
        Price = Price + PM_GetRowHeight;
    End;

```

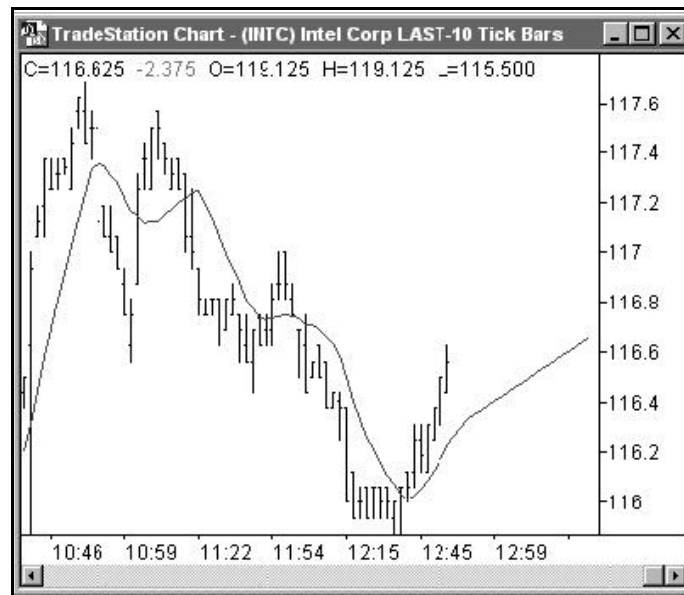
```

Price = value1;
While Price > BotTom Begin
    Color = MaxList(100- ((value1-Price)*100 /
    AvgTR),0)*MaxList((100-Counter)/100,0);
    If Price < Top AND Price > BotTom Then
        PM_SetCellValue(Counter, Price, Color);
        Price = Price - PM_GetRowHeight;
    End;
End;

```



Final example of Indicator



Initial example of Indicator plotting to the right

THIS PAGE LEFT BLANK INTENTIONALLY

CHAPTER 4

Jack-in-the-Box System

Jack-in-the-Box gets its name from a simple pattern that indicates a new uptrend. If a market has been in a downtrend, we look for a bar that posts a low above the high of the lowest bar in the downtrend. Although we may be guilty of overactive imaginations, this pattern seems like a jack-in-the-box popping up after being compressed down in the box.

Thus, our setup to buy is a low that is greater than the high of the bar that makes the lowest low of a downtrend; our setup to sell short is a high that is less than the low of the bar that makes the highest high of an uptrend. [*Figure 1, New Uptrend and New Downtrend*] This setup is very sensitive to potential changes in trend. It has the advantage of allowing us to enter new trends early, but it also has the corresponding disadvantage of suffering a large number of "whipsaw" losses (i.e. small losses that are incurred when the original trend reasserts itself).

Our entry requirement serves to filter out many of the potential "whipsaw" losses. After a buy setup, we'll go long at the open plus $n\%$ of the 10-bar average true range (ATR); after a sell setup, we'll go short at the open minus $n\%$ of the ATR. In both cases, the entry must be triggered within a specified number of bars after the setup, or the setup is cancelled.

When we enter a new position, we'll apply a version of the parabolic indicator as our initial stop and our trailing stop. Our parabolic is identical to the parabolic indicator developed by Welles Wilder, with one exception. In a new uptrend, Wilder's parabolic begins at the lowest low of the previous downtrend; in a new downtrend, Wilder's parabolic begins at the highest high of the previous uptrend. In our version of the indicator, we begin the parabolic for an uptrend at a specified number of ATRs below the bar of our entry into the trade. In a downtrend, we begin the parabolic at a specified number of ATRs above the bar on which we entered a short position. We believe that this placement of the parabolic gives us more control of the indicator and therefore more control of the initial risk on the trade than the conventional parabolic provides.

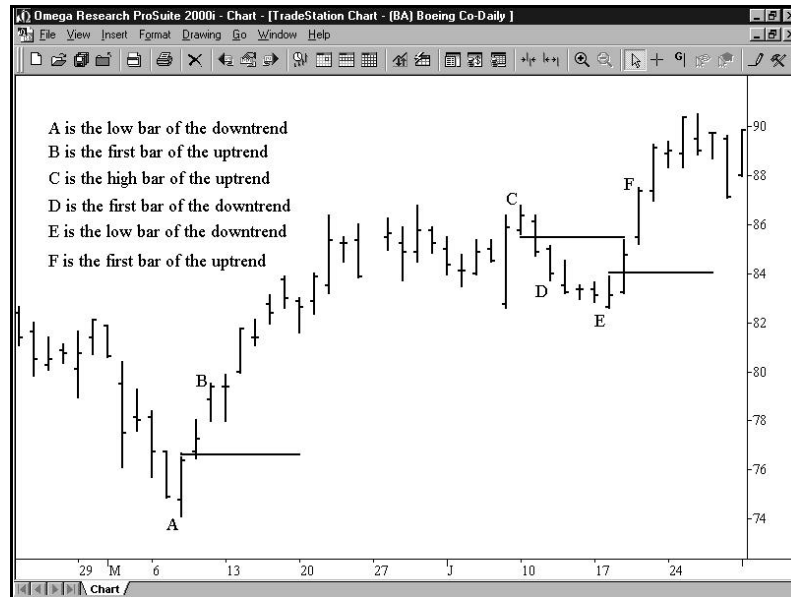


Figure 1. New Uptrend and New Downtrend

Defining Our Trading Rules

In this system, we defined both long and short entries and exits. We did some setup work to calculate the ATR and a percentage of the ATR to add to opens for long entries and to subtract from opens for short entries. We also set up the calculation of a multiple of the ATR to subtract from the low after a long entry and to add to the high after a short entry. Finally, we set up the calculation of the parabolic. The setups, entries, and exits are described next.

Setup

- Calculate a 10-bar ATR, a default value of 50% of the ATR to determine entry points, and a default of 2 ATRs to determine initial risks.
- Calculate our version of the parabolic.

Entries

- After a bar posts a low greater than the high of the lowest bar in a downtrend, buy at the open plus 50% of the ATR. The entry must occur within 5 bars of the beginning of the new uptrend.
- After a bar posts a high less than the low of the highest bar in an uptrend, sell short at the open minus 50% of the ATR. The entry must occur within 5 bars of the beginning of the new downtrend.

Exits

- After a long entry, initiate a parabolic trailing stop at the low of the entry bar minus 2 ATRs.
- After a short entry, initiate a parabolic trailing stop at the high of the entry bar plus 2 ATRs.

Designing & Formatting

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

Easy Language System Components: Jack-In-The-Box (STAD8: Jack-In-Box)

System Inputs (STAD8: Jack-In-Box):

INPUT	DEFAULT	DESCRIPTION
Qualifying_Bars	5	Number of bars for which the setup is valid
ATR_Pcnt	.50	Number of Average True Ranges above/below the Open of the next bar, at which the order is placed.

Signal Components:

1. Jack In The Box
2. Parabolic Trailing LX
3. Parabolic Trailing SX

EasyLanguage Signal: Jack In The Box:

Inputs: QBars(5), ATRPcnt(.50);

Variables: CurrentTrend(0), TrendHigh(High), TrendLow(Low), BuyFlag(False), SellFlag(False), Counter(0), ATR(0);

ATR = AvgTrueRange(10);

{Trend Direction Determination and Setup}

If Low > TrendHigh Then Begin

CurrentTrend = 1;
TrendHigh = High;
TrendLow = Low;
BuyFlag = True;
Counter = 0;

End;

If High < TrendLow Then Begin

CurrentTrend = -1;
TrendHigh = High;
TrendLow = Low;
SellFlag = True;
Counter = 0;

End;

{Trend High/Low Price Updates}

If CurrentTrend = 1 AND High > TrendHigh Then Begin

TrendHigh = High;
TrendLow = Low;

End;

If CurrentTrend = -1 AND Low < TrendLow Then Begin

```

TrendHigh = High;
TrendLow = Low;
End;

{Setup Period Tracker}
If MarketPosition = 1 OR Counter >= QBars Then
    BuyFlag = False;
If MarketPosition = -1 OR Counter >= QBars Then
    SellFlag = False;

Counter = Counter + 1;

{System Entries}
If BuyFlag Then
    Buy ("B1") Next Bar at Open Next Bar + (ATRPcnt * ATR) Stop;
If SellFlag Then
    Sell ("S1") Next Bar at Open Next Bar - (ATRPcnt * ATR) Stop;

```

Signal Inputs (Jack In The Box):

INPUT	DEFAULT	DESCRIPTION
Qbars	5	Number of bars for which the setup is valid
ATRPcnt	.50	Number of Average True Ranges above/below the Open of the next bar, at which the order is placed.

In addition to the Input above, we define the following variables:

Signal Variables (Jack In The Box):

VARIABLE	DEFAULT	DESCRIPTION
CurrentTrend	0	Indicates the direction of the current trend: Uptrend = 1; Downtrend = -1.
TrendHigh	0	The highest High during the current trend
TrendLow	0	The lowest Low during the current trend
BuyFlag	False	True/False variable that determines if we are within the BuySetup period.
SellFlag	False	True/False variable that determines if we are within the SellSetup period.
Counter	0	Keeps track of the number of bars that have elapsed in the Buy/Sell period
ATR	0	Holds the value of the Average True Range.

Entry Setup

During the setup, the Average True Range is calculated and assigned to the variable ATR.

```
ATR = AvgTrueRange(10);
```

When the Low crosses above the TrendHigh value, the setup is confirmed and the current trend becomes up (as indicated by setting CurrentTrend to 1). At this point the TrendHigh and TrendLow values are set to reflect the current uptrend. As the uptrend continues, these values will be updated as more recent breakouts occur. In addition, the Counter is set to 0, since a new setup has begun.

```
If Low > TrendHigh Then Begin
```

```
    CurrentTrend = 1;
    TrendHigh = High;
    TrendLow = Low;
    BuyFlag = True;
    Counter = 0;
```

```
End;
```

When the High crosses below the TrendLow value, the setup is confirmed and the current trend becomes down (as indicated by setting CurrentTrend to -1). At this point the TrendHigh and TrendLow values are set to reflect the current downtrend. As the downtrend continues, these values will be updated as more recent breakouts occur. In addition, the Counter is set to 0, since a new setup has begun.

```
If High < TrendLow Then Begin
```

```
    CurrentTrend = -1;
    TrendHigh = High;
    TrendLow = Low;
    SellFlag = True;
    Counter = 0;
```

```
End;
```

If the Trend continues upward and establishes new highs (Indicated by the High > TrendHigh), the TrendHigh and TrendLow variables are updated accordingly.

```
If CurrentTrend = 1 AND High > TrendHigh Then Begin
```

```
    TrendHigh = High;
    TrendLow = Low;
```

```
End;
```

If the Trend continues downward and establishes new lows (Indicated by the Low < TrendLow), the TrendHigh and TrendLow variables are updated accordingly.

```
If CurrentTrend = -1 AND Low < TrendLow Then Begin
```

```
    TrendHigh = High;
    TrendLow = Low;
```

```
End;
```

If a long position is taken OR the counter reaches the QBars value, the Setup for the Long Entry is ended by setting BuyFlag to FALSE. If a short position is taken or the counter reaches the QBars value, the Setup for the Short Entry is ended by setting SellFlag to FALSE.

```
If MarketPosition = 1 OR Counter >= QBars Then
    BuyFlag = False;
If MarketPosition = -1 OR Counter >= QBars Then
    SellFlag = False;
```

The Counter variable is used to count the number of bars that have occurred since the setup began.

```
Counter = Counter + 1;
```

Long Entry

During the Buy Setup period, a Long Entry order is placed at the Open of the next bar, plus the specified number of Average True Ranges.

```
If BuyFlag Then
    Buy ("B1") Next Bar at Open Next Bar + (ATRPcnt * ATR) Stop;
```

Short Entry

During the Sell Setup period, a Short Entry order is placed at the Open of the next bar, minus the specified number of Average True Ranges.

```
If SellFlag Then
    Sell ("S1") Next Bar at Open Next Bar - (ATRPcnt * ATR) Stop;
```

EasyLanguage Signal: Parabolic Trailing LX:

```
Inputs: Acceleration(.2), FirstBarMultp(1.5);
Variables: SAR(0), AF(0), StopPrice(0), MP(0), HighValue(0);
```

```
MP = MarketPosition;
If High > HighValue Then
    HighValue = High;

If MP = 1 Then Begin
    If MP[1] <> 1 Then Begin
        StopPrice = Low - Average(TrueRange, 3) * FirstBarMultp;
        AF = Acceleration;
        HighValue = High;
    End
    Else Begin
        StopPrice = StopPrice + AF * (HighValue - StopPrice);
        If HighValue > HighValue[1] AND AF < 0.2 Then
            AF = AF + MinList(Acceleration, 0.2 - AF);
        End;
        If StopPrice > Low Then
            StopPrice = Low;
    End;

If MP = 1 Then
    ExitLong ("PM") Next Bar at StopPrice Stop;
```

Signal Inputs (Parabolic Trailing LX):

INPUT	DEFAULT	DESCRIPTION
Acceleration	.02	the the acceleration factor of the Parabolic
FirstBarMultp	1.5	a factor used to multiply the three bar Average True Range to place a stop loss order in the bar of entry

In addition to the Input above, we define the following variables:

Signal Variables (Parabolic Trailing LX):

VARIABLE	DEFAULT	DESCRIPTION
AF	0	Holds the acceleration value
StopPrice	0	Holds the calculated Stop price value
MP	0	Holds the Market Position value
HighValue	0	Holds the highest price of the position

Setup

First, we will assign the MarketPosition keyword to a variable so we can reference previous values in subsequent steps.

```
MP = MarketPosition;
```

Next we will capture the highest high into the HighValue variable

```
If High > HighValue Then
    HighValue = High;
```

If the system is in a long position and it was not in a long position one bar ago (in other words the system is in the first bar of a long position) we will calculate the stop price as the low minus the three bar average true range multiplied by the input FirstBarMultp. Also, we will assign the Acceleration to the variable AF and the the high of this bar to the variable HighValue

```
If MP = 1 Then Begin
    If MP[1] <> 1 Then Begin
        StopPrice = Low - Average(TrueRange, 3) * FirstBarMultp;
        AF = Acceleration;
        HighValue = High;
    End
```

If the system is in a long position, but it is not in the first bar, we will calculate the StopPrice as the previous value of the stop price plus the difference from the high to the stop price times the acceleration value.

If the market is making a higher high, and the acceleration factor is less than 0.2 then we will increase the acceleration factor by the input Acceleration, or 0.2 minus the value of the AF value, whichever value is lower.

```
Else Begin
    StopPrice = StopPrice + AF * (HighValue - StopPrice);
    If HighValue > HighValue[1] AND AF < 0.2 Then
        AF = AF + MinList(Acceleration, 0.2 - AF);
End;
```

If the StopPrice is greater than the low, then we will assign the low to the variable StopPrice.

```
If StopPrice > Low Then
    StopPrice = Low;
End;
```

Long Exit

Finally, if we are in a long position, we will place a stop order to exit the market at the stop price.

```
If MP = 1 Then
    ExitLong ("PM") Next Bar at StopPrice Stop;
```

Signal Inputs (Parabolic Trailing SX):

INPUT	DEFAULT	DESCRIPTION
Acceleration	.02	the acceleration factor of the Parabolic
FirstBarMultp	3	a factor used to multiply the three bar Average True Range to place a stop loss order in the bar of entry

In addition to the Input above, we define the following variables:

Signal Variables (Parabolic Trailing SX):

VARIABLE	DEFAULT	DESCRIPTION
AF	0	Holds the acceleration value
StopPrice	0	Holds the calculated Stop price value
MP	0	Holds the Market Position value
LowValue	0	Holds the lowest price of the position

Setup

First, we will assign the `MarketPosition` keyword to a variable so we can reference previous values in subsequent steps.

```
MP = MarketPosition;
```

Next we will capture the lowest low into the `LowValue` variable

```
If Low < LowValue Then
    LowValue = Low;
```

If the system is in a short position and it was not in a short position one bar ago (in other words the system is in the first bar of a short position) we will calculate the stop price as the high plus the three bar average true range multiplied by the input `FirstBarMultp`. Also, we will assign the `Acceleration` to the variable `AF` and the the low of this bar to the variable `LowValue`.

```
If MP = -1 Then Begin
    If MP[1] <> -1 Then Begin
        StopPrice = High + Average(TrueRange, 3) * FirstBarMultp;
        AF = Acceleration;
        LowValue = Low;
    End
```

If the system is in a short position, but it is not in the first bar, we will calculate the `StopPrice` as the previous value of the stop price minus the difference from the stop price to the low value times the acceleration value. If the market is making a lower low, and the acceleration factor is less than 0.02 then we will increase the acceleration factor by the input `Acceleration`, or 0.2 minus the value of the `AF` value, whichever value is lower.

