CHAPTER 1

Fundamentals—What Is EasyLanguage?

When you code (slang for writing your ideas into a programming language) an analysis technique, you are directing the computer to follow your instructions to the T. A computer program is nothing but a list of instructions. A computer is obedient and speedy, but it is only as smart as its programmer. In addition, the computer requires that its instructions be in an exact format. A programmer must follow certain syntax rules.

EasyLanguage is the medium used by traders to convert a trading idea into a form that a computer can understand. Fortunately for nonprogrammers, EasyLanguage is a high-level language; it looks like the written English language. It is a compiled language; programs are converted to computer code when the programmer deems it necessary. The compiler then checks for syntactical correctness and translates your source code into a program that the computer can understand. If there is a problem, the compiler alerts the programmer and sometimes offers advice on how to fix it. This is different from a translated language, which evaluates every line as it is typed.

All computer languages, including EasyLanguage, have several things in common. They all have:

- *Reserved words*. Words that the computer language has set aside for a specific purpose. You can use these words only for their predefined purposes. Using these words for any other purpose may cause severe problems. (See the list of reserved words in Appendix B.)
- *Remarks*. Words or statements that are completely ignored by the compiler. Remarks are placed in code to help the programmer, or other people who may reuse the code, understand what the program is designed to do. EasyLanguage also utilizes *skip words*. These words are included in a statement to make the programming easier to read. For example, **Buy on next bar at myPrice stop** is the same as **Buy next**

bar myPrice stop. The words "on" and "at" are completely ignored. (See the list of skip words in Appendix B.)

- *Variables*. User-defined words or letters that are used to store information.
- *Data types*. Different types of storage; variables are defined by their data types. EasyLanguage has three basic data types: Numeric, Boolean, and String. A variable that is assigned a numeric value, or stored as a number, would be of the Numeric type. A variable that stores a true or false value would be of the Boolean type. Finally, a variable that stores a list of characters would be of the String type.

VARIABLES AND DATA TYPES

Programmers must understand how to use variables and their associated data types before they can program anything productive. Let's take a look at a snippet of code.

```
mySum = 4  + 5 + 6;
myAvg = MySum/3;
```

The variables in this code are **mySum** and **myAvg** and they are of the Numeric data type; they are storage places for numbers. EasyLanguage is liberal concerning variable names, but there are a few requirements. A variable name cannot

- Start with a number or a period (.)
- Be a number
- Be more than 20 alphanumeric characters long
- Include punctuation other than the period (.) or underscore (_)

Correct	Incorrect
myAvg	1MyAvg
mySum	.sum
sum	val+11
val1	the//sum
the.sum	my?sum
my_val	1234

Variable naming is up to the style of the individual programmer. EasyLanguage is not case sensitive (you can use uppercase or lowercase letters in the variable names). (Note: This is our preference—it may not be everybody's.) Lowercase letters are preferred for names that contain only one syllable. For variable names that have more than one syllable, we begin the name with a lowercase letter and then capitalize the beginning of each subsequent syllable.

```
sum, avg, total, totalSum, myAvg, avgValue, totalUpSum, totDnAvg
```

Still referring to the previous snippet of code, **mySum** is assigned the value of 15 and **myAvg** is assigned the value of 15/3, or 5. If a variable name is created, it must be

declared ahead of time. The declaration statement defines the initial value and data type of the variable. The compiler needs to know how much space to reserve in memory for all variables. The following code is a complete EasyLanguage program. (Note: Most of the code that you will see in this book will be particular to EasyLanguage and will probably not work in any other language.)

```
Vars: mySum(0),myAvg(0);
mySum = High  + Low + Close;
myAvg = mySum/3;
```

The Vars: (or Variables:) statement tells the computer what variables are being declared and initialized. We declare the variables by simply listing them in the Vars statement, and we initialize them by placing an initial value in parentheses following the variable name. In this case, mySum and myAvg are to be equal to zero. EasyLanguage is smart enough to realize that these variables should be of the Numeric data type, since we initialized them with numbers. Variable names should be self-descriptive and long enough to be meaningful. Which of the following is more self-explanatory?

```
mySum = High+Low+Close;    or     k  = High  + Low + Close;
myAvg = mySum/3;    or     j = k/3;
BuyPt = Close   + myAvg;  or     l = Close+j;
```

Variables of Boolean and String types are declared in a similar fashion.

```
Vars: myCondition(false),myString("abcdefgh");
```

The variable **myCondition** was initialized to **false**. The word "false" is a reserved word that has the value of zero. This word cannot be used for any other purpose. The variable **myString** was initialized to "**abcdefgh**." Sometimes you will need to use a variable for temporary purposes, and it is difficult to declare and initialize all of your variables ahead of time. In the case of a temporary variable (one that holds a value for a short period of time), EasyLanguage has declared and initialized several variables for your use; **value0** through **value99** have been predefined and initialized to zero and are ready for usage in your programs. The following is a complete EasyLanguage program:

```
value1 = High + Low + Close;
value2 = (High + Low)/2.0;
```

Notice there isn't a Vars statement. Since **value1** and **value2** are predefined, the statement isn't needed. You have probably noticed the semicolon (;) at the end of each line of code. The semicolon tells the compiler that we are done with this particular instruction. In programming jargon, instructions are known as statements. Statements are made up of expressions, which are made up of constants, variables, operators, functions, and parentheses. Some languages need a termination symbol and others do not. EasyLanguage needs the statement termination symbol. Remember to put a semicolon at the end of each line to prevent a syntax error.

Inputs are similar to variables. They follow the same naming protocol and are declared and initialized. However, an input remains constant throughout an analysis technique. An input cannot start a statement (a line of instruction) and cannot be modified within the body of the code. One of the main reasons for using inputs is that you can change input values of applied analysis techniques without having to edit the actual EasyLanguage code. Inputs would be perfect for a moving average indicator. When you plot this indicator on a chart, you simply type the length of the moving average into the Input box of the dialog box. You don't want to have to go back to the moving average source code and change it and then verify it. Also, when used in trading strategies, inputs allow you to optimize your strategies. This is discussed in Chapter 5.

Notice how inputs and variables are declared in similar style.

```
Inputs: length1(10),length2(20),flag(false);
Vars:  myLength1(10),myAvgVal(30);
```

However, notice how they are used differently in coding.

```
Variables
myLength1 = myAvgVal + myLength1;     {Correct}
Inputs
length1 = myAvgVal + length1;   {Incorrect}
myLength1 = length1*2;     {Correct}
```


Variables can start a statement and can be assigned another value. Since inputs are constants and cannot be assigned new values, they cannot start a statement.

In a strongly typed language, such as C, Pascal, or C++, if you assign a real value such as 3.1456 to an integer typed variable, the decimal portion is truncated and you end up with the number 3. As we all know, precision is important when it comes to trading, so EasyLanguage includes only one Numeric type. All numbers are stored with a whole and a fractional part. In the old days, when CPUs were slow, noninteger arithmetic took too much time and it was advised to use integer variables whenever possible.

OPERATORS AND EXPRESSIONS

Previously, we discussed statements and how they are made up of expressions. To review, an expression consists of a combination of identifiers, functions, variables, and values, which result in a specific value. Operators are a form of built-in functions and come in two forms: unary and binary. A binary operator requires two operands, whereas a unary operator requires only one. Most of your dealings with operators in EasyLanguage will be of the binary variety. Some of the more popular ones are: + − / * < = > >= <= <> AND OR. These binary operators can be further classified into two more categories: arithmetic and logical.

Expressions come in three forms: arithmetic, logical, and string. The type of operator used determines the type of expression. An arithmetic expression includes + − / ∗, whereas a logical or Boolean expression includes < = > >= <= <> AND OR.

Arithmetic Expressions **Logical Expressions**

```
myValue = myValue + 1;      myCondition1 = sum > total;
myValue = sum - total;      myCondition1 = sum <> total;
myResult = sum*total+20;    cond1 = cond1 AND cond2
```

Arithmetic expressions always result in a number, and logical expressions always result in true or false. True is equivalent to 1, and False is equivalent to 0. String expressions deal with a string of characters. You can assign string values to string variables and compare them.

```
myName1 = "George Pruitt";
myName2 = "John Hill";
cond1 = (myName1 <> myName2);
myName3 = myName1 + " " + myName2;
```

Concatenation occurs when two or more strings are added together. Basically, you create one new string from the two that are being added together.

Precedence of Operators

It is important to understand the concept of precedence of operators. When more than one operator is in an expression, the operator with the higher precedence is evaluated first, and so on. This order of evaluation can be modified with the use of parentheses. EasyLanguage's order of precedence is as follows:

1. Parentheses
2. Multiplication or division
3. Addition or subtraction
4. <, >, =, <=, >=, <>
5. AND
6. OR

Here are some expressions and their results:

```
1. 20 - 15/5      equals 17 not 1
   20 - 3         division first, then subtraction
2. 10 + 8/2       equals 14 not 9
   10 + 4         division first, then addition
```

```
3. 5 * 4/2          equals 10
   20/2             division and multiplication are equal
4. (20 - 15)/5      does equal 1
   5/5              parentheses overrides order
5. (10 + 8)/2       equals 9
   18/2             parentheses overrides order
6. 6 + 2 > 3        true
   8 > 3
7. 2 > 1 + 10       false
   2 < 11
8. 2 + 2/2 * 6      equals 8 not 18
   2 + 1 * 6        division first
   2 + 6            then multiplication
   8                then addition
```

These examples have all the elements of an arithmetic expression—numeric values and operators—but they are not complete EasyLanguage statements. An expression must be part of either an assignment statement—**myValue = mySum + myTot**—or a logical statement—**cond1 = cond2 OR cond3**.

The overall purpose of EasyLanguage is to translate an idea and perform an analysis on a price data series over a specific time period. A price chart consists of bars built from historical price data. Each individual bar is a graphical representation of the range of prices over a certain period of time. A 5-minute bar would have the opening, high, low, and closing prices of an instrument over a five-minute time frame. A daily bar would graph the range of prices over a daily interval. Bar charts are most often graphed in an Open, High, Low, and Close format. Sometimes the opening price is left off. A candlestick chart represents the same data but in a different format. It provides an easier way to see the relationship between the opening and closing prices of a bar chart. Other bar data such as the date and time of the bar's close, volume, and open interest is also available for each bar. Since EasyLanguage works hand-in-hand with the charts that are created by TradeStation, there are many built-in reserved words to interface with the data. These reserved words were derived from commonly used verbiage in the trading industry. You can interface with the data by using the following reserved words. (Note: Each word has an abbreviation that can be used as a substitute.)

Reserved Word	Abbreviation	Description
Date	D	Date of the close of the bar.
Time	T	Time of the close of the bar.
Open	O	Open price of the bar.
High	H	High price of the bar.
Low	L	Low price of the bar.
Close	C	Close price of the bar.
Volume	V	Number of contracts/shares traded.
OpenInt	OI	Number of outstanding contracts.

If you wanted to determine that the closing price of a particular instrument was greater than its opening price, you would simply type "Close > Open," or "C > O." The beauty of EasyLanguage is its ability to have all of the data of an instrument at your fingertips. The reserved words that we use to access the different prices of the current bar are also used to access historical data. You do this by adding an index to the reserved word. The closing price of yesterday would be: Close [1]. The closing price two days ago would be: Close [2], and so on. The number inside the bracket determines the number of bars to look back. The larger the number, the further you go back in history. If you wanted to compare today's closing price with the closing price 10 days prior, you would type "Close > Close[10]."

Before we move on, we should discuss how TradeStation stores dates and times. January 1, 2001, is stored as 1010101 instead of 20010101 or 010101. When the millennium changed, instead of incorporating the century into the date, TradeStation simply added a single digit to the year. The day after 991231 was 1000101 according to TradeStation. Time is stored as military time. For example, one o'clock in the afternoon is 1300 and one o'clock in the morning is 100.

After that brief introduction to the world of programming, let's go ahead and set up and program a complete trading strategy. The EasyLanguage Editor is where all of the magic takes place. This is your interface between your ideas and their applications. Your entire coding takes place here, and a thorough understanding of this editor will reduce headaches and increase productivity. In the older TradeStations 4.0 and 2000i, and now in versions greater than 8.8, the PowerEditor (a.k.a. TradeStation Development Environment) is basically a stand-alone application; it is an independent program and can be run with or without TradeStation. In versions 6.0 through 8.7, the PowerEditor is more of a component program; it runs within the confines of TradeStation.

TRADESTATION 2000I VS TRADESTATION 9.0

As of the original writing of this book in 2002, there were two versions of TradeStation being used: 2000i and 6.0. At this time there are basically three versions available (2000i, 8.x, and 9.0), but by the time this book is published there may be just two. TradeStation 2000i is no longer supported, and the majority of users have migrated to the later versions, but there are still some holdouts. In some countries where data is not readily available via an Internet connection, 2000i is still being used. For this reason, the remainder of this chapter will be broken into two main sections: 2000i and version 9.0. (Beyond this chapter, however, we will include the EasyLanguage code for 2000i in Appendix A. We will concentrate on version 9.0 in the remaining chapters, as it is the latest iteration.) TradeStation 9.0 is TradeStation Group's (formerly known as Omega Research) current all-inclusive trading tool. Everything that you need to design, test, monitor, and execute an analysis technique is in one slick and complete package. TradeStation Group

and TradeStation Securities are now software/brokerage companies, and equities and futures trades can be executed through direct access with TradeStation.

TradeStation 2000i

PowerEditor Once this program is up and running, go under the *File* menu and select *New*. A dialog box titled *New* will open. If the *General* tab is not selected, go ahead and select it. Your screen should look like Figure 1.1.