```
Else Begin
    StopPrice = StopPrice - AF * (StopPrice - LowValue);
    If LowValue < LowValue[1] AND AF < 0.2 Then
        AF = AF + MinList(Acceleration, 0.2 - AF);
End;
```

If the `StopPrice` is lower than the high, then we will assign the high to the variable `StopPrice`.

```
If StopPrice < High Then
    StopPrice = High;
End;
```

Short Exit

Finally, if we are in a short position, we will place a stop order to exit the market at the stop price.

```
IF MP = -1 Then
    ExitShort ("PM") Next Bar at StopPrice Stop;
```

Testing & Improving

We tested Jack-in-the-Box on a British Pound futures contract and on IBM. For the British Pound, the optimized inputs are as follows:

Qbars = 4
 ATRPcnt = 1 (100%)
 SX acc = .03
 SX mult = 1
 LX acc = .01
 LX mult = 4

Testing & Improving

We tested a continuous back-adjusted British Pound futures contract on daily data from 1/2/84 to 3/12/99. \$40 was deducted for slippage, and \$10 was deducted for commission. The TradeStation System Report [Figure 2, *TradeStation System Report: Summary*] shows that our system earned \$73,220 on 150 trades, with 41% of the trades profitable. The average win (\$3,164) was 2.26 times as large as the average loss (\$1,397), and the average trade made \$488. The system let profits run for an average of 23 bars but cut losses short in an average of 11 bars. The profit factor of \$1.60 means that the system earned \$1.60 for each \$1.00 it lost.

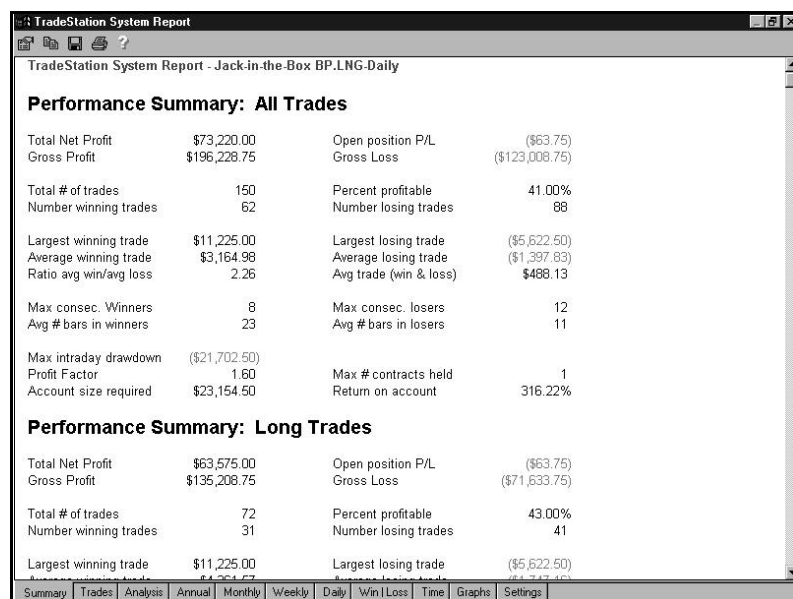


Figure 2. TradeStation System Report: Summary

The Analysis section of the System Report [Figure 3, *TradeStation System Report: Analysis*] adds valuable information to the Summary Report. The RINA Index (which combines select net profit, time in the market, and drawdown into one ratio) was 42.54 for Jack-in-the-Box, with 30 being the standard for an acceptable system. The Net Profit/Largest Loss Ratio was 13.02 (7 or higher can be considered good), and the Net Profit/Maximum Drawdown Ratio of 13.14 exceeded the reference value of 5. The system produced 3 positive outliers (trades more than 3 standard deviations greater than the average trade) for a total gain on the 3 trades of \$32,075, and there were no negative outliers.

The Win/Loss section of the System Report [Figure 4, *TradeStation System Report: Win/Loss*] breaks the winning and losing trades down into their respective series.

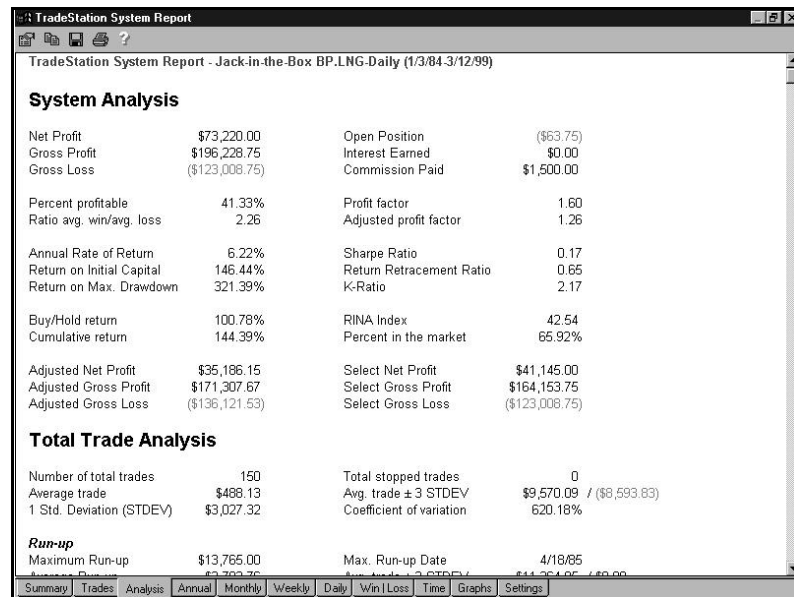


Figure 3. TradeStation System Report: Analysis

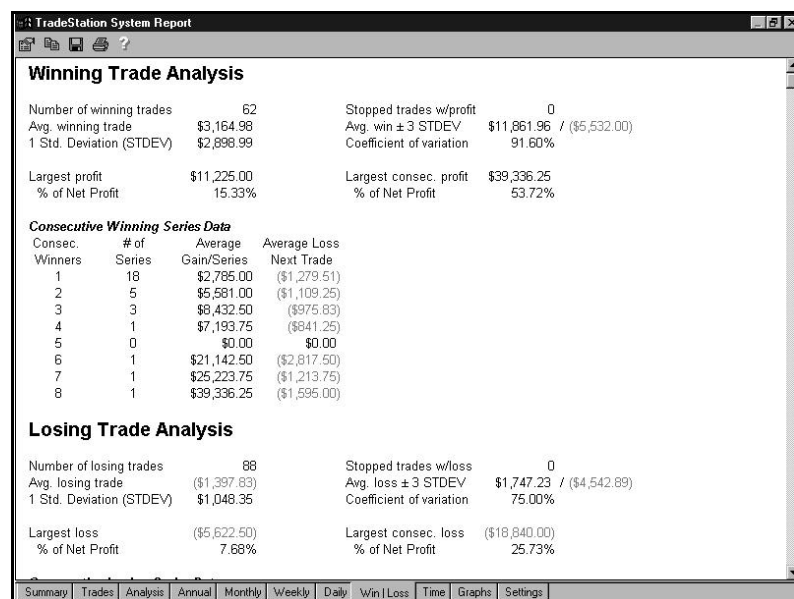


Figure 4. TradeStation System Report: Win/Loss

For example, the system generated 5 series of 2 consecutive winners and 3 series of 3 consecutive winners. The series of 2 consecutive winners averaged \$5,581 per series, while the series of 3 consecutive winners averaged \$8,432 per series. On the losing side, there was only 1 series of 2 consecutive losers, but there were 7 series of 3 consecutive losers. The series of 2 consecutive losing trades lost \$6,270, and the 7 series of 3 consecutive losing trades averaged -\$4,461. In many of our trading systems, we incur longer series of losses than series of profits, but the average profits are generally greater than the average losses.

Next, let's look at several of the graphs that are available in the TradeStation 2000i System Report. The Equity Curve in Figure 5 [Figure 5, *Equity Curve*] displays an accumulation of profits from 0 to almost \$80,000 on a one-contract basis. The Underwater Equity Curve [Figure 6, *Underwater Equity Curve*] as usual looks a lot worse than it really is. The shaded areas ("underwater") represent

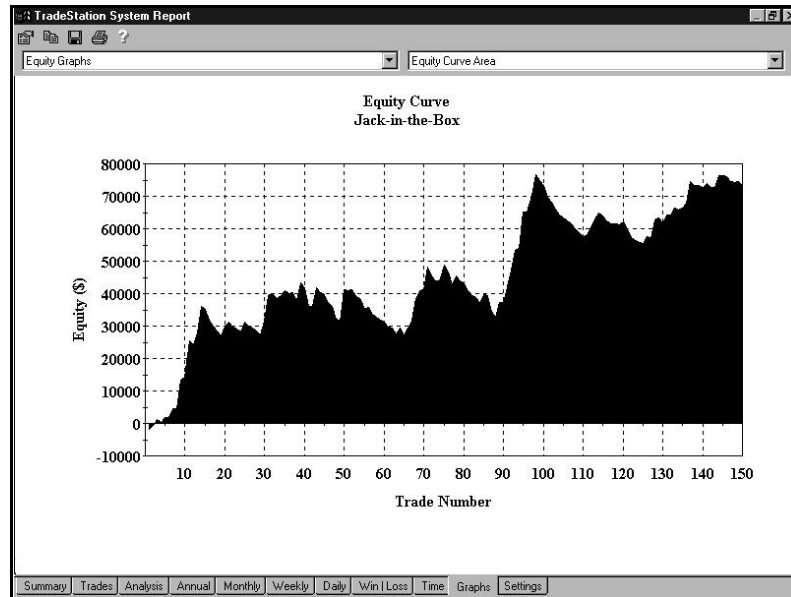


Figure 5. Equity Curve

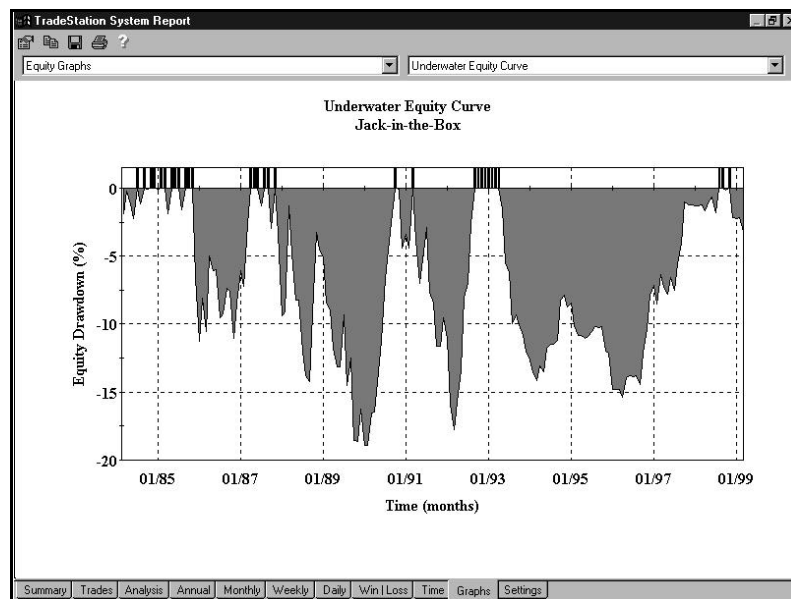


Figure 6. Underwater Equity Curve

drawdowns between equity peaks. Note that the worst drawdown in our 15-year test period was less than 19%. The last graph we'll focus on for Jack-in-the-Box and the British Pound is Monthly Rolling Net Profit. [Figure 7, *Monthly Rolling Net Profit*] This graph is like a snapshot of our equity taken at the end of each month. It shows that the system earned substantial profits over a long period of time, and that our account was never in serious trouble.

We also tested Jack-in-the Box on IBM daily data from 1/84 to 3/99. We bought 100 shares per trade and deducted \$.13 per share for slippage and \$.10 per share for commission.

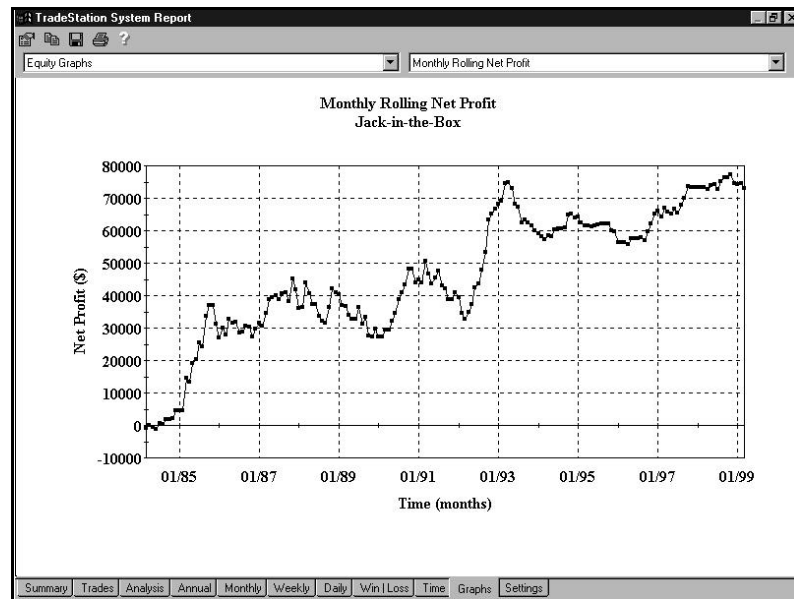


Figure 7. Monthly Rolling Net Profit

Here are the optimized inputs for IBM (for the long side only):

Qbars = 6
 ATRPcnt = .75
 LX acc = .02
 LX mult = 4

The System Analysis Report [Figure 8, *System Analysis*] tells us that our system earned \$13,794 for each 100 shares traded. In the 15-year test period, the system made 44 trades (long side only) of which 68% were profitable. The profit factor was exceptional at 5.28, indicating that the system made \$5.28 for each \$1.00 it lost. The average trade (counting both winners and losers) earned \$313 per 100 shares.

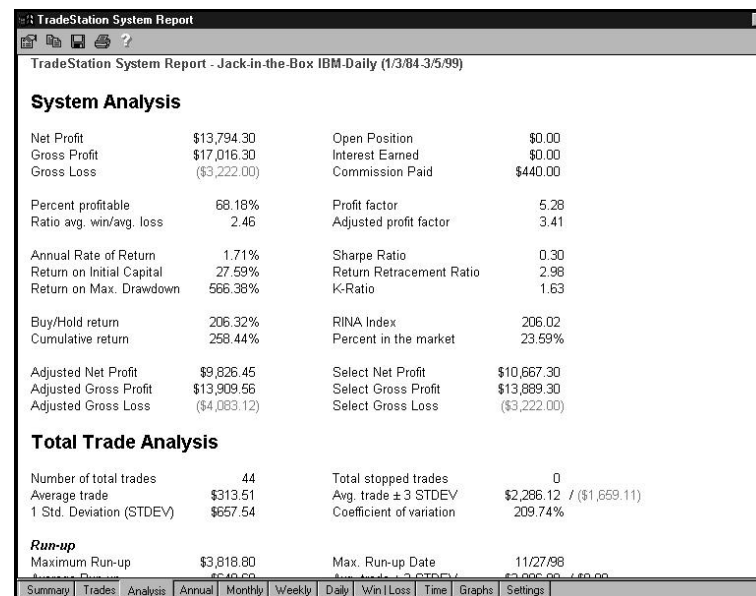


Figure 8. System Analysis



Figure 9. Bar Chart

Figure 9 [Figure 9, bar chart] depicts the system's trades for the first 11 months of 1998. Four of the trades were winners, while only one trade was a small loser. The Equity Curve [Figure 10, Equity Curve] shows our trading account increasing slowly but steadily for the first 30 trades and then increasing powerfully.

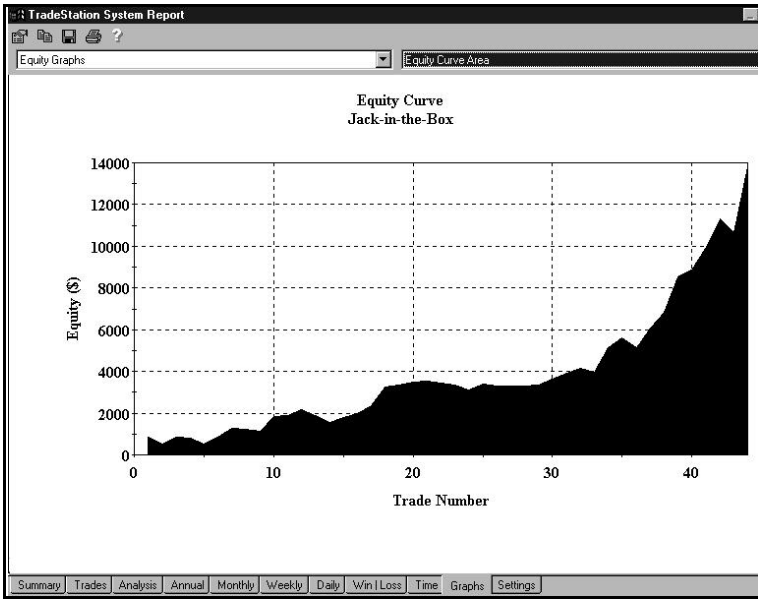


Figure 10. Equity Curve

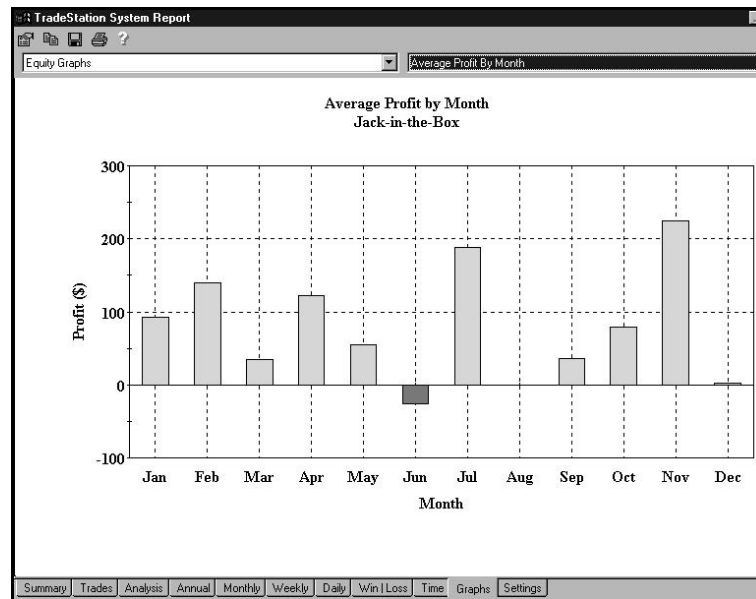


Figure 11. Average Profit by Month

The graph Average Profit by Month [Figure 11, *Average Profit by Month*] shows that our system produced 11 months that were profitable when averaged over the 15-year test period, versus only 1 losing month.

To try to improve this system, we would suggest enabling the Pyramiding setting in the System Properties section of SystemBuilder. It might be extremely profitable to add to your position on each new entry signal rather than limiting your trading to one entry in the same direction at a time. Of course, pyramiding also increases your risk. Rather than pyramiding an unlimited number of times in the same trend, you should probably limit your pyramiding to a specified number of add-ons per position.

THIS PAGE LEFT BLANK INTENTIONALLY

CHAPTER 5

ActivityBar™ Price Distribution System

ActivityBar studies provide the ability to actually see where the price action occurred throughout the bar. This provides us with a new perspective on the price charts. In this section we use a System to take advantage of our expanded point of view.

The EasyLanguage for the System is very simple, but it relies completely on the results and compression of the ActivityBar study that is applied to the chart. For this System we used the Price Distribution ActivityBar, although any ActivityBar study could realistically be used. The Price Distribution ActivityBar was applied to a 60 minute chart, with a 1 minute ActivityData compression. Although a 60/1 minute combination was used in our tests of the System, other Chart/ActivityData compression combinations can be used based on your own trading style.

The System first determines the Modal price of the current bar. The Modal price is the price at which the most action occurred during that bar. Thus, it was the price at which the market felt most "comfortable" during that period. Remember that the price that was most heavily traded does not necessarily correlate with the price at which the bar closed. If the next bar opens above the Modal price and the ActivityBar average is greater than the previous average, it may be an indication that the market is ready to continue above this "comfort zone". If the next bar opens below the Modal price, or below 1 standard deviation of the Mode, as specified in this system and the ActivityBar average is greater than the previous average, it may be an indication that the market is ready to move downward from this "comfort zone".

Long trades are closed out if the Open of the next bar is below the Modal Price of the current bar. Short trades are closed out after a fixed period that can be optimized.

If the Open of the next ActivityBar is greater than the Mode price, a Buy order will be placed on the Open of the next bar. The idea behind this is that if the Open is greater than the Modal price, the upward trend will continue on that bar. Remember that the most heavily traded price does not necessarily correlate with the price at which the bar closed.

Defining Our Trading Rules

Long Entry

- a) Determine the short term-trend. The trend is considered up if the ActivityBar average is greater than the ActivityBar average of the previous bar.
- b) If the short-term trend is up and the Open of the next bar is greater than the Modal price, a Buy order is placed at the Open of the next bar.

Short Entry

- a) Determine the short term-trend. The trend is considered down if the ActivityBar average is less than the ActivityBar average of the previous bar.
- b) If the short-term trend is down and the Open of the next bar is less than a specified standard deviation below the Modal price, a Sell order is placed at the Open of the next bar.

Long Exit

- a) Determine the short term-trend. The trend is considered down if the ActivityBar average is less than the ActivityBar average of the previous bar.
- b) If the short-term trend is down and the Open of the next bar is below the Modal price, a Long Exit order is placed at the Open of the next bar.
- c) If the short-term trend is down and the Modal price of the current ActivityBar is below the Modal price of the previous bar, a Long Exit order is placed at the Open of the next bar.

Short Exit

- a) If Open of the next bar is below the Modal price, a Short Exit order is placed at the Open of the next bar. Since the short positions tend to be of a much shorter duration, the short-term average is not taken into consideration.
- b) If Modal price of the current ActivityBar is above the Modal price of the previous bar, a Short Exit order is placed at the Open of the next bar. Since the short positions tend to be of a much shorter duration, the short-term average is not taken into consideration.

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 System Components: Price Distribution (STAD8: Price Dist)

System Inputs (STAD8: Price Dist):

INPUT	DEFAULT	DESCRIPTION
Average_Length	3	Length, expressed in bars, used to calculate the short-term Moving Average.

Signal Components:

1. AB Price Dist
2. AB Price Dist Exits

EasyLanguage Signal: Price Distribution (AB Price Dist)

Inputs: AvgLength(3);

Variables: AvgVal(0), ModePrice(0);

{Calculation Variables}

AvgVal = Average(Close, AvgLength);

ModePrice = AB_ModePrice(RightSide);

{Long Entry}

If Open of Next Bar > ModePrice AND AvgVal > AvgVal[1] Then

 Buy Next Bar at Market;

{Short Entry}

If Open of Next Bar < AB_GetZoneLow(RightSide) AND AvgVal < AvgVal[1] Then

 Sell Next Bar at Market;

{Long Exit (Next Open)}

If Open of Next Bar < ModePrice AND AvgVal < AvgVal[1] Then

 ExitLong ("LX1") Next Bar at Market;

{Short Exit (Next Open)}

If Open of Next Bar > ModePrice Then

 ExitShort ("SX1") Next Bar at Market;

Signal Inputs (AB Price Dist):

INPUT	DEFAULT	DESCRIPTION
AvgLength	3	Length, expressed in bars, used to calculate the short-term Moving Average.

In addition to the Input above, we define the following variables:

Signal Variables (AB Price Dist):

VARIABLE	DEFAULT	DESCRIPTION
AvgVal	0	Maintains the value of the Moving Average
ModePrice	0	Maintains the Modal value of the ActivityBar cells on the right side of the ActivityBar.

Setup

The Average and the Modal Price are calculated for use throughout the signal.

```
AvgVal = Average(Close, AvgLength);
ModePrice = AB_ModePrice(RightSide);
```

Long Entry

If the Open of the next bar is greater than the modal price and the ActivityBar average is greater than the average of the previous bar, a Buy order is placed at the Open of the next bar.

```
If Open of Next Bar > ModePrice AND AvgVal > AvgVal[1] Then
    Buy Next Bar at Market;
```

Short Entry

For the Short Entry, we compare the Open of the next bar to the lower extreme of the Price Distribution ActivityZone. This ActivityZone, by default in the ActivityBar study, is made up of 1 Standard Deviation above and 1 Standard Deviation below the Modal price. By requiring that the Open of the next bar be below the lower extreme of the zone as opposed to just the Modal price, we try to confirm the downward move. Thus, if the above criterion is True and the ActivityBar average is less than the average of the previous bar, a Sell order is placed at the Open of the next bar.

```
If Open of Next Bar < AB_GetZoneLow(RightSide) AND AvgVal < AvgVal[1] Then
    Sell Next Bar at Market;
```

Long Exit

The Exits below are included in this Signal along with the Entries, since they reference next bar prices like the Entries. First, the Open of the next bar is compared to the Modal price in order to determine a possible change in market direction. Next the ActivityBar average is evaluated in order to determine if there has been a change in the short term trend. If both criteria are met, a Long Exit is placed at the Open of the next bar.

```
If Open of Next Bar < ModePrice AND AvgVal < AvgVal[1] Then
    ExitLong ("LX1") Next Bar at Market;
```


Short Exit

For the Short Exit, only the relationship of the Open of the next bar and the Modal price are evaluated. Since the Short side tends to reverse much quicker, the short-term average is not taken into consideration. Thus, if the Open of the next bar is greater than the Modal price, indicating a potential shift in market direction, a Short Exit is placed at the Open of the next bar.

```
If Open of Next Bar > ModePrice Then
    ExitShort ("SX1") Next Bar at Market;
```

EasyLanguage Signal: Price Distribution Exits (AB Price Dist Exits)

```
Inputs: AvgLength(3);
Variables: AvgVal(0), ModePrice(0);
```

```
{Calculation Variables}
AvgVal = Average(Close, AvgLength);
ModePrice = AB_ModePrice(RightSide);
```

```
{Long Exit 2}
If ModePrice < ModePrice[1] AND AvgVal < AvgVal[1] Then
    ExitLong ("LX2") This Bar on Close;
```

```
{Short Exit 2}
If ModePrice > ModePrice[1] Then
    ExitShort ("SX2") This Bar on Close;
```

Signal Inputs (AB Price Dist Exits):

INPUT	DEFAULT	DESCRIPTION
AvgLength	3	Length, expressed in bars, used to calculate the short-term Moving Average.

In addition to the Input above, we define the following variables:

Signal Variables (AB Price Dist Exits):

VARIABLE	DEFAULT	DESCRIPTION
AvgVal	0	Maintains the value of the Moving Average
ModePrice	0	Maintains the Modal value of the ActivityBar cells on the right side of the ActivityBar.

Setup

The Average and the Modal Price are calculated for use throughout the signal.

```
AvgVal = Average(Close, AvgLength);
ModePrice = AB_ModePrice(RightSide);
```

Long Exit

If the Modal price begins to fall and the ActivityBar average is lower than the average of the previous bar, it is another indication that the market direction may be shifting. Thus, when these two criteria are met, a Long Exit is placed at the Close of the current bar.

```
If ModePrice < ModePrice[1] AND AvgVal < AvgVal[1] Then
    ExitLong ("LX2") This Bar on Close;
```

Short Exit

For the Short Exit, only the direction of the Modal price is evaluated. Since the Short side tends to change direction much quicker, the short-term average is not taken into consideration. Thus, if the Modal price is lower than the Modal price of the previous bar, a Short Exit is placed at the Close of the current bar.