Once your screen looks like Figure 1.1, select the icon with the title *Signal* and click *OK*. We will ignore the other tabs in this dialog box at the moment. Another dialog box titled *Create a New Signal* will open and ask for a name and notes about the signal that you are creating. In the Name field go ahead and type "MySignal-1" and in the Notes field type "Donchian Break Out" and then click *OK*. A window titled *MySignal-1* should open. This is your clean sheet of paper on which to type your wonderful ideas. Before we do some coding, let's briefly take a look at some of the menus. Table 1.1 details the selections that are available under the *File* menu. Table 1.2 details the selections under the *Edit* menu. Table 1.3 details the selections under the *View* menu. Table 1.4 details the selections under the *Tools* menu.

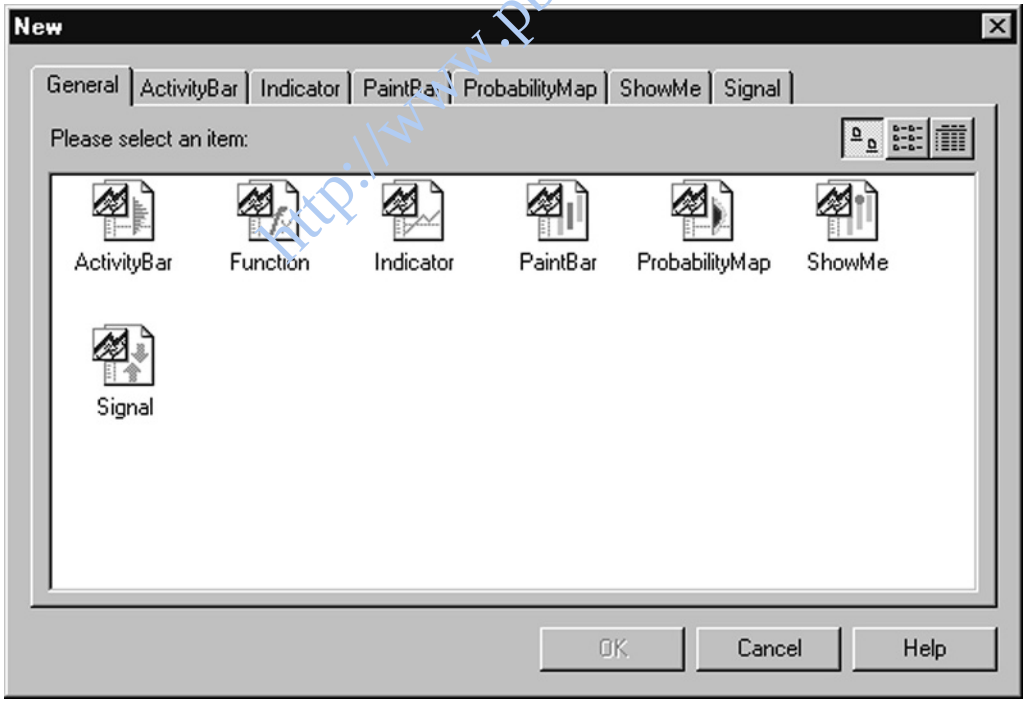

FIGURE 1.1 New Dialog—TradeStation 2000i

TABLE 1.1 File Menu

Menu Item	Action
New	Creates a new blank analysis technique.
Open	Opens an existing analysis technique.
Close	Pretty straightforward.
Save	Saves the current analysis technique.
Save As	Allows renaming of the current analysis technique.
Save As Template	Saves the current analysis technique as a template for future use.
Save All	Saves all currently open analysis techniques.
Import and Export	Imports existing analysis techniques from other sources or exports analysis techniques to other sources.
Protect	Password protects the analysis technique.
Verify	Verifies the analysis technique. Checks for syntax errors in code.
Verify All	Verifies all functions and analysis techniques.
Properties	Shows the Name and Notes of the analysis technique. Allows the user to change these.
Page Setup	Allows changing of the page setup.
Print	Prints the code.
Print Preview	Shows what will be printed.
Exit	Exits out of PowerEditor.

TABLE 1.2 Edit Menu

Menu Item	Action
Undo	Undoes the last action.
Redo	Redoes the last action.
Cut	Cuts the selected text.
Copy	Copies the selected text.
Paste	Pastes the selected text.
Clear	Clears the selected text.
Select All	Selects all text in the analysis technique.
Find	Finds a specific string of text.
Find Next	Finds the next occurrence of the specified string of text.
Find in Files	Powerful multifile search tool.
Replace	Replaces a string of text with another string of text.

TABLE 1.3 View Menu

Menu Item	Action
Toolbars	Customizes the tool bars.
Status Bar	Hides/shows the Status bar.
Output Bar	Hides/shows the Output bar. This little window will become very important when we start debugging.
Bookmarks	Adds a marker to a line of code so that you can quickly move between marked lines.
Font	Sets the PowerEditor's font.
Options	Displays options for the PowerEditor.

TABLE 1.4 Tool Menus

Menu Item	Action
EasyLanguage Dictionary	Inserts EasyLanguage components into an analysis technique.
Errors Window Options	Change the look of the Errors window.
Find In Files Window Options	Change the look of the Find In Files window.
Debug Window	Change the attributes of the Debug window.

A Simple Program Now that we are familiar with the menus and their functions, let's go ahead and code a simple program. We don't need to know everything about the selections in the menus to start coding. Jump in headfirst and type the following text exactly as you see it here:

```
Inputs:  longLength(40), shortLength(40);
Buy tomorrow  at  Highest(High,longLength) stop;
Sell tomorrow  at  Lowest(Low,shortLength) stop;
```

(Note: For those of you who may be moving to TradeStation 9.0, you must type "Sell Short" to initiate a short position.)

After you have typed this, go under the *File* menu and select *Verify* or hit the F3 key. Many of the commands in the menus have keyboard equivalents or shortcuts. You can determine the shortcuts by selecting the menu and then the menu item. (The keyboard shortcut is listed to the far right of the menu item.) If you look at the *Verify* menu item on the *File* menu, you will see F3. You should get a small dialog box that first says "Verifying" and then "Excellent." If you get an error, simply check your code for any typos and verify again. Congratulations, you have just written an analysis technique in the form of a signal! Now let's break down each line of code so that we can fully understand what is going on.


```
Inputs:  longLength(40), shortLength(40);
```

By typing this line you have created two constant variables of the numeric data type. These two variables, **longLength** and **shortLength**, have been initiated with the value of 40. These variables cannot be changed anywhere in the body of the analysis technique. They can be changed only from the user interface of this signal or in the program heading. This will be discussed later in this chapter.

```
Buy tomorrow  at  Highest(High,longLength) stop;
```

This line instructs the computer to place a buy stop tomorrow at the highest high of the last 40 days. **Highest** is a function call. Functions are subprograms that are designed for a specific purpose and return a specific value. To communicate with a function, you must give it the information it needs. An automatic teller machine is like a function;

you must give it your PIN before it will give you any money. In the case of the Highest function, it needs two bits of information: what to look at and how far to look back. We are instructing this function to look at the highs of the last 40 bars and return the highest of those highs. For now, just accept that **High** means high prices and **Low** means low prices. (This is yet another subject that will be touched on later.) When an order is instructed through EasyLanguage, you must tell the computer the type of order. In this case, we are using a stop order. Orders that are accepted by EasyLanguage are:

- *Stop.* Requires a price and is placed above the market to buy and below the market to sell.
- *Limit.* Requires a price and is placed below the market to buy and above the market to sell.
- *Market.* Buys/sells at the current market price.

So, if the market trades at a level that is equal to or greater than the highest high of the past 40 days, the signal would enter long at the stop price.

```
Sell tomorrow  at  Lowest(Low,shortLength) stop;
```

The instructions for entering a short position are simply the opposite for entering a long position.

TradeStation StrategyBuilder Now let's create a trading strategy with our simple Donchian Break Out signal. The StrategyBuilder is a program that asks all of the pertinent information concerning a trading strategy. It helps to organize all of the different trading ideas and makes sure all of the parameters are set before the signal is evaluated. Under the *Go* menu, select *TradeStation StrategyBuilder*. A dialog box titled *TradeStation StrategyBuilder* should open and look similar to Figure 1.2.

Click on the *New* button. Another dialog box opens and asks for a name and notes for this strategy. In the Name field type "MyStrategy-1," and in the Notes field type "A simple Donchian Break Out." After typing the information into these fields, click the *Next* button. The next dialog box asks for the signal to be used in this strategy. Click on the *Add* button. The next dialog box should look similar to Figure 1.3.

Scroll up/down and find *MySignal-1*. You will notice that the boxes under the Long Entry and Short Entry column headings are checked, but the boxes underneath the Long Exit and Short Exit are not. This tells us that the system only enters the market; it is never flat—the system is either long or short. Long positions are liquidated when a short position is initiated, and vice versa. Our simple Donchian Break Out is a pure stop-and-reverse system. Select *MySignal-1* and click on the *OK* button. Click the *Next* button, and a dialog box like the one in Figure 1.4 will open.

This dialog is informing us that there are two input variables for our signal. You can change the inputs now, but let's wait until later. If you did want to change the input values, you would simply select the Name of the input and edit the Value. Right now,