```
If ModePrice > ModePrice[1] Then
    ExitShort ("SX2") This Bar on Close;
```

Testing & Improving

The System was evaluated on Dell and Microsoft. We specified an Initial Equity of \$50,000, an amount per transaction of \$10,000, and a minimum lot size of 10 shares. Remember though, that regardless of what symbol(s) the system may be tested upon, the Chart/ActivityData compression combination can play a major role in the results of the System. For both stocks, a 60 minute chart was used, with the ActivityData compression set to 1 minute. These compression values can be adjusted as necessary to suit your own trading needs/style.

Although the System itself has only a single Input, the system results can vary greatly depending on the actual chart compression that is used and any additional Stop signals that may be added. A Dollar Risk Trailing Stop comes to mind as a good protective Stop that could be used.

On the 60 minute chart of Microsoft (*Figure 1*) over most of the month of March, the System results were not spectacular, but they were acceptable (*Figure 2*). Although the System was only 42% profitable, the Average Winning Trade was almost four times the size of the Average Losing Trade. In addition, the System rode out the winners, while at the same time cutting the losers short.

The 60 minute chart of Dell proved to be somewhat more encouraging. The overall numbers looked good, with a 68% profitability rating (*Figure 3*), but the supporting statistics were not as strong as those from Microsoft. The Equity Curve was also very encouraging with a strong upward trend (*Figure 4*).

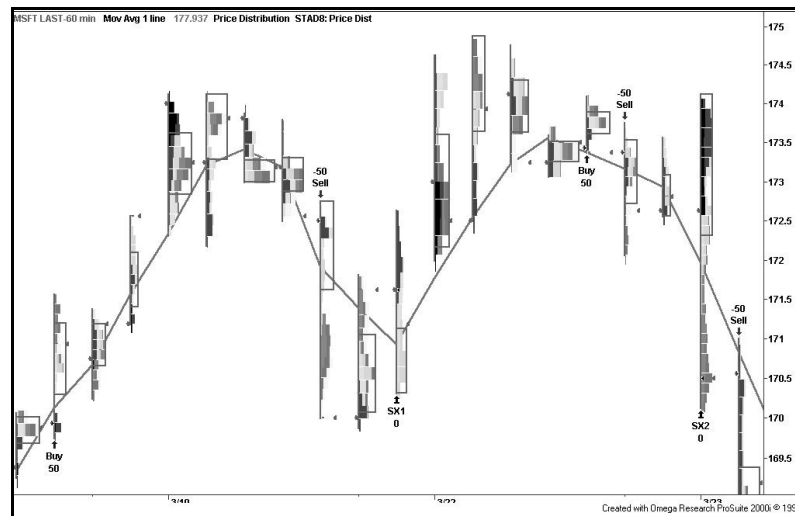


Figure 1. 60 Minute Chart: Microsoft

TradeStation System Report			
TradeStation System Report - STAD8: Price Dist MSFT-60 min. (3/1/99-3/26/99)			
Performance Summary: All Trades			
Total Net Profit	\$1,171.24	Open position P/L	\$0.00
Gross Profit	\$1,951.86	Gross Loss	(\$780.62)
Total # of trades	24	Percent profitable	42.00%
Number winning trades	10	Number losing trades	14
Largest winning trade	\$631.25	Largest losing trade	(\$131.25)
Average winning trade	\$195.19	Average losing trade	(\$55.75)
Ratio avg win/avg loss	3.50	Avg trade (win & loss)	\$48.80
Max consec. Winners	3	Max consec. losers	4
Avg # bars in winners	6	Avg # bars in losers	1
Max intraday drawdown	(\$407.76)		
Profit Factor	2.50	Max # contracts held	60
Account size required	\$407.76	Return on account	287.24%

Figure 2. TradeStation Performance Summary: Microsoft



Figure 3. TradeStation Performance Summary: Dell

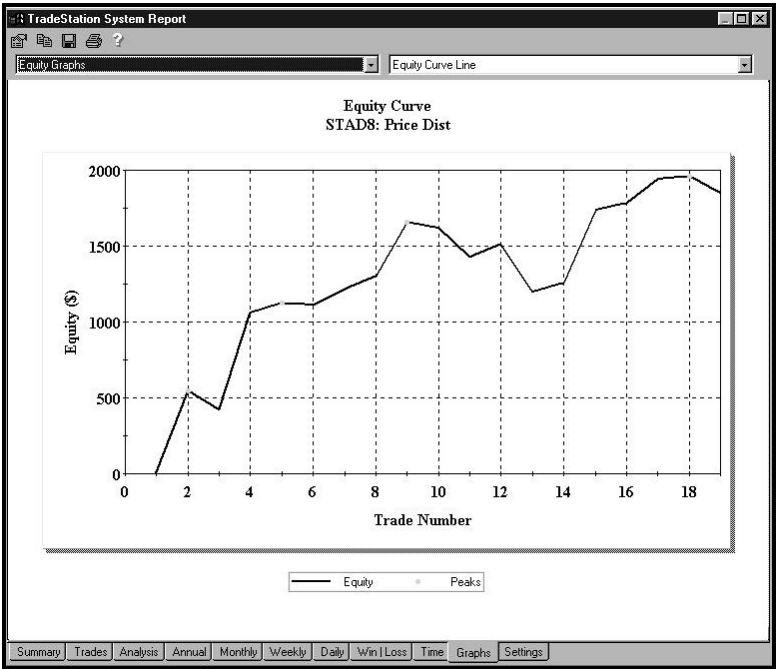


Figure 4. Equity Curve

CHAPTER 6

Fixed Ratio Money Management

by Ryan Jones

There is no question that today's trader is more equipped with knowledge, ideas, trading strategies, indicators, and the ability to combine these elements into his trading than ever before in the history of this industry.

Yet, it is amazing that the statistics of profitable traders versus those who lose money remain relatively constant year after year. These statistics are bleak at best; upwards of 80 to 90% of all traders losing money at year-end. Traders may trade to make more money than they would investing in conventional investment vehicles, yet the percentages indicate that few make money trading and most don't accomplish that goal. There are many reasons and many theories as to why these facts remain true. One of these reasons is because people get involved in trading with very short-term goals. They are falsely lead to believe that the short-term goal of getting rich quick is valid and even realistic through trading futures and commodities because trading itself is short-term in comparison to standard investment vehicles. However, if one is going to successfully trade in these markets, one must have a longer-term outlook.

There are two basic aspects of trading, with the first being where to get in and where to get out of which markets. This aspect is covered through indicators, oscillators, moving averages, price action patterns, trading systems, software, newsletters, hot lines, fundamental reports, and the list goes on. It encompasses exit rules such as where to place stops, where to place profit targets, trailing stops and whatever other means are used to decide when to exit a trade. This aspect covers all types of trading: options trading, futures trading, spreads, writing options, short-term trading, day-trading, long-term trading, break-out trading, trend-following trading, and again the list goes on. Sadly, most traders' emphasis is placed solely on this one aspect of trading. I believe that a trader's emphasis is confined to this one aspect as a result of their inability or unwillingness to address a longer-term trading plan.

If you happen to be in the camp that does not regard money management principles as a highly important and powerful aspect of trading, consider the following example:

A coin flipping game is offered in Vegas. The rules are that a coin is going to be flipped in the air 100 times. If the coin lands heads up, you will lose \$1.00 for every \$1.00 you bet. But if the coin lands tails up you will win \$2.00 for every \$1.00 you bet. Obviously, if this game were really offered in Vegas, the coin would not have a tails side, but that is beside the point. Regard this game as having an equal chance of landing heads or tails with each flip. You are given \$100.00 to bet on the coin flipping game. You may bet as much or as little as you wish on each flip (\$1.00 minimum increments). However, being the savvy trader that you are, you are going to apply some sort of money management rules to maximize your return over the 100 flips. These are the choices you have.

- **You may risk 10% of the account balance on each flip of the coin.**
- **You may risk 25% of the account balance on each flip of the coin**
- **You may risk 40% of the account balance on each flip of the coin.**
- **You may risk 51% of the account balance on each flip of the coin.**

As your account balance increases, the size of your bet will also increase. As the account balance decreases, the size of your bet on the next flip will also decrease. If you choose 10% and the next flip of the coin is a winner, you bet \$10.00 on that flip and therefore won \$20.00. Your account balance is now \$120 and you bet 10% of the new amount on the next flip or \$12. If the flip is a loser, you lose only \$12 and the account balance is \$108. Round down and the next bet will be \$10 again. This one is a winner and you are up to \$128. And the cycle goes on for 100 flips of the coin. Which of the above options would you choose? Does it make a big difference?

If you chose:

- **10%, your account would have grown to \$4,700 after 100 flips!**
- **25%, your account would have grown to \$36,100 after 100 flips!**
- **40%, your account would have grown to \$4,700 after 100 flips!**
- **51%, your account would have decreased down to \$31 after 100 flips!**

This assumes that the flips would have been 50/50 regardless of order. The question was, "Does it make a big difference?" and the answer is a resounding YES! Increasing the percentage at risk on each flip by only 15% (going from 10% to 25%) increased the total return by 768%!!! We'll get back to this example later...

The bottom line is that money management does play a huge role in the overall success of traders. It will play a huge role in how successful you are as a trader in the next 5 years. What can proper money management do?

- 1. Keep you from being blown out.**
- 2. Allow you to approach system trading with a plan and the ability to continue trading even if the system fails miserably.**
- 3. Increase profits 5 to 10 fold (500% to 1,000%) without increasing the overall risk of the account.**
- 4. Protect those profits should the system end up failing miserably.**

Consider the following example:

One trader trades a relatively inexpensive hotline for one year. During that one year, the trader's account reaches \$27,000 in profits. The following year, the hotline only makes \$15,000. The total profit of the trader is now \$42,000.

Another trader also trades the same hotline for the first year. However, this trader applied conservative money management principles to the trading. The first year, his profits reach \$63,000 instead of only \$27,000, which is an increase of 233% from the first trader. The second year, however, total profits were \$113,000 instead of only \$15,000 bringing the total profits for the two years of trading up to \$176,000, which computes to a total increase of 419% more profits for the second trader.

The third year of trading for the hotline takes a terrible turn. It loses \$15,000. Trader number 1 is at \$27,000 in profits for the three years of trading. Trader number 2, however, is at \$102,000 in total profits. This means that trader number 1 gave back ALL (100%) of the profits gained during the second year of trading, while the second trader only gave back 65% of the profits gained during the second year.

The question for this example is which trader do you want to be?

We will begin our look at money management with the most common method called the Fixed Fractional method. This method encompasses everything from risking no more than 2% - 3% of your account on each trade to trading one contract for every \$10,000 in your account, to what is called Optimal f (stands for optimal fixed fraction). This article will take a brief look at each while pointing out the pros and cons of using such commonly recommended methods.

The first is risking 2% - 3% of your account on each trade. This is recommended by many brokers and used by most CTA's and CPO's. It states that a trader will not risk more than 2% or 3% of the total account balance on each trade. For this example, the total current account balance will be \$25,000. The next trade signal comes in the Bonds with \$1,000 total risk on the trade per contract. Since no more than 2% can be risked on each trade, a simple math calculation is performed to determine how many contracts should be traded. That calculation is as follows: Total risk on trade divided by total percent of account at risk. $\$1,000 / 2\%$ or $.02 = \$50,000$. In other words, to follow the rules of this method, the Bond trade can not be placed. The reason is that the total % of the account being risked is 4%. (Total \$\$ at risk/account balance = total percentage at risk) Using this percentage, risk can also be translated into trading one contract for every \$50,000 in the account.

There are a couple of things that should be pointed out using this method. First, it is not very efficient. Using this method with the example given about the two traders leaves the first trader trading one contract the entire time, assuming that the trader started out with at least \$50,000 in the account. It also assumes that the trader did not go into an immediate drawdown. If the trader goes into an immediate drawdown, according to the rules, the trader can no longer take the trades, as the risk on the account will be greater than 2% of the account on each trade.

The next method we will take a look at is the "1 contract for every \$10,000 in your account" method. This one is widely recommended because of its simplicity and general ease of implementation. I find it interesting the number of traders who are willing to use this method for those two reasons. Hundreds, even thousands of hours, thousands, even hundreds of thousands of dollars are spent on where to get in and where to get out of which markets, but money management is reduced to a 3.5 second line of "1 contract for every \$10,000". Few traders know the actual effect this statement has on the overall performance of a system.

“The bottom line is that money management does play a huge role in the overall success of traders”

One (1) contract for every \$10,000 in an account is just another way of saying that you are going to risk X% of your equity on each trade. For example, if your largest potential losing trade is \$1,500 and you trade 1 contract for every \$10,000 in your account, you are risking 15% of your account on the next trade. The calculation for this is largest loss/required equity per contract = % at risk on every trade. If your largest potential loss is \$1,000, then you are risking 10% of your equity on every trade.

There are a couple of things that should be pointed out about this method as well. First, it is a flat statement that does not take into account any consideration of the markets being traded, how many markets are being traded, the single trade risks of those markets, and much less the overall potential drawdown of the markets. Let me give you some numbers that will illustrate why these things need to be taken into consideration.

Let's say you are trading a longer-term bond and currency system. Because of the nature of longer-term trading, your stops are not exactly small, ranging in the \$1,200 to \$2,000 area. If your stop on the next trade is \$2,000 and you are trading 1 contract for every \$10,000, you are risking 20% of your account on this trade. If you have \$100,000 in the account, you divide that \$100,000 by \$10,000 and should place 10 contracts on this trade. If it is a loser, your account goes down to \$80,000 on a single trade. But that is not the bad part. If you suffer three of these losers in a row, (which I have done before trading a long-term system of mine), your account drops from \$100,000 to \$52,000 on just three trades! That computes to a 48% drawdown potential on THREE trades! And, it does not matter where your account is, it will always be exactly the same. \$1,000,000 has a potential drawdown of \$480,000 on just three trades.

With this method of managing your money, I could stop there and the point would be quite clear. But to drive home the argument, I will go to the next most commonly recommended money management method called Optimal f. I will then tie the one contract for every \$10,000 method with the Optimal f method before going into what should be used.

Optimal f is short for the optimal fixed fraction to be used on any given trading method and/or system. For example, I gave a coin flipping illustration where risking 25% of your account would have yielded more return than any other percentage possible. Twenty-five percent (25%) is the Optimal f of that particular situation. Optimal f is not 25% for every trading situation. As a matter of fact, Optimal f is different for every trading situation.

Notice that risking 10% of the account on the coin flipping example yielded the same amount of profits as risking 40%. Twenty-five percent (25%) is at the peak. This means that 11% yields the same as 39%, 20% yields the same as 30%, and 24% yields the same as 26%. It is a perfect bell curve. However, notice that this bell curve ends at 50%, because at 51% the positive expectation actually loses money.

This bell curve exists with every trading system, methodology and performance record. Trading is not a controlled statistical game like flipping a coin. The win/loss ratio on each trade is different, and the odds of the next trade being a winner are not necessarily a fixed number either. Optimal f is determined by filing the history of trades through a mathematical formula (which is not relevant for this article) to determine what the Optimal f WAS for the last however — many trades. Notice that I said was. It is impossible to project what Optimal f will be in the future. This becomes rather important when you realize that if you trade a fixed fraction that is higher than the optimal f, you could actually end up losing money. For example, if you calculate Optimal f for the past 100 trades at 15% and trade risking 15% on every trade, the next 100 trades may have an Optimal f of only 10%. If that is the case, you are trading with too much risk. If you were to trade 21% you would lose money.

Going back to the one contract for every \$10,000 scenario, Optimal f for trading may only be 10% and a \$2,000 losing trade is risking 20% according to that method. You will end up making the exact same amount of money as you would by not using money management at all.

I think I have made a pretty good argument for finding a better solution to the type of money management to use in trading. There are literally dozens of other reasons why any type of fixed fractional method should be avoided, but I do not think I need to continue to beat the proverbial dead horse.

There were three main problems that led me to do extensive research on the subject of money management. The first was the fact that the fixed fractional method was the only method available. The second was the fact that since I have a very low tolerance for risk, implementing a small fixed fraction such as 2% - 3% on each trade satisfied my low risk tolerance but sent my alarm of impatience screaming. It would take forever for money management to have any real effect on my trading by using that method. The only other alternative was to increase that percentage, which sent my risk tolerance alarm screaming. I think the Optimal f nailed the last nail in that coffin. This is why and how I developed the Fixed Ratio money management method.

Before I go any further, I do want to acknowledge for those extremely brilliant traders out there who are saying, "A ratio and a fraction are the same thing," that yes, generally, a ratio and fraction are the same thing. However, I want to make clear that the point of reference is completely different. At no time does the Fixed Ratio method reference the percentage being risked on a trade. It is a Fixed Ratio of something entirely different. And, with that out of the way, here's the method.