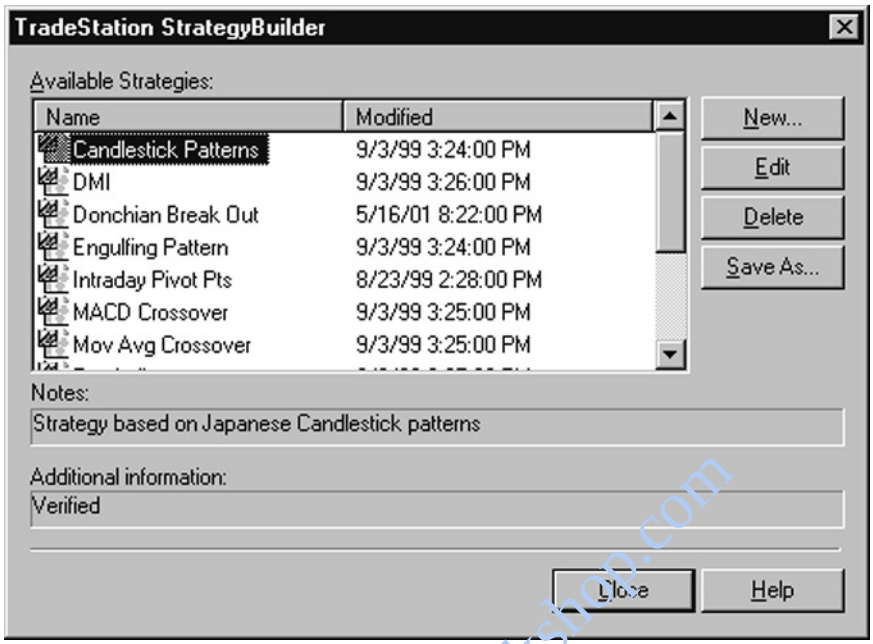

FIGURE 1.2 TradeStation 2000i—StrategyBuilder

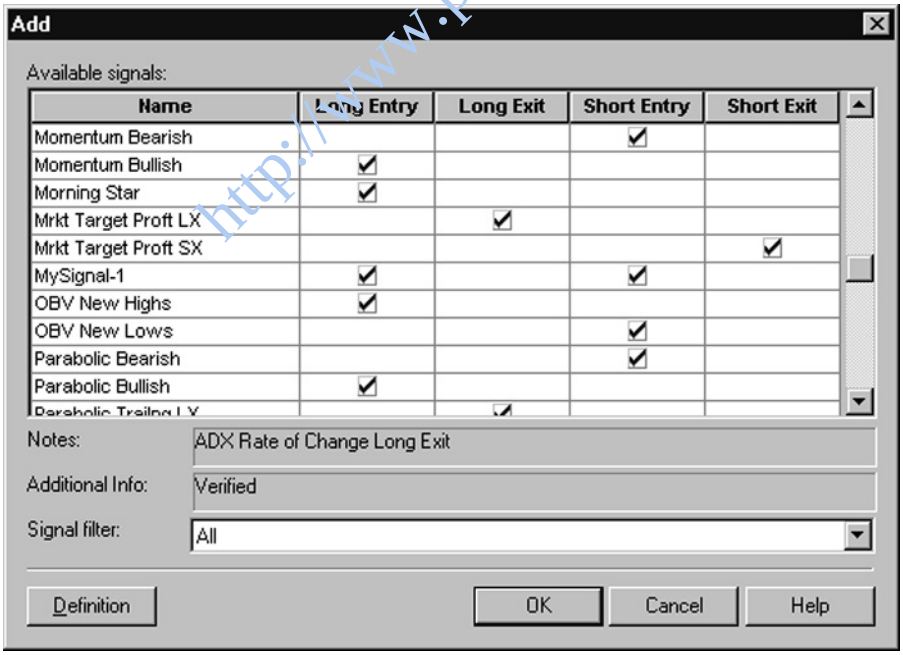

FIGURE 1.3 TradeStation StrategyBuilder—Add

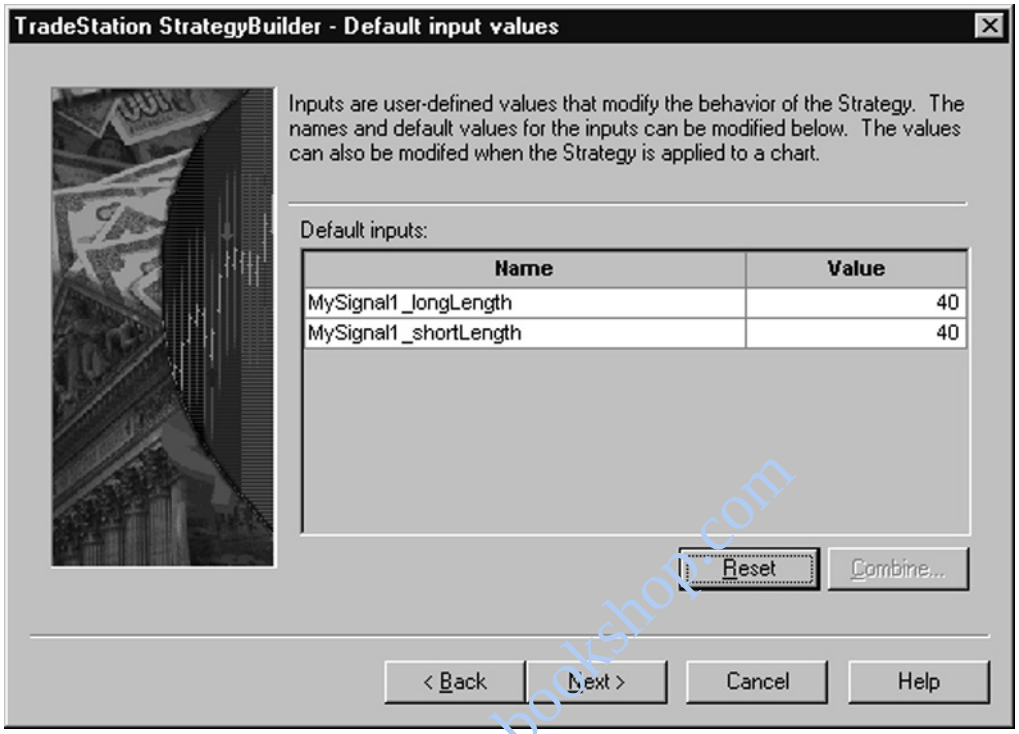

FIGURE 1.4 TradeStation StrategyBuilder—Input Values

simply click the *Next* button. The *Pyramiding* dialog box opens and asks if you would like to add positions in the same direction. Adding positions in the same direction occurs when our entry logic issues another buy signal and we are already long. We know that we will buy at the highest high of the past 40 days. This dialog is asking if we would like to continue adding positions at each subsequent 40-day high. If the market is trending, a new 40-day high could be made several times in succession. In this book, we will almost always take only one position per trade signal.

Click on the *Next* button. The *Position Information* dialog now opens and asks for the maximum number of open entries per position and the maximum number of contracts/shares per position. Maximum number of open entries per position limits the number of positions you can add as a result of pyramiding. Maximum number of contracts/shares per position limits the number of total contracts/shares that can be put on per position. The dialog box also asks if you would like it to send a notification to the Tracking Center when our strategy generates a new order. For now, let's accept the default values for the first two fields and make sure the box asking to send a notification to the Tracking Center is checked, and then click *Next*.

The *Data Referencing* dialog opens and asks for the maximum number of bars the study will reference. Our strategy needs only 40 days of data to generate a signal.

Remember, we are looking back 40 days to find the highest high and lowest low. Always keep in mind how much data the strategy that you are working on requires and make sure that you tell the computer, via this dialog box, that number. Make sure there is 40 or more in the field and click *Finish*. Congratulations again! You have just created your first strategy. Seems like a lot of work, doesn't it? TradeStation is just making sure that all the parameters are correct before testing a signal. Most of the time, these parameters don't change and you can simply click *Next*, *Next*, *Next*, *Next*, and finally, *Finish* without paying much attention. Okay, now let's apply our strategy. This book assumes the reader knows how to create daily and intraday charts in TradeStation. Create a daily bar chart of a continuous Japanese Yen contract going back 500 days. When this chart has been plotted, go under the *Insert* menu and select *Strategy*. A dialog box titled *Insert Analysis Technique* should open. Click on the *Strategy* tab and select *MyStrategy-1* from the list of available strategies and click on *OK*. Another dialog box titled *Format Strategy: MyStrategy-1* appears. Click on the *Inputs* tab, and you will see the two inputs that we have coded in our Donchian Break Out signal. By using the following line in our code, we have given the user of the strategy, be it ourselves or someone else, the ability to change the **longLength** and **shortLength** inputs of MySignal-1.

```
Inputs:  longLength(40), shortLength(40);
```


You do not need to change the code to change the system from a 40- to a 50-day Donchian Break Out strategy. These inputs can be edited at any time—before the analysis technique is inserted or after. You will notice that the values of these inputs default to those values that we had initiated when we programmed the signal in the PowerEditor. If you change these inputs from this dialog, they do not permanently change; they will only change during this session. If you want to change the inputs permanently, then change the value and click on the *Set Default* button. If you do change these inputs, always make sure you change the maximum number of bars the analysis technique will reference parameter. You can do this by selecting the *Properties* tab in our dialog box and changing the parameter. Let's take a look at the *Properties* dialog. Click on the *Properties* tab. Your dialog window should change and look like the one in Figure 1.5.

This dialog box allows the user to observe and change the current properties for MyStrategy-1. We initialized these properties when we first created the Strategy with the StrategyBuilder. You should be familiar with all of the property parameters except for the back-testing resolution. TradeStation enables you to specify the resolution or data compression to use for back-testing your trading strategy. When you create a chart using data that has already been collected, TradeStation must make certain assumptions about price movement. If you are back-testing on daily bars, TradeStation does not know when the high or the low of the day was made. In some strategies, this information may be important. TradeStation calculates the chronological order of the high and low by using a formula. This formula is not always accurate and may lead to inaccuracies. (This concept will be discussed in further depth in Chapter 6, but for right now, let's ignore this option.) Click on the *Costs* tab and you will be presented with the Commission and

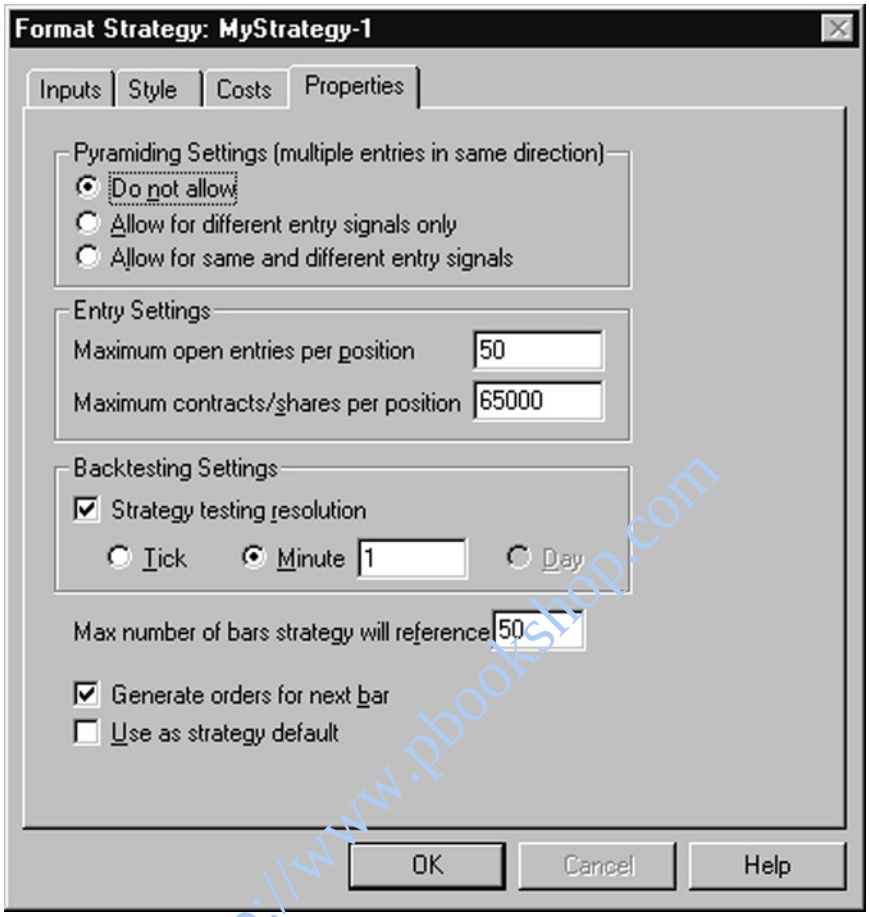

FIGURE 1.5 Format Strategy—TradeStation 2000i

Slippage $ values deducted for each trade. We all know what a commission is. Slippage is the dollar value expected when the actual fill price minus the calculated fill price is calculated on each trade. Slippage is either positive or negative; it is positive when you get in at a better price and negative when you get in at a worse price. Most of the time, slippage is negative. These costs can be charged on a per-contract/share basis or on a per-transaction basis. If you trade 300 shares of AOL, you can have TradeStation charge a commission/slippage on each share or for the entire trade. In addition, you will see the number of contracts/shares your strategy will assume with each new trade signal. If you select Fixed Unit, then this will be the fixed number of contracts/shares that will be traded throughout a historic back-test. If you choose Dollars per Transaction and type in a value in the associated text box, the number of shares or contracts will be calculated by dividing the input amount by the price of the stock or by the margin of the futures contract. Go ahead and accept all of the default values by clicking the *OK* button.

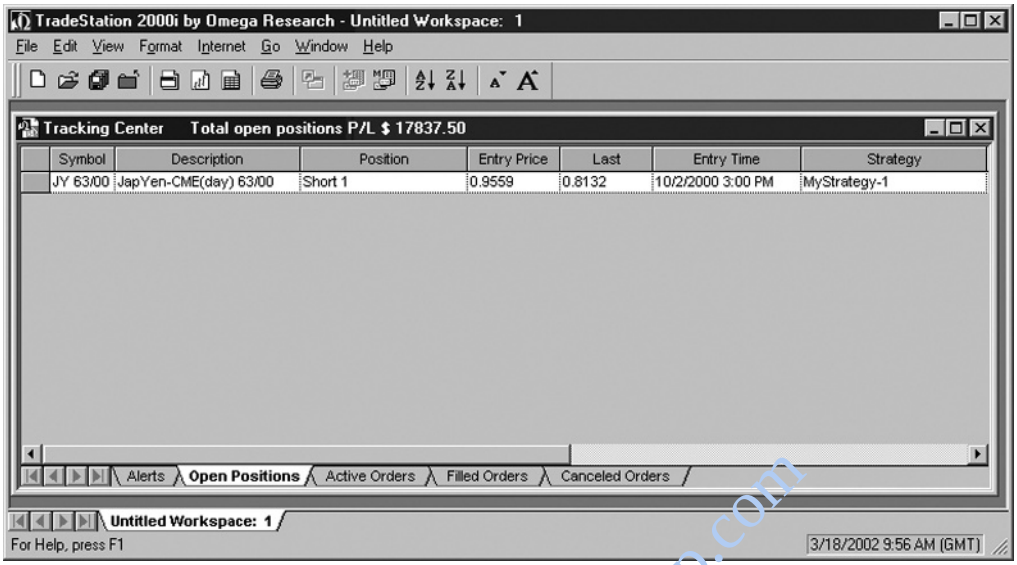

FIGURE 1.6 Tracking Center—TradeStation 2000i

Since MyStrategy-1 is a pure stop-and-reverse system, you will probably get a *New Open Position* dialog box that states that the market position has changed for JY. Let's close this dialog box by clicking on the *Close* button. When this dialog window disappears, there should be one underneath it titled *New Active Order*. This dialog box lets you know that an order needs to be placed today. It will either say "Buy 1 at a certain price stop" or "Sell 1 at a certain price stop." We need to learn more about the Tracking Center, so click on the *Go to Tracking Center* button. You may get another dialog box that states "No Open Tracking Center windows were found. Would you like to create one?" Go ahead and click *Yes*. Your Tracking Center should look similar to one in Figure 1.6.

Click on the *Open Positions* tab and you will see the symbol we are currently testing, current position, entry price, entry time, open profit, and various other statistics. Click on the *Active Orders* tab and you should see an LE (Long Entry) order to Buy 1 at a certain price stop and an SX (Short Exit) order to Sell 1 at a certain price stop, if you are currently short.

If you are currently long, you would see an SE (Short Entry) order to Sell 1 at a certain price stop and an LX (Long Exit) order to Buy 1 at a certain price stop. In real-life order placement, you would simply place a single order to Buy/Sell 2 at whatever price was issued on a stop. Reduce this window and the chart of the Japanese Yen with buy and sell signals should now be the only window on the screen. If you like, you can go under the *View* menu and select *Strategy Performance Report* and look at how well the system performed over the test period. We will go much further in detail concerning the reports that TradeStation creates for the analysis of trading strategies in Chapter 5.

TradeStation 9.0

TradeStation Development Environment Since the TradeStation Development Environment can run independent of TradeStation 9.0, you can launch it by finding it under the *Start* menu or double clicking its icon. This is helpful sometimes when you simply want to modify some code or start a new strategy or analysis technique and don't need to see the results of your work. However, if you do want to see the results of your coding instantaneously, then you should have both running. Let's do the latter and launch TradeStation 9.0. For this exercise let's go ahead and log on by typing in your User Name and Password. Your screen should look somewhat similar to Figure 1.7.

The TradeStation Development Environment, TDE for short, can also be accessed through the useful Shortcut bar. If your screen doesn't show the Shortcut bar (labeled *Tools*), go under the *View* menu and make sure the Shortcut bar is checked. Once the bar is open, click on the *EasyLanguage* button. The TDE should launch momentarily. Once the TDE is up and running, select *New* from the *File* menu. A new submenu similar to the one in Figure 1.8 should now be on your screen. All the different types of analysis techniques that can be created by EasyLanguage are now at your disposal.

Slide down the menu and select *Strategy*. A *New Strategy* dialog box like the one in Figure 1.9 will open and ask for a name and notes about the strategy that you are creating.