The Fixed Ratio method accomplishes two things. First, it does not take long for the method to have a substantial effect on my trading and, in particular, my profits. And second, my risk level is very tolerable, known and controlled at all times. The formula for increasing and decreasing the number of contracts (or options or shares of stock) is as follows:

Starting with 1 contract:

Largest Drawdown/2 = Variable Input

Start account balance + (1 * Variable Input) = account level which must be reached before number of contracts is increased.

Largest Drawdown = \$10,000

Variable Input = \$5,000

Starting account balance = \$25,000

Number of contracts traded = 1

Account level to increase to 2 contracts = \$30,000

$\$25,000 + (1 * \$5,000) = \$30,000$ (PLI Previous level of increase).

For trading a number of contracts greater than 1:

PLI + (number of contracts traded * Variable Input) = account level required for additional increase.

PLI = \$30,000

Variable Input = \$5,000

Number of contracts traded = 2

Account level to increase to 3 contracts = \$40,000

$\$30,000 + (2 * \$5,000) = \$40,000$.

“There is one thing that most traders cannot get around when applying money management to trading...”

Essentially, what this method does is require that the same amount of profits be generated for each contract being traded prior to increasing to an additional unit. In the fixed fractional method, one contract was being traded for every X dollars in the account. For example, the 1 contract for every \$10,000 in the account stated that, regardless of the number of contracts being traded, there would always be an increase once another \$10,000 in the account was accumulated. Therefore, if you start out trading one contract with \$10,000 in the account, you would go to two contracts once the account reached \$20,000. One contract alone would have to pull in \$10,000 to increase. But, what happens when there is \$100,000 in the account and you are trading 10 contracts? You increase at \$110,000. What 1 contract was required to produce in the beginning is now being required with 10 contracts. What took maybe 30 trades to produce the first increase in contracts now may only take 3 trades to produce. A \$1,000 winning trade will now increase contracts from 10 to 11.

However, with the Fixed Ratio method, if it takes \$5,000 in profits to increase from one to two contracts, it will take \$50,000 in profits to increase from 10 to 11. ($10 * \$5,000 = \$50,000$). What took 15 trades on average to increase from 1 to 2 contracts will take on average 15 trades to increase from 10 to 11 and 15 trades on average to increase from 99 to 100.

The term Fixed Ratio comes from the fact that whatever variable input used has a fixed ratio to the largest drawdown. If the largest drawdown is \$10,000 and the variable input used is \$5,000, then the variable input is a fixed ratio of 2:1 to the drawdown. If the variable input is \$2,500, then it is a fixed ratio of 4:1 to the largest drawdown.

Before breaking this method down further and illustrating its potential power, there are a few general rules of thumb to keep in mind. First, the higher the ratio, the more aggressive the method is. This, in turn, means that the lower the ratio, the more conservative the method is. More aggressive means more profits and more risk, and more conservative means less profits and less risk. Second, there is no "Optimal r" or optimal fixed ratio that will yield more than others. There is no bell curve with this method. Finally, the lower the variable input, the higher the profits regardless of what relationship it is to the largest drawdown. For example, if the largest drawdown is \$10,000 and a variable ratio of \$5,000 is traded, it will not yield as much profit as a system that has only \$5,000 in drawdowns, and a 2:1 ratio of \$2,500 is used for the variable input. However, the total % of the account at risk is relatively the same at any given time. Therefore, the lower the drawdown of a system and/or portfolio, the more effective the Fixed Ratio will be.

One of the most popular questions I get when traders learn this method has to do with increasing the first contract so quickly. In the first example of the method, the starting account balance was \$25,000 while the first increase came at only \$30,000. Obviously, this satisfies my level of impatience, but does it satisfy my low risk level too? The answer is yes. There is one thing that most traders can not get around when applying money management to trading, and it does not matter what type of money management is being applied. That one thing is the first contract or first option increase. Going from one unit to two units is something every trader must do if they are going to start applying money management (provided they start with one contract). Yet, it does not matter when the trader decides to increase from one to two. It is doubling the risk of the next trade whether the trader decides to do it after only \$5,000 in profits or if the trader waits to do it at \$25,000 in profits. If the first trade with two units is a loser and drops the equity back below the level at which it was increased, it does not matter whether that level was at \$30,000 or only \$5,000. The trader is still only trading one contract in that situation.

But, it does matter. If the trader who waited to increase contracts until \$30,000 was achieved would have started increasing at \$5,000 using the Fixed Ratio method, that trader would have not \$30,000 in profits, but \$90,000 in profits! This is due to the fact that for the remaining \$25,000 in profits, more than one contract would be traded. In fact, there would be five increases during that \$30,000 profitable period. ($\$30,000/\$5,000 = 6$)

This brings us to the power of the method. When I first developed the Fixed Ratio method, the logic of where contract increases should take place was definitely there. However, I first thought that the method would lack the geometric growth potential to achieve great numbers. I was mistaken. To achieve \$1,000,000 in total profits using a variable input of \$5,000 with the Fixed Ratio method would only require \$100,000 based on one contract. In other words, if trader number 1 accomplished \$100,000 in profits starting with one contract and never increasing, that same trader could turn that \$100,000 in profits into \$1,000,000 in profits by applying the Fixed Ratio method using a \$5,000 variable input!

Further, at the \$1,000,000 level, the trader would only be trading 20 contracts! Compare this to the trader who is using the 1 contract for every \$10,000 approach. That trader would reach 20 contracts with only \$200,000 in the account. Given the potential of a \$10,000 drawdown, that trader is risking nearly 60% of the account compared to only 18% with the Fixed Ratio method at 20 contracts! In all fairness, the trader using the 1 contract for every \$10,000 method would reach \$1,000,000 in profits much quicker. At \$1,000,000 that trader is trading 100 contracts but is still risking approximately 60% of the account should a \$10,000 drawdown be suffered. A \$2,000 drawdown with that method would equal where the account would be after a \$10,000 drawdown using the Fixed Ratio method.

The conclusion is that the Fixed Ratio method meets the low risk levels, yet can be applied to trading quickly, effectively and with some monstrous results. It is the best of both worlds. There is, of course, much more to be learned about this method which time and space in this article will not allow. You should at least have enough here to realize that fixed fractional trading should be avoided and to be able to begin applying the Fixed Ratio method to your own trading.

NOTE: Ryan Jones has generously offered to send (at no charge) the EasyLanguage for his basic Fixed Ratio money-management strategy to any STAD Club members who request it. If interested, you can reach Ryan at P.O. Box 25611, Colorado Springs, CO 80936, by telephone at 719.264.8982, by fax at 719.264.8986, and by e-mail at RumeryInc@aol.com.

Of course, this STAD Club article only scratches the surface of Ryan's money-management strategy. Ryan's first book, *The Trading Game: Playing by the Numbers to Make Millions*, has recently been released by John Wiley & Sons. You can order the book directly from Ryan for \$49.95.

THIS PAGE LEFT BLANK INTENTIONALLY

CHAPTER 7

Ryan's Hope Trading System

The entry technique in Ryan's Hope comes from a system that Ryan Jones included in his new book *The Trading Game: Playing by the Numbers to Make Millions* (John Wiley & Sons, 1999). We added some of our favorite trend-following exit techniques to Ryan's entry, and the result is a system with encouraging performance numbers.

Ryan's Hope determines the intermediate-term trend by comparing a simple moving average of closes to the same moving average n-bars ago. For example, the system compares today's 20-bar moving average (Length 1) to the moving average 5-bars ago (Length 2). If today's average is greater than the average 5-bars ago, the trend is up; if today's average is less than the average 5-bars ago, the trend is down.

Next, the system looks for a minor pullback within the intermediate-term trend. The system identifies a pullback in an uptrend when the current close is less than the close a specified number of bars ago, and it identifies a pullback in a downtrend when the current close is greater than the close a certain number of bars ago. For example, if the trend is up, but the close is less than the close 5-bars ago, a countertrend decline has occurred. If the trend is down, but the close is greater than the close 5-bars ago, a countertrend rally is in effect.

The final step in the entry technique is to confirm that the intermediate-term trend hasn't changed due to the countertrend price swing. Add Length 1 (20 in our example above) to Length 2 (5 in our example) for a total of 25. For an uptrend, the current close must be greater than the close 25 bars ago; for a downtrend, the close must be less than the close 25 bars ago. If this condition is true, the system enters in the direction of the trend on the next open.

To Ryan's entry technique, we've added four exit strategies that have served us well in other trend-following systems. Let's look at the exits for a long position. First, we'll exit at our entry price minus a multiple of the average true range (ATR). This stop is intended to cut our losses short if the trade goes against us right away. Second, we'll exit at the highest high since our entry into the long position minus a multiple of the ATR. This stop will allow the trade room to "breathe" but will also lock in profits when the trade has gone our way. Third, we'll include a volatility stop. If the market abruptly moves against us by a much larger than normal amount, we'll exit the trade before things get even worse. The volatility stop will be set at a multiple of ATRs below the previous close for our long position. Fourth, for those exceptional times when we are enjoying greater than usual open profits, we'll tighten our stop to protect as much of the windfall profits as we can. Our big profit stop will be set one point below the lowest low of the past n bars (usually between 4 and 8). Of course, the stops for our short trades will be the mirror image of the stops we've described for our long trades. Finally, we'll add a command (IncludeSystem: "Last Bar";) that will tell the system to exit open positions on the last bar of the test data.

Defining Our Trading Rules

For this system, we defined both long and short entries as well as long and short exits. The setup involved calculating the moving average and the average true range. Setup, entries, and exits are described next.

Setup

- a) Calculate a simple moving average
- b) Calculate an average true range

Long Entries

- a) The moving average must be greater than the moving average of Length 1 bars ago
- b) The close must be less than the close of Length 2 bars ago
- c) The close must be greater than the close (Length 1 + Length 2) bars ago
- d) If a,b, and c are true, buy on the next open

Short Entries

- a) The moving average must be less than the moving average of Length 1 bars ago
- b) The close must be greater than the close of Length 2 bars ago
- c) The close must be less than the close (Length 1 + Length 2) bars ago
- d) If a,b, and c are true, sell on the next open

Long Exits

- a) Exit at the entry price minus n1 ATR's
- b) Exit at the highest high since entry minus n2 ATR's
- c) Exit at the close minus n3 ATR's
- d) Exit at the lowest low of the last n4 bars minus one point if the close is equal to or greater than the entry price plus n5 ATR's

Short Exits

- a) Exit at the entry price plus n1 ATR's
- b) Exit at the lowest low since entry plus n2 ATR's
- c) Exit at the close plus n3 ATR's
- d) Exit at the highest high of the last n4 bars plus one point if the close is equal to or less than the entry price minus n5 ATR's

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 System Components: Ryan's Hope (STAD8: Ryan's Hope)

Inputs

INPUT	DEFAULT	DESCRIPTION
RyansHope1_Length1	20	Length, expressed in bars, used to calculate Moving Average.
RyansHope1_Length2	5	Length, expressed in bars, used to identify a retracement
Volatility_Stop_ATRs	2	The number of Average True Ranges that are used to determine the required volatility to place an Exit order
ATR_Length	10	Length, expressed in bars, used to calculate the Average True Range
Trailing_Stop_ATRs	4	The number of Average True Ranges that are risked from the highest/lowest price of the position
Protective_Stop_ATRs	3	The number of Average True Ranges that are risked in the position.
Big_Profit_ATRs	7	Number of Average True Ranges used to determine the "Big Profit" level
Big_Profit_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. Ryan's Hope
2. ATR Big Profit Stop
3. ATR Protective Stop
4. ATR Trailing Stop
5. ATR Volatility Stop
6. Last Bar Exit

EasyLanguage Signal: Ryan's Hope:

Inputs: Length1(20), Length2(5);

Variables: AvgVal(0);

AvgVal = Average(Close, Length1);

{Long Entry}

If AvgVal > AvgVal[Length2] AND Close < Close[Length2] AND Close > Close[Length1 + Length2]
Then

Buy Next Bar at Market;

{Short Entry}

If AvgVal < AvgVal[Length2] AND Close > Close[Length2] AND Close < Close[Length1 + Length2]
Then

Sell Next Bar at Market;

Signal Inputs (Ryan's Hope):

INPUT	DEFAULT	DESCRIPTION
Length1	20	Length, expressed in bars, used to calculate the short-term Moving Average.
Length2	5	Length, expressed in bars, used to identify a retracement

In addition to the Input above, we define the following variables:

Signal Variables (Ryan's Hope):

VARIABLE	DEFAULT	DESCRIPTION
AvgVal	0	Holds the value of the Moving Average calculation

Setup

The Moving Average calculation is assigned to the variable AvgVal.

AvgVal = Average(Close, Length1);

Long Entry

In order to generate a Long Entry order there are three conditions which must be met. First, the Moving Average is compared to the same Moving Average a specified number of bars in the past. The condition is True if the Moving Average is greater than the Moving Average n-bars ago. Next, we compare the current bar's Close to the Close a specified number of bars in the past. The condition is True if the current close is less than the Close n-bars ago. Finally, to make sure the trend is still intact, we compare the current Close to the Close of the bar a specified number of bars in the past. The number of bars ago is determined by adding the values of Length1 and Length2. If the 3 conditions above are true, then place an order to go Long on the next Open.

If AvgVal > AvgVal[Length2] AND Close < Close[Length2] AND Close > Close[Length1 + Length2]
Then

Buy Next Bar at Market;

Short Entry

In order to generate a Short Entry order there are also three conditions which must be met. First, the Moving Average is compared to the same Moving Average a specified number of bars in the past. The condition is True if the current Moving Average is less than the same Moving Average n-bars ago. Next, we compare the current bar's Close to the Close a specified number of bars in the past. The condition is true if the Close is greater than the Close n-bars ago. Finally, to make sure the trend is still intact, we compare the current Close to the Close of the bar a specified number of bars in the past. The number of bars ago is determined by adding the values of Length1 and Length2. If the 3 conditions above are true, then place an order to go Short on the next Open.

If AvgVal < AvgVal[Length2] AND Close > Close[Length2] AND Close < Close[Length1 + Length2]
Then

Sell Next Bar at Market;

EasyLanguage Signal: ATR Big Profit Stop:

**See Chapter 9

EasyLanguage Signal: ATR Protective Stop:

**See Chapter 9

EasyLanguage Signal: ATR Trailing Stop:

**See Chapter 9

EasyLanguage Signal: ATR Volatility Stop:

**See Chapter 9

EasyLanguage Signal: Last Bar Exit:

**See Chapter 9

Testing & Improving

We tested this system on Crude Oil from 1/84 to 3/99, using a continuous, back-adjusted futures contract. The results include \$40 per contract for slippage and \$10 per contract for commission. First, we optimized Length1 and Length2 with the default values for the other inputs. Second, we optimized IPS (initial protective stop), TS (trailing stop), and VolSt (volatility stop) with the optimized values for Length1 and Length2 and the default values for BigProfit and Nbars. Finally, we optimized BigProfit and Nbars with the optimized values for the lengths and the other stops.

The optimized parameters for Crude Oil were as follows:

Length1 = 20

Length2 = 3

IPS = 3

TS = 5

VolSt = 2

BigProfit = 6

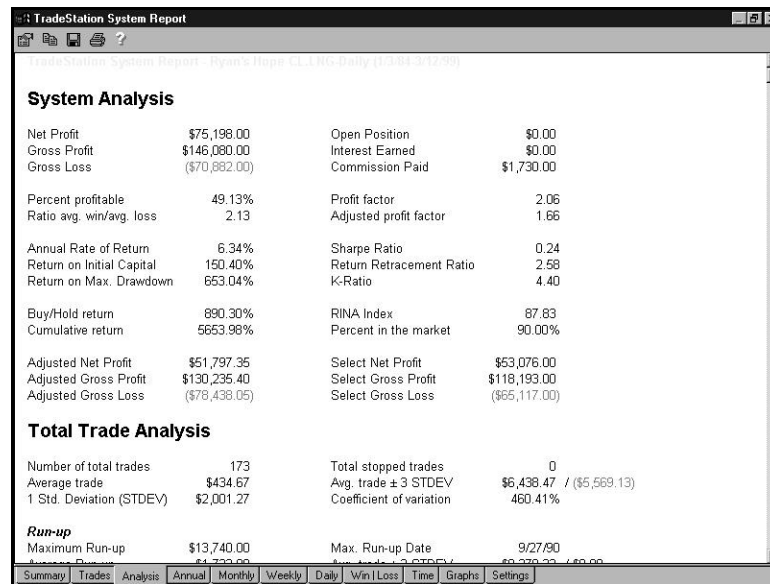
Nbars = 8

The performance summary [Figure 1. TradeStation System Report] shows that Ryan's Hope earned \$75,198 on 173 trades, with 49% of the trades profitable and an average trade of \$434. The average winning trade was 2.13 times as large as the average losing trade, a good ratio with the 49% winning trades. The profit factor of 2.06 means that the system gained \$2.06 for each \$1.00 it lost. The system let profits run by staying in the winning trades for an average of 29 bars, while it cut losses short by exiting the average losing trade in only 10 bars. Long trades and short trades were balanced nicely, as the long side earned \$40,136, while the short side netted \$35,062.

TradeStation System Report			
TradeStation System Report - Ryan's Hope CL 1 RG Daily (1/3/84-3/12/99)			
Performance Summary: All Trades			
Total Net Profit	\$75,198.00	Open position P/L	\$0.00
Gross Profit	\$146,080.00	Gross Loss	(\$70,882.00)
Total # of trades	173	Percent profitable	49.00%
Number winning trades	85	Number losing trades	88
Largest winning trade	\$11,369.00	Largest losing trade	(\$5,765.00)
Average winning trade	\$1,718.59	Average losing trade	(\$805.48)
Ratio avg win/avg loss	2.13	Avg trade (win & loss)	\$434.67
Max consec. Winners	4	Max consec. losers	6
Avg # bars in winners	29	Avg # bars in losers	10
Max intraday drawdown	(\$6,265.00)	Max # contracts held	1
Profit Factor	2.06	Return on account	907.09%
Account size required	\$8,290.00		
Performance Summary: Long Trades			
Total Net Profit	\$40,136.00	Open position P/L	\$0.00
Gross Profit	\$77,134.00	Gross Loss	(\$36,998.00)
Total # of trades	87	Percent profitable	53.00%
Number winning trades	46	Number losing trades	41
Largest winning trade	\$9,795.00	Largest losing trade	(\$5,765.00)

Figure 1. TradeStation System Report

The Analysis section of the performance summary [Figure 2. Analysis] adds several items of interest. The RINA Index, which combines select net profit (the profit if both winning and losing outliers are removed), time in the market, and drawdown) was 87.83, while 30 is considered good; the higher the RINA Index, the more efficient the system. The average run-up per trade was \$1,722, compared to an average drawdown per trade of only \$671. The net profit divided by largest loss ratio was 13.04 (7 or higher is good), and the net profit divided by maximum drawdown was 13.16 (5 or higher is good). The three positive outliers (trades more than 3 standard deviations from average) earned \$27,887 versus the one negative outlier, which lost \$5,765.



The screenshot shows the 'TradeStation System Report' window. The title bar reads 'TradeStation System Report'. The main window title is 'TradeStation System Report - Ryan's Hope CL1 NG-Daily (11-30-12/99)'. The 'Analysis' tab is selected in the bottom navigation bar. The report is divided into three main sections: 'System Analysis', 'Total Trade Analysis', and 'Run-up'.

System Analysis			
Net Profit	\$75,198.00	Open Position	\$0.00
Gross Profit	\$146,080.00	Interest Earned	\$0.00
Gross Loss	(\$70,882.00)	Commission Paid	\$1,730.00
Percent profitable	49.13%	Profit factor	2.06
Ratio avg. win/avg. loss	2.13	Adjusted profit factor	1.66
Annual Rate of Return	6.34%	Sharpe Ratio	0.24
Return on Initial Capital	150.40%	Return Retracement Ratio	2.58
Return on Max. Drawdown	653.04%	K-Ratio	4.40
Buy/Hold return	890.30%	RINA Index	87.83
Cumulative return	5653.96%	Percent in the market	90.00%
Adjusted Net Profit	\$51,797.35	Select Net Profit	\$53,076.00
Adjusted Gross Profit	\$130,235.40	Select Gross Profit	\$118,193.00
Adjusted Gross Loss	(\$78,438.05)	Select Gross Loss	(\$65,117.00)
Total Trade Analysis			
Number of total trades	173	Total stopped trades	0
Average trade	\$434.67	Avg. trade ± 3 STDEV	\$6,438.47 / (\$5,569.13)
1 Std. Deviation (STDEV)	\$2,001.27	Coefficient of variation	460.41%
Run-up			
Maximum Run-up	\$13,740.00	Max. Run-up Date	9/27/90
Average Run-up	\$1,722.00	Avg. trade ± 3 STDEV	\$6,438.47 / (\$5,569.13)

Figure 2. System Analysis

The Win/Loss section of the performance summary indicates that the largest profitable trade earned \$11,369, which is 15% of the total net profit. Coincidentally, 15% is the highest number we want to see in this category. We don't want the biggest winning trade to account for too large a percentage of the total net profit, because there's no guarantee that such a large winner will be repeated in the future. The largest loss, \$5,765, was only 7.67% of net profit.

The Time section of the report shows that the system was in the market 90% of the time, with a longest flat period (time not in the market) of 20 days. We averaged 28 days in each trade — 41.85 days in our winners and 15.13 days in the losers. Perhaps the most important statistic in this section is the average time between equity peaks of 179 days, which seems long. Fortunately, there's a simple solution that generally yields superior results. Trading a system on more than one market or trading more than one system per market usually leads to smaller drawdowns and more frequent new equity highs.

Let's take a look at some of the graphs that are now available in TradeStation 2000i. The bar chart [Figure 3. *Crude Oil*] shows a very profitable long trade, a long entry that turns a small profit, a long entry that suffers a small loss, and a short entry that is going our way so far. That's a good example of a typical distribution of winning and losing trades for this system (and most other trend-following systems). The next chart [Figure 4. *Equity Chart by Bar*] presents a spectacular, steadily rising equity curve. The Underwater Equity Curve [Figure 5. *Underwater Equity Curve*] looks much worse than it really is. The small bars rising above the zero line represent new equity peaks, often referred to as high-water marks. Because the bars are not drawn to scale (they're all the same height), they don't provide a true sense of how much money we're making as we trade the system. That's intentional, as this graph is designed to paint a pessimistic portrait of our system's performance. The equity drawdowns extend down from the zero line, and they ARE drawn to scale, depicting the duration and magnitude of the drawdowns. Here's why they're not really so bad: notice the percent of our equity that the underwater curves represent — an average drawdown of about 5% and a maximum drawdown of about 9%, based on our initial equity of \$50,000.

Figure 6 [Average Profit by Month] depicts a month-by-month average of the system's performance over the approximately 15 years of data we tested. Eleven months generated positive returns, while only one month (October) suffered a negative return. Our system has been very consistent. Total Trades [Figure 7. *Total Trades*] displays each trade at its eventual level of profit or loss. Note the three positive outliers versus only one negative outlier.

Maximum Adverse Excursion [Figure 8. *Maximum Adverse Excursion*] adds a new perspective on our system's history. The vertical axis represents the profit or loss on each trade, and the horizontal axis represents each trade's largest drawdown (its maximum adverse excursion). The triangles stand for winning trades, while the diamonds stand for losing trades. Note that only three winning trades experienced a drawdown of more than \$1,000. That insight can be very valuable as you decide where to place your protective stops. Finally, let's take a look at Maximum Favorable Excursion, a graph of how well each trade was doing at its equity peak [Figure 9. *Maximum Favorable Excursion*]. Note that only two losing trades had enjoyed a run-up of over \$2,000. Consider how you could use that knowledge to earn more money. It might be advantageous, for example, to add to positions as soon as they're in the black by at least \$2,000 because such trades usually continue to accumulate profits.

Ryan's Hope performed well on our futures contract (Crude Oil), so we also tested it on a bellweather stock, IBM (for the long side only). The optimized values for IBM weekly data were as follows:

Length 1 = 40
 Length 2 = 3
 IPS = 4
 TS = 4
 VolSt = 2
 BigProf = 5
 Nbars = 8

Figure 10 [International Business- Weekly] charts a series of winning trades from mid-1996 to early 1999. The Performance Summary [Figure 11] shows a total net profit of \$11,502 (per 100 shares) on 17 trades, with \$.13 per share deducted for slippage and \$.10 per share deducted for commission. 53% of the trades were profitable, and the average trade earned \$676. The average win was 3.21 times as large as the average loss, and the system gained \$3.61 for each \$1.00 it lost.

Ryan's Hope is a good example of a simple entry technique combined with a variety of effective stops to produce a profitable trading system. The markets don't pay us any extra for making our systems more complicated than they need to be.

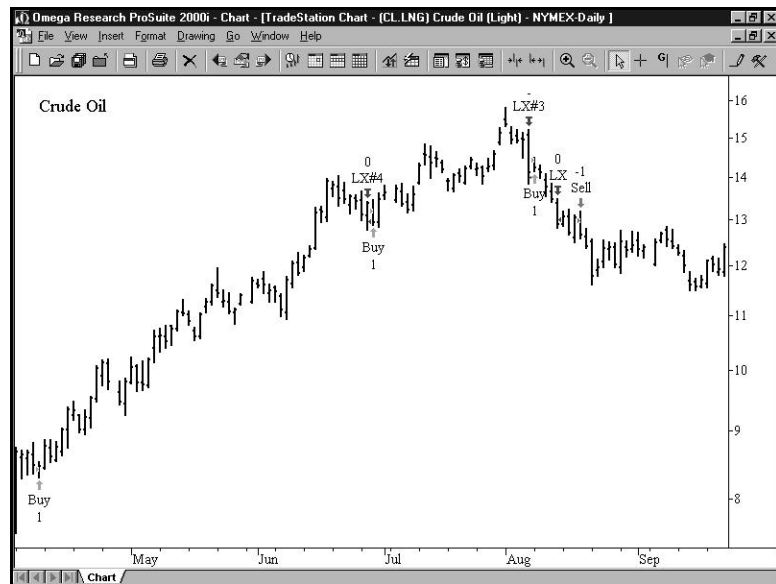


Figure 3. Crude Oil

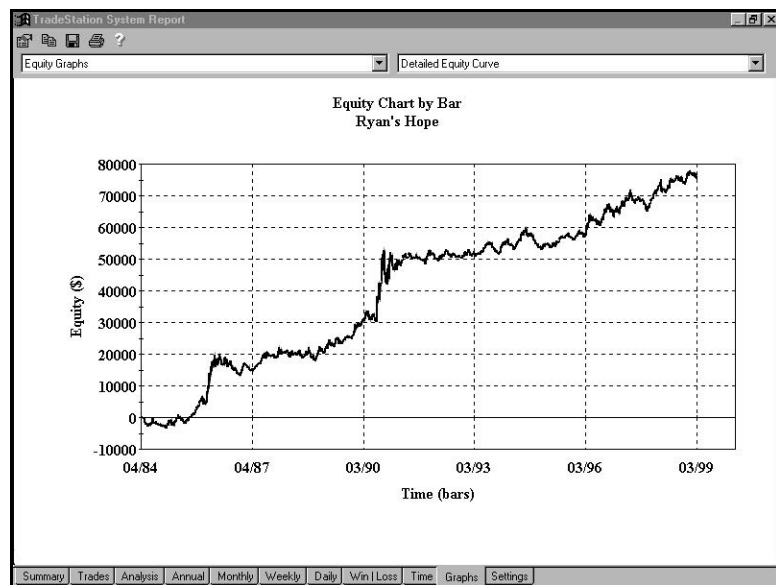


Figure 4. Equity Chart by Bar

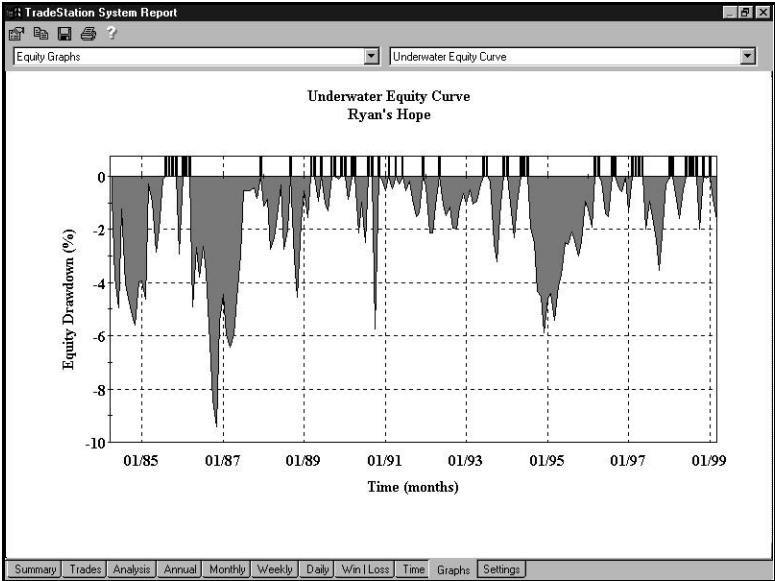


Figure 5. Underwater Equity Curve

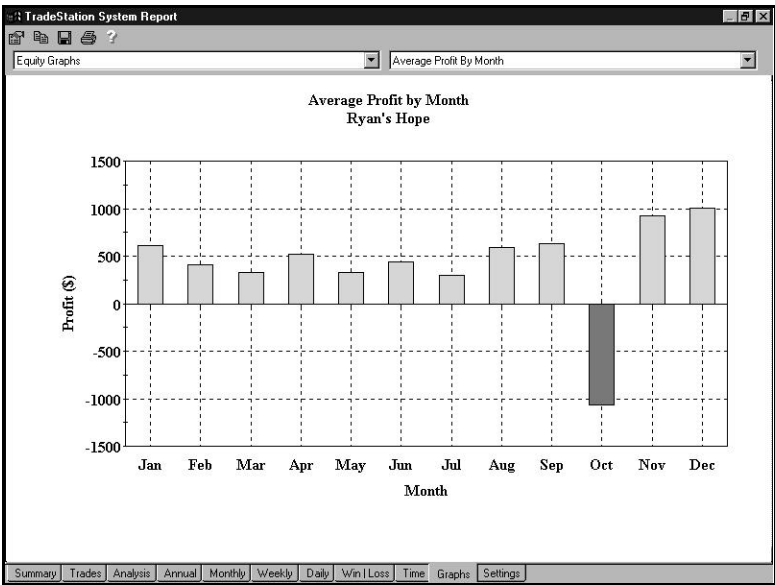


Figure 6. Average Profit by Month

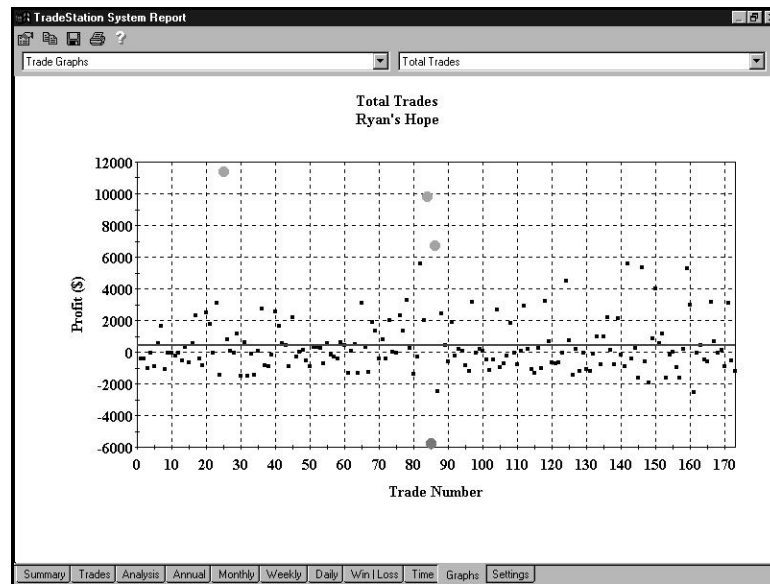


Figure 7. Total Trades

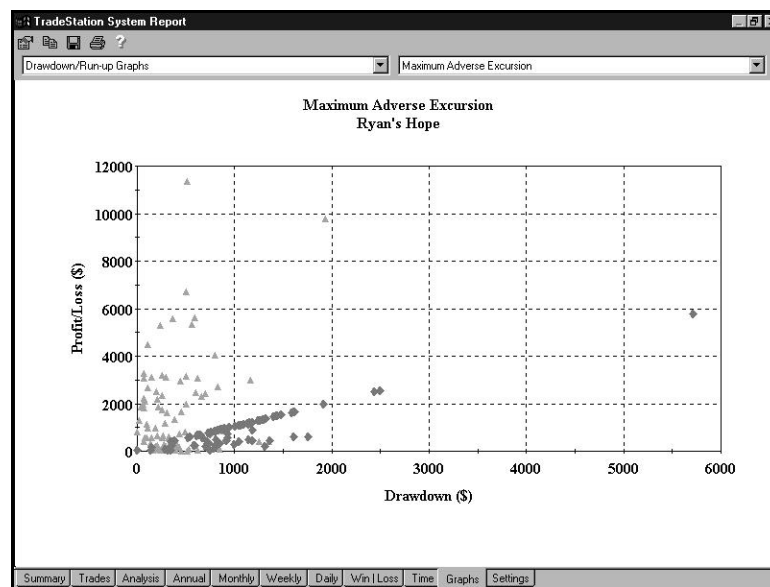


Figure 8. Maximum Adverse Excursion

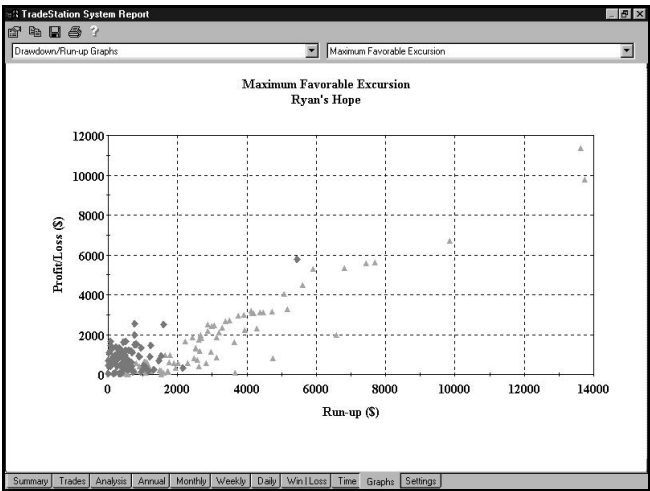


Figure 9. Maximum Favorable Excursion

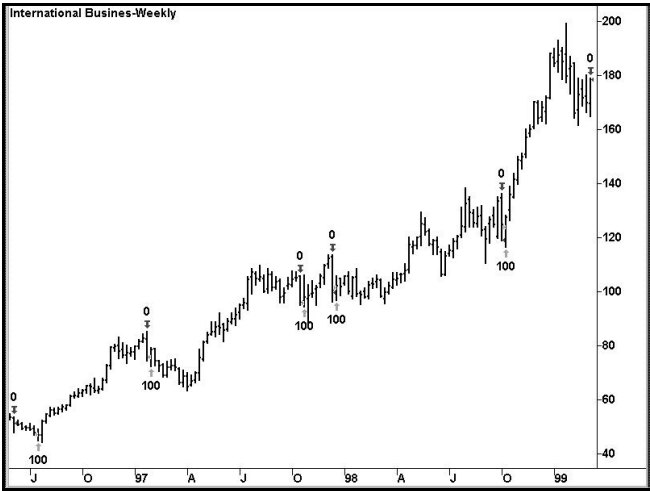


Figure 10. Weekly Chart: IBM

System Report: Performance Summary			
Ryan's Hope International Business-Weekly 01/06/94 - 03/05/99			
Performance Summary: All Trades			
Total net profit	\$ 11502.30	Open position P/L	\$ 0.00
Gross profit	\$ 15911.50	Gross loss	\$ -4409.20
Total # of trades	17	Percent profitable	53%
Number winning trades	9	Number losing trades	8
Largest winning trade	\$ 5439.50	Largest losing trade	\$ -1160.50
Average winning trade	\$ 1767.94	Average losing trade	\$ -551.15
Ratio avg win/avg loss	3.21	Avg trade(win & loss)	\$ 676.61
Max consec. winners	8	Max consec. losers	7
Avg # bars in winners	36	Avg # bars in losers	21
Max intraday drawdown	\$ -4132.30	Max # contracts held	100
Profit factor	3.61	Return on account	278%
Account size required	\$ 4132.30		
Performance Summary: Long Trades			
Total net profit	\$ 11502.30	Open position P/L	\$ 0.00
Gross profit	\$ 15911.50	Gross loss	\$ -4409.20
Total # of trades	17	Percent profitable	53%
Number winning trades	9	Number losing trades	8
Largest winning trade	\$ 5439.50	Largest losing trade	\$ -1160.50
Average winning trade	\$ 1767.94	Average losing trade	\$ -551.15
Ratio avg win/avg loss	3.21	Avg trade(win & loss)	\$ 676.61

Figure 11. Performance Summary: IBM

CHAPTER 8

Stampede Trading System

The price channel breakout, one of our favorite entry techniques, is also one of the simplest. Since the most bullish thing a market can do is go up, and the most bearish thing a market can do is go down, we like to buy a market when it makes a new n-bar high and sell short when a market makes a new n-bar low. This strategy should at least give us an edge over random entries. If we combine price channel breakout entries with logical trade management techniques, we should produce a winning system. We're calling this system STAMPEDE because it includes a pyramiding technique that often resembles a stampede of wild horses breaking out of a corral.

The default value for the length of the price channel is 30 bars; we'll buy one tick above the highest high of the last 30 bars and sell short one tick below the lowest low of the last 30 bars. Perhaps the most interesting fact about this entry technique is that it is at least marginally profitable in most markets and most timeframes as a stop-and-reverse system without any other rules or considerations.

Here's how we hope to improve the price channel breakout entry technique. First, we'll add an initial protective stop so that we won't be risking from our long entry above the price channel all the way down to a tick below the price channel. In the case of selling short, we won't be risking from our entry one tick below the price channel all the way back up to a tick above the price channel. Our initial protective stop will be a multiple of the average true range subtracted from our entry price for a long position and added to our entry price for a short position. The default value for the initial protective stop is 3 average true ranges. (True range refers to the largest of the following: the current bar's high minus the low, the previous bar's close minus the current bar's low, or the current bar's high minus the previous bar's close.)

Second, we'll add a trailing stop. In most cases, we prefer to exit our positions more aggressively than by waiting for prices to penetrate the opposite side of the channel. Our trailing stop will be a multiple of the average true range subtracted from the highest high achieved since our entry into a long position or added to the lowest low reached since our entry into a short position. The default value for the trailing stop is 4 average true ranges.

Third, we'll add a volatility stop. When a market is making a much bigger than normal move against our position, we'll exit before matters become even worse. Our volatility stop will be a multiple of the average true range subtracted from the previous bar's close for a long position and added to the previous bar's close for a short position. The default value for the volatility stop is 3 average true ranges.

Fourth, we'll add a big profit stop. When an open position is earning exceptional profits, we'll move our stop closer to current prices to lock in a significant portion of the profits. We'll define a big profit as a multiple of the average true range added to our entry price for a long position and subtracted from our entry price for a short position. When a long position has earned big profit status, we'll raise our stop to a tick below the lowest low of the last n bars; when a short position has earned a big profit, we'll lower our stop to a tick above the highest high of the last n bars. The default value for the big profit is 8 average true ranges, and the default value for the n -bar high or low is 5 bars back.

Finally, we'll add a pyramiding feature to our price channel breakout system. Instead of limiting our position to one entry per breakout above or below the channel, we'll add to our position on each subsequent breakout in the same direction until our total position reaches a predefined limit of entries. The default value for the maximum number of entries in the same direction is 9.

Defining Our Trading Rules

In this system, we defined long and short entries and exits. We also did some setup work to calculate the price channel and the average true range. The setup, entries, and exits are described next.

Setup

- a. Calculate a 30 bar price channel.
- b. Calculate a 30 bar average true range.

Long Entries

- a. Buy one tick above the highest high of the last 30 bars.

Short Entries

- a. Sell short one tick below the lowest low of the last 30 bars.

Long Exits

- a. Exit at an initial protective stop 3 average true ranges below the entry point into the trade.
- b. Exit at a trailing stop 4 average true ranges below the highest high achieved since entry into the trade.
- c. Exit at a volatility stop 3 average true ranges below the previous close.
- d. Exit one tick below the 5 bar low when open profits exceed 8 times the average true range.

Short Exits

- a. Exit at an initial protective stop 3 average true ranges above the entry point into the trade.
- b. Exit at a trailing stop 4 average true ranges above the lowest low since entry into the position.
- c. Exit at a volatility stop 3 average true ranges above the previous close.
- d. Exit one tick above the 5 bar high when open profits exceed 8 times the average true range.

Pyramiding

Enter an additional position each time the market makes a new 30 bar high or low until the number of open positions reaches the limit. The default value for the limit is 9.

Designing & Formatting

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

Easy Language System Components: Stampede (STAD8: Stampede)

System Inputs (Stampede)

INPUT	DEFAULT	DESCRIPTION
Channel_Length	20	Length, expressed in bars, used to calculate the channel
MaximumEntries	10	The maximum number of consecutive entries in the same direction allowed by the System.
Volatility_Stop_ATRs	2	The number of Average True Ranges that are used to determine the required volatility to place an Exit order
Trailing_Stop_ATRs	4	The number of Average True Ranges that are risked from the highest/lowest price of the position
Protective_Stop_ATRs	3	The number of Average True Ranges that are risked in the position.
Big_Profit_ATRs	7	Number of Average True Ranges used to determine the "Big Profit" level
Big_Profit_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. Chan Break Pyramid
2. ATR Big Profit Stop
3. ATR Protective Stop
4. ATR Trailing Stop
5. ATR Volatility Stop
6. Last Bar Exit

EasyLanguage Signal: Chan Break Pyramid:

Inputs: ChanLength(20), MaximumEntries(10);
 Variables: Counter(1), CC(0), MP(0);

CC = CurrentContracts;
 MP = MarketPosition;

If CC > CC[1] AND MP = MP[1] AND Counter < MaximumEntries Then