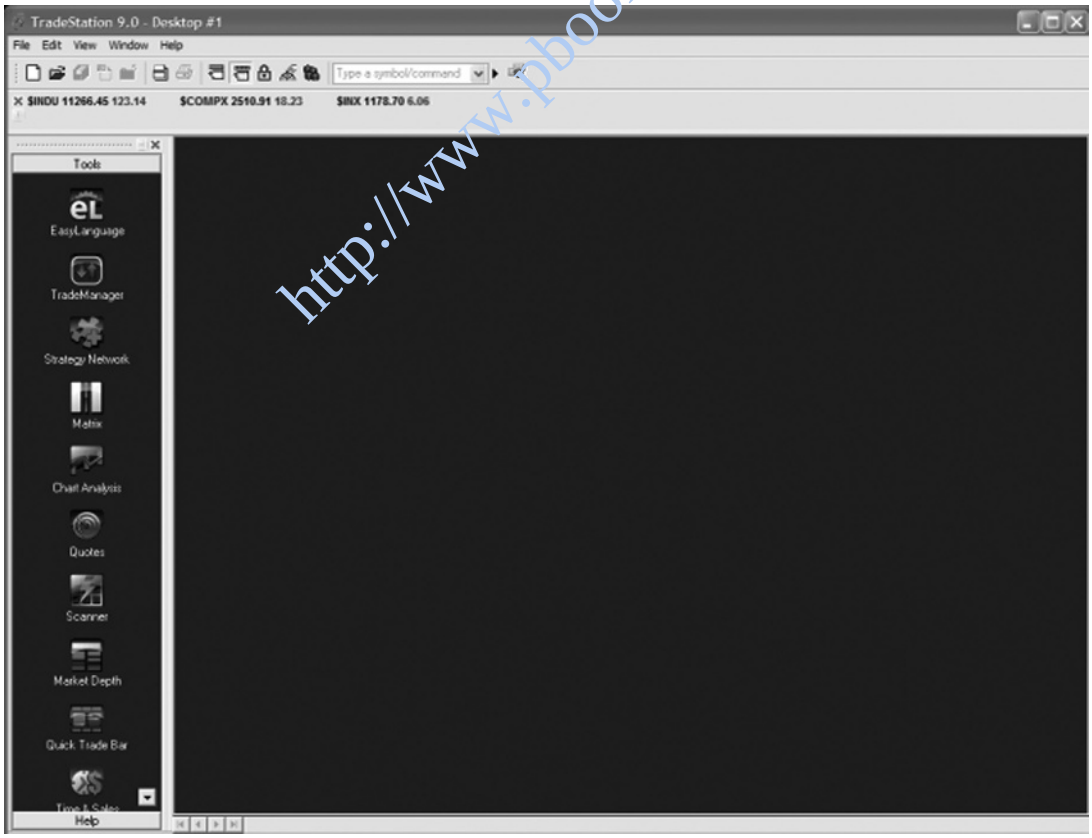

FIGURE 1.7 Shortcut Bar—TradeStation 9.0

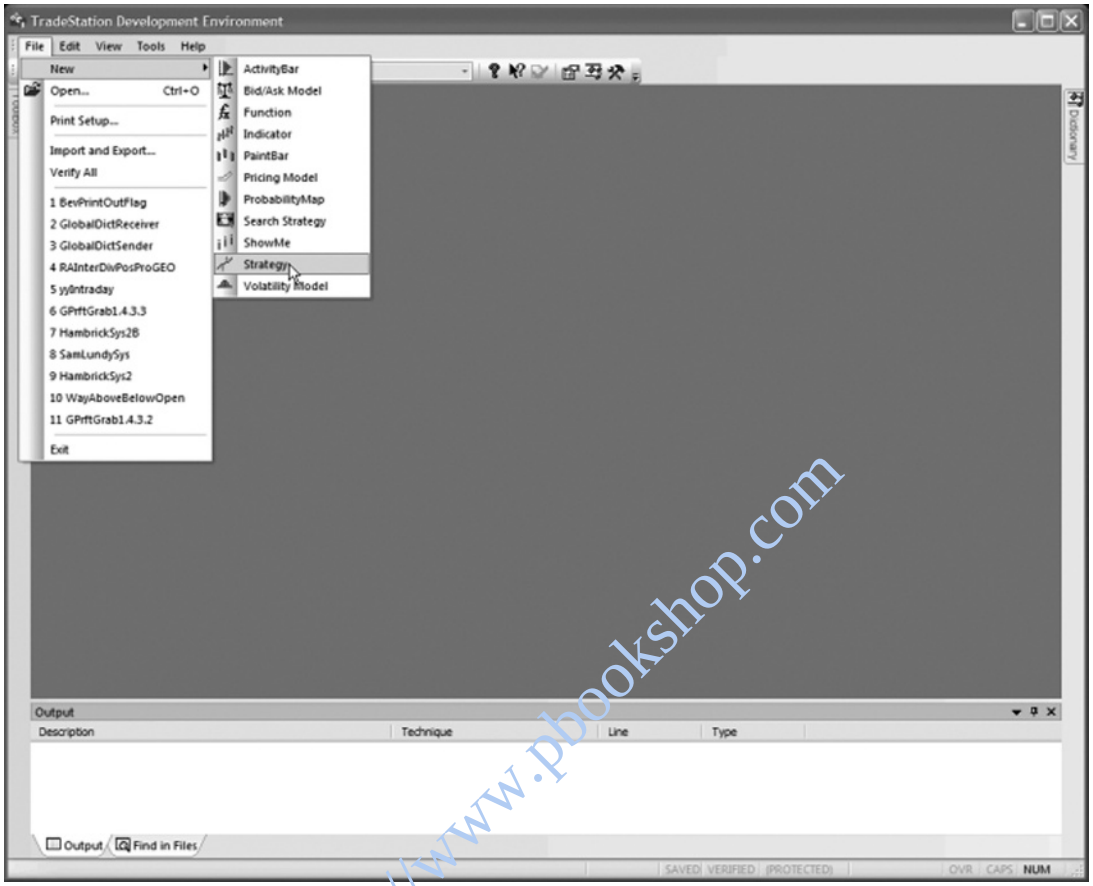

FIGURE 1.8 New File Submenu—TradeStation 9.0

In the Name field go ahead and type "MyStrategy-1" and in the Notes field type "Donchian Break Out." In the Select Template field, leave it at "none" and then click *OK*. A blank window will open and a tab associated with this strategy will also appear. This is your clean sheet of paper on which to type your wonderful ideas. It is also known as the EasyLanguage Editor and is part of the TDE. Also, the title of the TDE window will reflect the name of the analysis technique or function you are currently working on. Make sure your screen looks like Figure 1.10.

You have successfully started a new trading strategy. Before we do some programming, let's briefly take a look at some of the menus.

- Table 1.5 details the selections that are available under the *File* menu.
- Table 1.6 details the selections under the *Edit* menu.
- Table 1.7 details the selections under the *View* menu.
- Table 1.8 details the selections under the *Tools* menu.