```
Counter = Counter + 1;
```

```
If MP <> MP[1] Then
    Counter = 1;
```

```
If Counter < MaximumEntries Then Begin
    Buy Next Bar at Highest(High, ChanLength) + 1 Point Stop;
    Sell Next Bar at Lowest(Low, ChanLength) - 1 Point Stop;
End;
```

Signal Inputs (Chan Break Pyramid):

INPUT	DEFAULT	DESCRIPTION
Channel_Length	20	Length, expressed in bars, used to calculate the channel
MaximumEntries	10	The maximum number of consecutive entries in the same direction allowed by the System.

In addition to the Input above, we define the following variables:

Signal Variables (Chan Break Pyramid):

VARIABLE	DEFAULT	DESCRIPTION
Counter	0	Keeps track of the number of consecutive entries in the same direction
CC	0	Holds the CurrentContracts value
MP	0	Holds the MarketPosition value

Setup

First, CurrentContracts and MarketPosition are calculated and assigned to Variables so that previous values can be referenced as necessary.

```
CC = CurrentContracts;
MP = MarketPosition;
```

Next, we check Current Contracts and Market Position. If Current Contracts is greater than Current Contracts of one bar ago, and the Market Position is the same as one bar ago, and Counter is less than the maximum number of entries allowed, then the Counter is incremented by 1. While the Counter is less than the maximum number of entries allowed, the System will monitor for a price channel breakout. If the Market Position reverses, the Counter is reset.

```
If CC > CC[1] AND MP = MP[1] AND Counter < MaximumEntries Then
    Counter = Counter + 1;
```

```
If MP <> MP[1] Then
    Counter = 1;
```

Entries

While the Counter is below the number of Maximum Entries, a Buy and Sell order are placed at the highest High and lowest Low respectively, plus/minus 1 point.

```
If Counter < MaximumEntries Then Begin
    Buy Next Bar at Highest(High, ChanLength) + 1 Point Stop;
    Sell Next Bar at Lowest(Low, ChanLength) - 1 Point Stop;
End;
```

Easy Language Signal: ATR Big Profit Stop:

**See Chapter 9

Easy Language Signal: ATR Protective Stop:

**See Chapter 9

Easy Language Signal: ATR Trailing Stop:

**See Chapter 9

Easy Language Signal: ATR Volatility Stop:

**See Chapter 9

Easy Language Signal: Last Bar Exit:

**See Chapter 9

Testing & Improving

We tested and optimized our enhanced version of the price channel breakout system on four futures markets: British Pound, Crude Oil, Japanese Yen, and T-Bonds. Then we tested and optimized the system on four stocks: American Express (AXP), Intel (INTL), Microsoft (MSFT), and WalMart (WMT). The encouraging results confirm our belief that a simple entry technique combined with a variety of stop-loss and profit-protect stops can generate consistent and significant profits.

Here's how we conducted our optimizations: First, we chose channel length, maximum entries in the same direction, initial protective stop, trailing stop, volatility stop, big profit, and n-bars as the inputs to optimize. Second, we decided on a tentative default value for each input based on experience and observation. Third, we determined the range of values we wanted to test on each side of the default value. For example, for channel length, we chose a default value of 30 bars, and we tested from 10 to 50 in increments of 10 (two values less than 30 and two values greater than 30). As another example, we chose 9 as the default value for the maximum number of entries and tested from 3 to 15 in increments of 3 (two values less than the default and two values greater than the default). Fourth, we added the command `IncludeSystem:"LastBar"`. This command instructs the system to exit from any open positions on the last bar of the data being tested, so that the open profit or loss will be included as a closed trade in the performance summary.

Following is a list of inputs, defaults, ranges, and increments.

Channel length: 30, testing from 10 - 50 in increments of 10
 Maximum entries: 9, testing from 3 - 15 in increments of 3
 Initial protective stop: 3, testing from 2 - 4 in increments of 1

Trailing stop: 4, testing from 3 - 5 in increments of 1
 Volatility stop: 3, testing from 2 - 4 in increments of 1
 Big Profit: 8, testing from 5 - 11 in increments of 3
 N-bars: 5, testing from 3 - 7 in increments of 2.

We could have conducted these optimizations in one extravagant frenzy of number crunching, but we didn't. Optimizing the seven inputs with the ranges and increments shown above would have required 6,076 tests per market, taking 8 hours and 26 minutes for each market. Our preference, however, (especially when we have a large number of inputs), is to break the tests down into smaller but related segments. First, we optimized the channel length and maximum entries while keeping the default values of the other inputs. Second, we optimized the initial protective stop, the trailing stop, and the volatility stop with the newly optimized values for the channel length and maximum entries and keeping the default values for big profit and n bars back. Third, we optimized big profit and n bars back with the optimized values for all the other inputs. Conducting the optimization in this manner reduced the number of tests from 6,076 to 64 and the time required from 8 hours 26 minutes to 6 minutes 10 seconds. In our experience, conducting optimizations of several inputs by dividing the tests into smaller, related segments yields results very similar to optimizing all the inputs at once, saves a lot of time, and actually reduces the risk of over optimizing (curve fitting) the system to the data.

Now, let's move on to the test results. We tested the four futures markets on daily data from 01/84 to 03/99, deducting \$40 from each trade for slippage and \$10 for commission. The optimized values for the British Pound were a price channel of 10 bars, a maximum number of entries in the same direction of 15, an initial protective stop of 4 ATRs, a trailing stop of 5 ATRs, a volatility stop of 4 ATRs, and a big profit of 8 bars with an n-bar high/low of 5 bars. Applied to the British Pound, the system earned \$919,457 on 1,056 trades from 1/3/84 to 3/5/99. 42.7 percent of the trades were profitable, with an average trade of \$870. The profit factor (dollars won per dollar lost) was 2.04, and the ratio of average win to average loss was 2.73. 18 positive outliers (trades more than 3 standard deviations from the average trade) earned \$276,853, and there were no negative outliers. The system posted a net profit/largest loss ratio of 150.55 (7 or higher is considered good) and a net profit/maximum drawdown ratio of 151.79 (5 or higher is considered good). The Equity Curve was strong from trade 1 to about trade 650, but it failed to make a new high from trade 651 to trade 1,056. [Figure 1. British Pound Equity Curve]

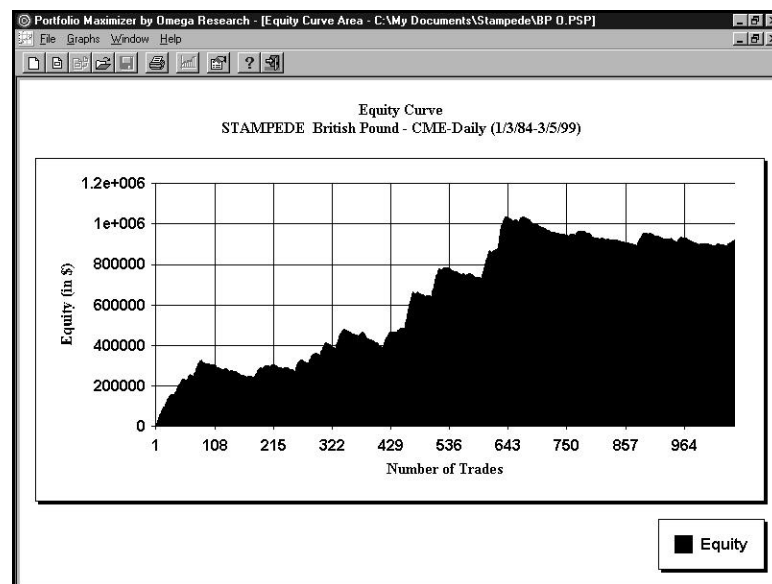


Figure 1. British Pund Equity Curve

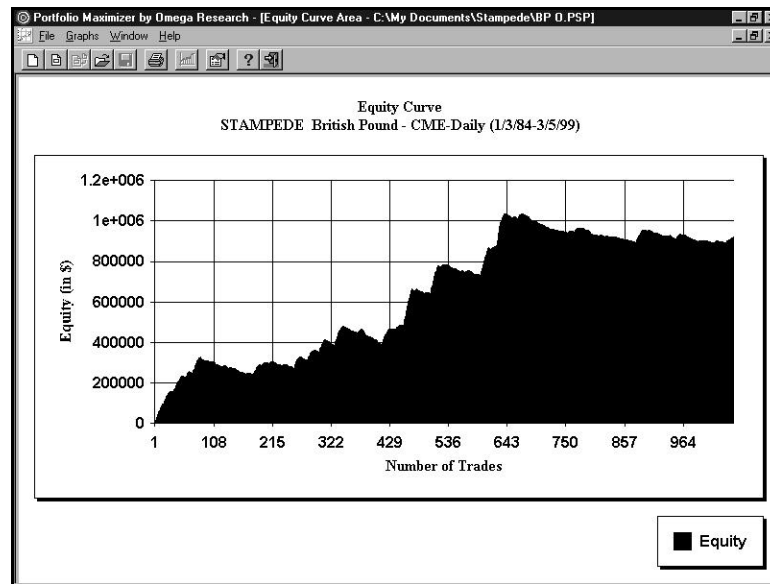


Figure 2. Crude Oil Equity Chart by Bar

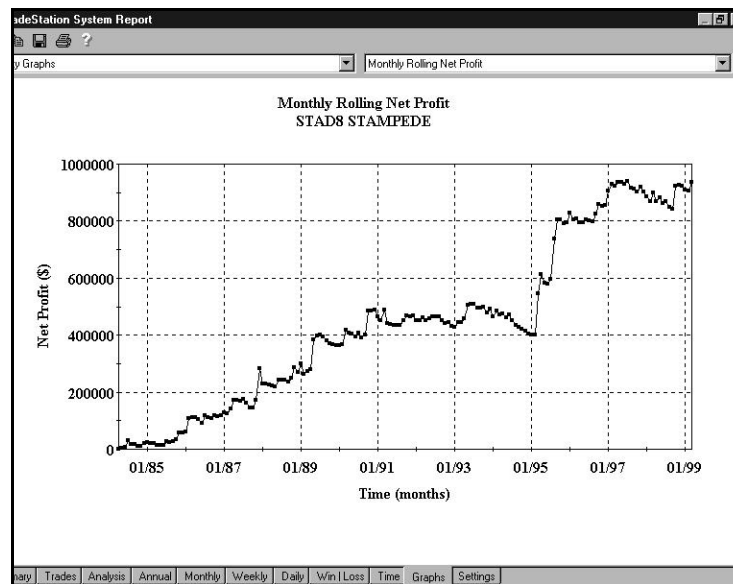


Figure 3. Japanese Yen Monthly Rolling Net Profit

For Crude Oil, the optimized values were a price channel of 20 bars, a maximum number of entries in the same direction of 15, an initial protective stop of 4 ATRs, a trailing stop of 5 ATRs, a volatility stop of 4 ATRs, and a big profit of 11 bars with an n-bar high/low of 3 bars. The system earned \$625,943 on 772 trades from 1/3/84 to 3/5/99. 52.72 percent of the trades were profitable, with an average trade of \$810. The profit factor was 2.52, and the ratio of average win to average loss was 2.26. 16 positive outliers earned \$221,482, and 2 negative outliers lost \$17,241, yielding a net outlier profit of \$204,241. The net profit/largest loss ratio was 69.77 (7 is good); and the net profit/maximum drawdown ratio was 70.17 (5 is good). The bar-by-bar equity chart (Equity Chart by Bar) is exceptionally strong, continuing to post new equity highs near the end of the test period). [Figure 2. *Crude Oil Equity Chart by Bar*]

The optimized values for the Japanese Yen were a price channel of 20 bars, a maximum number of entries in the same direction of 15, an initial protective stop of 3 ATRs, a trailing stop of 4 ATRs, a

volatility stop of 4 ATRs, and a big profit of 8 bars with an n-bar high/low of 5 bars. The system earned \$954,903 on 828 trades from 1/3/84 to 3/5/99. 49.40 percent of the trades were profitable, with an average trade of \$1,153. The profit factor was 2.28, and the ratio of average win to average loss was 2.34. 17 positive outliers earned \$328,725, and there were no negative outliers. The net profit/largest loss ratio was 124.80, and the net profit/maximum drawdown ratio was 70.17. The graph of Monthly Rolling Net Profit (a "snapshot" of the system's performance marked-to-market on the last trading day of each month) is exceptional, rising at about a 45 degree angle for the last 15 years. [Figure 3. Japanese Yen Monthly Rolling Net Profit]

For Treasury Bonds, the optimized values were a price channel of 20 bars, a maximum number of entries in the same direction of 12, an initial protective stop of 3 ATRs, a trailing stop of 5 ATRs, a volatility stop of 2 ATRs, and a big profit of 11 bars with an n-bar high/low of 3 bars. The system earned \$995,430 on 672 trades from 1/3/84 to 3/5/99. 49.26 percent of the trades were profitable, with an average trade of \$1,481. The profit factor was 2.54, and the ratio of average win to average loss was 2.61. 6 positive outliers earned \$98043, and there were no negative outliers. The net profit/largest loss ratio was 141.16, and the net profit/maximum drawdown ratio was 142.16. The Equity Curve shows that the system reached 19 new equity peaks in the 15-year test period. [Figure 4. Treasury Bonds Equity Curve]

Next, let's see how the system performed on the four stocks we chose. We tested the four stocks on daily data from 01/84 to 03/99. Each trade purchased 100 shares, and we deducted \$.13 per share for slippage and \$.10 per share for commission. The optimized values for American Express were a price channel of 10 bars, a maximum number of entries in the same direction of 15, an initial protective stop of 4 ATRs, a trailing stop of 5 ATRs, a volatility stop of 4 ATRs, and a big profit of 11 bars with an n-bar high/low of 5 bars. The system earned \$120,481 on 521 trades from 1/3/84 to 3/5/99. 51.63 percent of the trades were profitable, with an average trade of \$231. The profit factor was 3.13, and the ratio of average win to average loss was 2.93. 10 positive outliers earned \$27,162, and there were no negative outliers. The net profit/largest loss ratio was 66.47 (remember that 7 or higher is good); the net profit/maximum drawdown ratio was 66.47 (5 or higher is considered good). The graph Average Profit by Month can be viewed as a monthly seasonal study for the system. Averaging the 15 Januaries, the 15 Februaries, etc., shows that 10 months were profitable and only 2 on average were losers. [Figure 5. American Express Average Profit by Month]

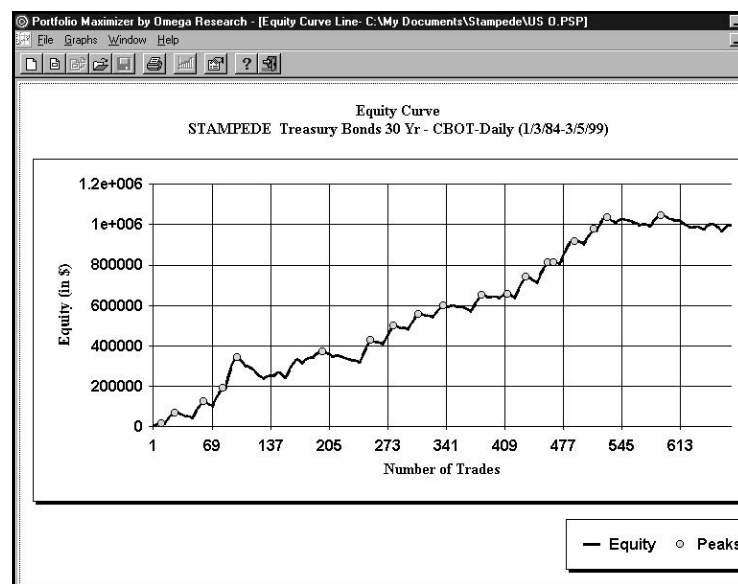


Figure 4. Treasury Bonds Equity Curve

For Intel, the optimized values were a price channel of 20 bars, a maximum number of entries in the same direction of 15, an initial protective stop of 2 ATRs, a trailing stop of 4 ATRs, a volatility stop of 2 ATRs, and a big profit of 8 bars with an n-bar high/low of 7 bars. The system earned \$100,016 on 376 trades from 1/3/84 to 3/5/99. 52.13 percent of the trades were profitable, with an average trade of \$266. The profit factor was 4.18, and the ratio of average win to average loss was 3.84. 12 positive outliers earned \$43,374, and there were no negative outliers. The net profit/largest loss ratio was 73.57; and the net profit/maximum drawdown ratio was 73.57. Maximum Adverse Excursion offers an interesting view of a system's trades. The vertical axis represents profit (or loss) in dollars; the horizontal axis represents the largest drawdown for each trade. The upward-pointing triangles signify winning trades, and the downward-pointing triangles signify losing trades. The Maximum Adverse Excursion chart for Intel shows that no winning trades experienced a drawdown of \$600 or more, and that most winning trades experienced a drawdown of \$350 or less. [Figure 6. Intel Maximum Adverse Excursion]

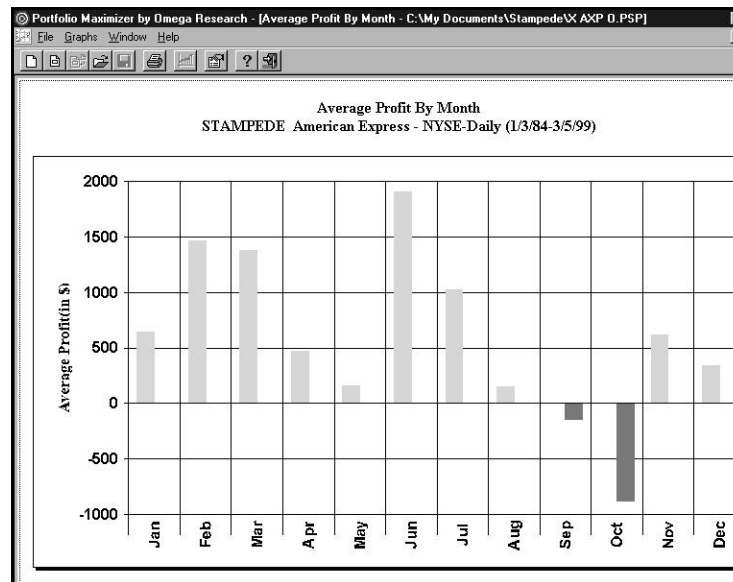


Figure 5. American Express Average Profit by Month

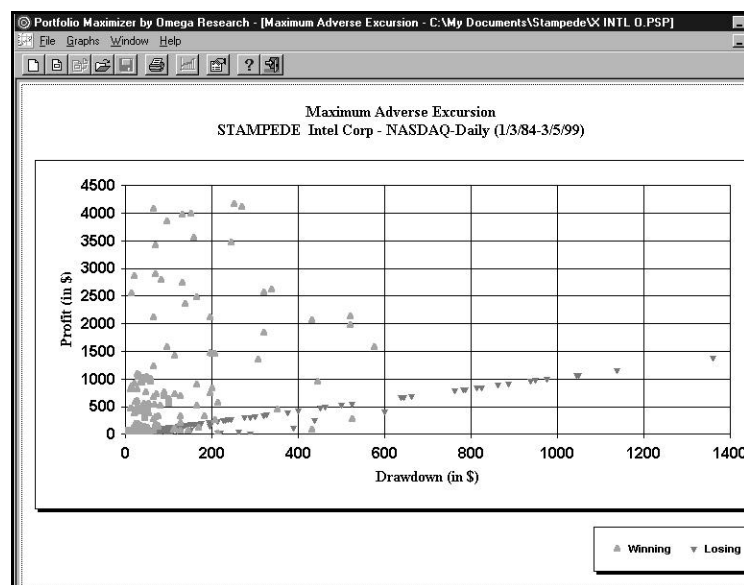


Figure 6. Intel Maximum Adverse Excursion

The optimized values for Microsoft were a price channel of 30 bars, a maximum number of entries in the same direction of 15, an initial protective stop of 2 ATRs, a trailing stop of 5 ATRs, a volatility stop of 4 ATRs, and a big profit of 8 bars with an n-bar high/low of 3 bars. The system earned \$122,184 on 313 trades from 3/13/86 to 3/5/99. 67.41 percent of the trades were profitable, with an average trade of \$390. The profit factor was 10.40, and the ratio of average win to average loss was 5.03. 11 positive outliers earned \$46,250, and there were no negative outliers. The net profit/largest loss ratio and the net profit/maximum drawdown ratio were both 148.10. The Underwater Equity Curve is designed to provide an extremely pessimistic view of a system's performance. The vertical axis shows drawdown as a percentage of equity (we set the account size to \$50,000). The vertical bars extending above zero represent new equity highs. Obviously, they are not drawn to scale. Each bar represents a greater amount of money than the previous one. The horizontal scale shows time (in this case, based on daily bars). The shaded areas below zero ("under water" represent the duration and magnitude of drawdowns. Visually, it appears to be pretty bad, until you notice that the worst drawdown in the 15-year history of trading this system on this stock was only 5%. The system made 28 new "high-water marks" (new equity highs) during the test period. [Figure 7. Microsoft Underwater Equity Curve]

For Wal Mart, the optimized values were a price channel of 20 bars, a maximum number of entries in the same direction of 15, an initial protective stop of 2 ATRs, a trailing stop of 5 ATRs, a volatility stop of 3 ATRs, and a big profit of 8 bars with an n-bar high/low of 5 bars. The system earned \$56,637 on 448 trades from 1/3/84 to 3/5/99. 53.13 percent of the trades were profitable, with an average trade of \$126. The profit factor was 3.06, and the ratio of average win to average loss was 2.70. 15 positive outliers earned \$32,512, and there were no negative outliers. The net profit/largest loss ratio was 62.07 (7 is good); the net profit/maximum drawdown ratio was also 62.07 (5 is good). Maximum Favorable Excursion is, of course, the opposite of Maximum Adverse Excursion, which we looked at for Intel. The horizontal axis represents profit (or loss) in dollars, and the vertical axis represents each trade's maximum run-up in dollars. Triangles that point upward are winning trades, and triangles that point downward are losing trades. This graph shows that no trades with a run-up of \$500 or more turned out to be losers. Therefore, adding shares to a position with open profits of \$500 or more might be a wise tactic. [Figure 8. Wal Mart Maximum Favorable Excursion]

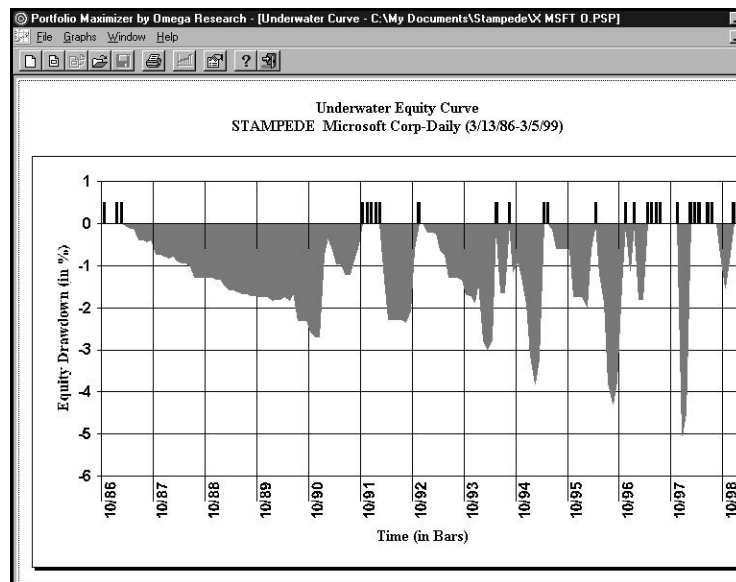


Figure 7. Microsoft Underwater Equity Curve

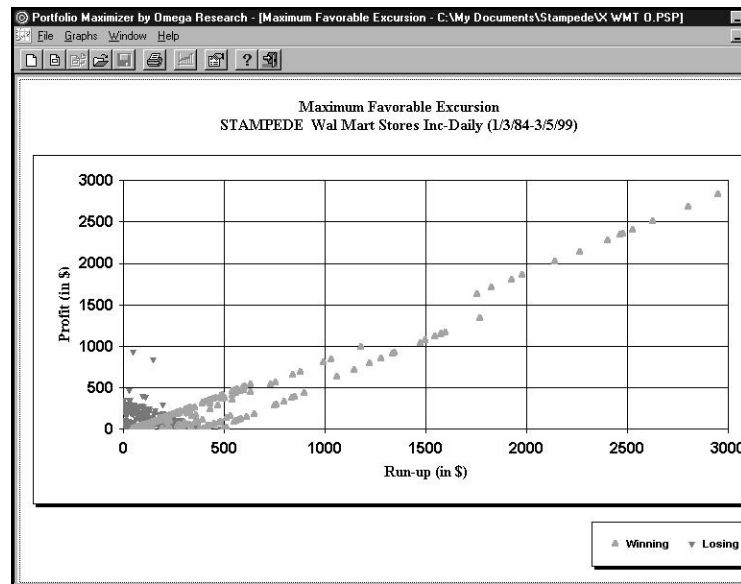


Figure 8. Wal Mart Maximum Favorable Excursion

Suggestions for Improving

We are very pleased with our price channel breakout's exits, so we'll focus on entries as we try to improve the system. As it stands, the system buys a new 30-bar high (default value) and sells short on a new 30-bar low. Perhaps the entries would be more reliable if we added a volume or volatility requirement. For example, we could buy on the next open after a new 30-bar high only if the volume on that bar was greater than the average volume of the last 10 bars. As another example, we could sell short on the open after a new 30-bar low only if the true range of that bar was greater than the average true range of the last 10 bars. Maybe some STAD Club members will test these suggestions and post their results on our STAD Club Forum. We look forward to seeing your improvements to our price channel breakout system!

THIS PAGE LEFT BLANK INTENTIONALLY

CHAPTER 9

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:

- STAD8: Ryan's Hope
- STAD8: Stampede

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.

In addition to the Inputs above, we define the following Variables:

Signal Variables (ATR Big Profit Stop):

VARIABLE	DEFAULT	DESCRIPTION
ATRVal	0	Holds the value of the Average True Range multiplied by the Big Profit amount
PosHL	0	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 Protective Stop:

Applicable Systems in this issue:

- STAD8: Ryan's Hope
- STAD8: Stampede

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

In addition to the Inputs above, we define the following Variables:

Signal Variables (ATR Protective Stop):

VARIABLE	DEFAULT	DESCRIPTION
ATRVal	0	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:

- STAD8: Ryan's Hope
- STAD8: Stampede

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

In addition to the Inputs above, we define the following Variables:

Signal Variables (ATR Trailing Stop):

VARIABLE	DEFAULT	DESCRIPTION
PosHigh	0	Holds the value of the position High
PosLow	0	Holds the value of the position Low
ATRVAl	0	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: ATR Volatility Stop:

Applicable Systems in this issue:

- STAD8: Ryan's Hope
- STAD8: Stampede

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	4	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

In addition to the Inputs above, we define the following Variables:

Signal Variables (ATR Volatility Stop):

VARIABLE	DEFAULT	DESCRIPTION
ATRVal	0	Holds the value of the Average True Range multiplied by the number of VolatilityATRs

Setup

In the Setup portion of the 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: Last Bar Exit:

Applicable Systems in this issue:

- STAD8: Ryan's Hope
- STAD8: Stampede

```
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;
```

THIS PAGE LEFT BLANK INTENTIONALLY

APPENDIX A

Volume in Review

In each issue of STAD Club, we try to correct any errors from the previous volume and/or answer questions from our club members. If you come across an error or have a question about systems trading, please write to stadclub@omegaresearch.com. Although we are unable to reply individually to all of our STAD Club e-mail, we will try to include your question and an answer in the next volume. Also, keep in mind that you can submit your EasyLanguage questions to our EasyLanguage Support Department at easylang@omegaresearch.com.

To discuss the current STAD Club volume or to get feedback on other systems trading issues, you can post your question or comment at our STAD Club Forum: www.omegaresearch.com.

Does anyone know of any books on the theory of system design that take up the issue of backtesting requirements? I mean the fact that the performance of trading systems seems to degrade when tested on large numbers of charts, yet such testing would seem precisely what is necessary to establish that a system is profitable in itself, and not simply because a bull market makes almost any system profitable. For my part, I'm troubled by the explanation that not every system works on every market, that there's no holy grail, etc. No, there isn't, but if a system does in fact work systematically, then it ought to work on a class of charts, and not just one or two.

Here's a short list of good books that include useful information on systems trading and backtesting:

The Business One Irwin Guide to Trading Systems, Bruce Babcock

Computer Analysis of the Futures Market, Chuck LeBeau and David Lucas

Schwager on Futures: Technical Analysis, Jack Schwager

Trading As A Business, Charlie Wright

I'm an EasyLanguage novice. Can someone help with a statement that limits a daytrading system to one trade per day?

If you have an intraday chart (minute or tick bars), you can write the following:
 If date <> date[1] then value1 = 0;
 If MarketPosition = 1 then value1 = 1;
 If value1 = 0 then Begin {your instructions to buy here} end;

In TradeStation 2000i, the breakeven stop does not work properly, i.e. if I set a BE stop to \$2,500, the system exits at that profit rather than at breakeven. What's wrong?

In TradeStation 2000i, all stops are based on EasyLanguage. If you open the BreakEven signal in the PowerEditor, it should match this:

```
*****
Description: BreakEven Stop (Long Exit) Provided By: OmegaResearch, Inc. ©
Copyright 1999
*****
Inputs: BreakEvenFloor(500), PositionBasis(True);
Variables: ProfitCalc(0), OrderPrice(0);
IF PositionBasis Then ProfitCalc = MaxPositionProfit - Commission
Else ProfitCalc = MaxContractProfit - Commission;
OrderPrice = EntryPrice + Commission;
IF ProfitCalc >= BreakEvenFloor AND MarketPosition = 1
Then ExitLong ("BkEv") Next Bar at OrderPrice Stop;
```

INDEX

A

ActivityBar Price Distribution System	61
Additional Educational Services	6
EasyLanguage Resource Center	6
Workshops	6
Appendix A	109
ATR Big Profit Stop	101
ATR Protective Stop	103
ATR Trailing Stop	104
ATR Volatility Stop	106

B

Benefits of Systems Trading	8
-----------------------------------	---

C

Common Exits	101
Contents at a Glance	6
Creating New Systems	12

E

EasyLanguage Resource Center	6
EasyLanguage Support Department	7
Escalator Trading System	19
Exits	101

F

Fixed Ratio Money Management	69
Forecasting	34 - 43

G

Getting Ideas For Systems	9
Getting Started	6

I

Importing your work from older versions	16
---	----

J

Jack-in-the-Box Trading System	45
Jones, Ryan	69, 75

L

Last Bar Exit	107
---------------------	-----

M

Money Management	69
------------------------	----

O

Obtaining Technical Support	7
EasyLanguage Support Department	7
STAD Club E-mail Address	8

P

Price Distribution	61
ProbabilityMaps	35
Pyramiding	14, 91

R

Ryan Jones	69, 75
Ryan's Hope Trading System	77

S

Space to the right	33
STAD Club e-mail address	8
Stampede Trading System	89
Stops	101
SystemBuilder	11

V

Volume in Review	109
------------------------	-----

W

Workshops	6
-----------------	---