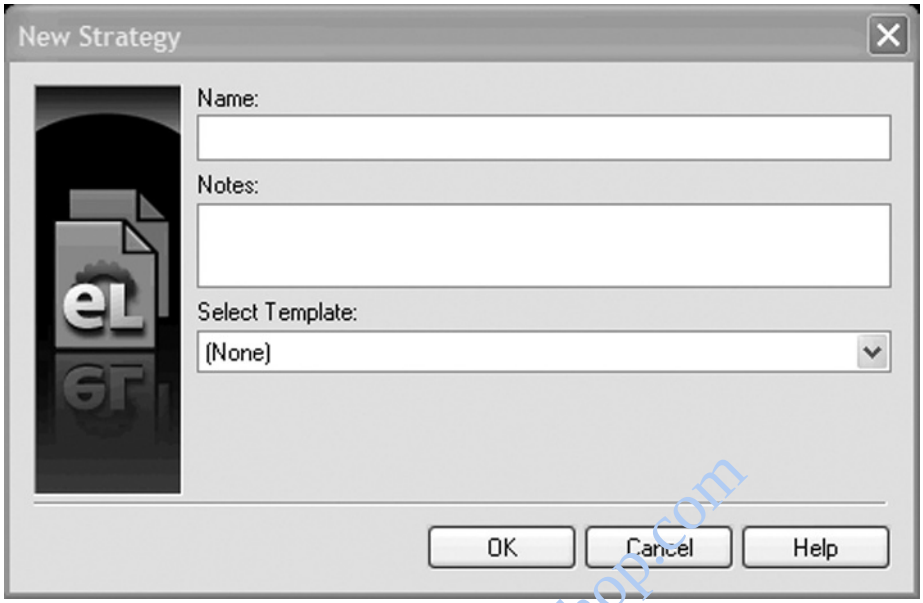

FIGURE 1.9 New Dialog—TradeStation 9.0

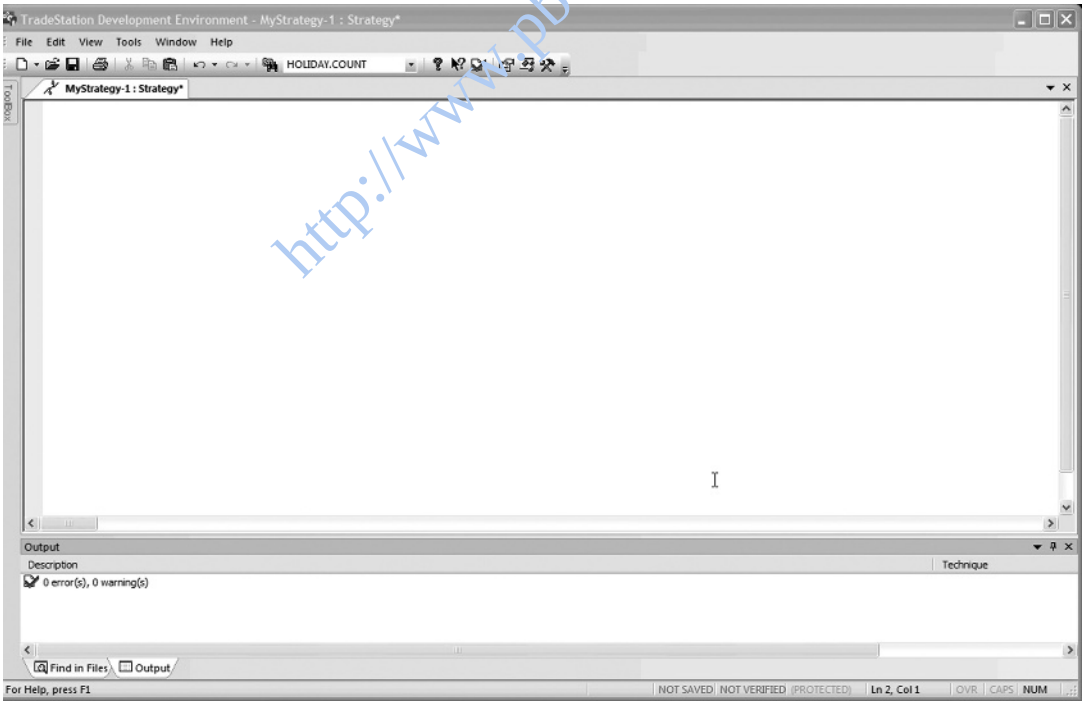

FIGURE 1.10 Our MyStrategy-1 Blank Sheet

TABLE 1.5 File Menu

Menu Item	Action
New	Creates many different types of analysis techniques.
Open	Opens an existing analysis technique.
Close	Closes window.
Save	Saves the current analysis technique.
Save As	Allows renaming the current analysis technique.
Save As Template	Saves the current analysis technique as a template for future use.
Save All	Saves all the currently opened analysis techniques.
Import/Export	Imports existing analysis techniques from other sources or exports analysis techniques to other sources. (Note: You can import from previous versions but you cannot export to previous versions.)
Protect	Allows you to password protect your analysis technique.
Verify	Verifies the analysis technique. Checks for syntax errors in the code.
Verify All	Verifies every function and analysis technique in your library. This can take a very long time, so make sure this is what you want to do.
Properties	Gives certain information pertaining to the current analysis technique. You can view/edit the Short Name, Notes, Max. number of look-back bars, Position Sizing, and Pyramiding.
Print	Prints the code of the current analysis technique.
Exit	Closes the EasyLanguage Editor.

TABLE 1.6 Edit Menu

Menu Item	Action
Undo	Undoes the last action.
Redo	Redoes the last action.
Cut	Cuts the selected text.
Copy	Copies the selected text.
Paste	Pastes the selected text.
Clear	Clears the selected text.
Select All	Selects all text in the analysis technique.
Find	Finds a specific string of text.
Find Next	Finds the next occurrence of the specified string of text.
Replace	Replaces a string of text with another string of text.
Find in Files	Powerful multifile search tool.
Outlining	Outlining is a feature in the EasyLanguage Code Editor that allows you to manage whether blocks of code collapse or expand.
Advanced	Converts to uppercase/lowercase. Also allows quick commenting of selected code.
Bookmarks	Bookmarking is an EasyLanguage Code Editor feature that allows you add a marker to a line of code so that you can quickly move between marked lines in larger EasyLanguage documents.

TABLE 1.7 View Menu

Menu Item	Action
Tool Bars	Hides/shows different tool bars.
Status Bar	Hides/shows the Status Bar.
Dictionary	Hides/shows the EasyLanguage Dictionary. Very handy when looking for the correct reserved word. When found, the word can be dragged directly into your source code.
Designer Generated Code	Advanced feature, which will not be discussed in this book.
Launch TradeStation Platform	Launches TradeStation Platform if not already opened.

A Simple Program Now that we are familiar with some of the menus and their functions, let's code a simple program. We don't need to know everything about the selections in the menus to start coding. Type the following text exactly as you see it here:

```
Inputs:  longLength(40), shortLength(40);
Buy tomorrow  at  Highest(High,longLength) stop;
Sell Short tomorrow  at  Lowest(Low,shortLength) stop;
```

After you have typed this in, go under the *File* menu and select *Verify* or press F3. You should get a small dialog box that says, "If this analysis technique/strategy is currently applied, it will automatically be recalculated." Click the *OK* button. If you typed everything properly, you will see 0 error(s), 0 warning(s) in the output pane of the EasyLanguage Editor. If you get an error, simply check your code for any typos and verify again. Congratulations, you have just written an analysis technique in the form of a strategy! Now let's break down each line of code so that we can fully understand what is going on.


```
Inputs:  longLength(40), shortLength(40);
```

By typing this line you have created two constant variables of the numeric data type. These two variables, **longLength** and **shortLength**, have been initiated with the value of 40. These variables cannot be changed anywhere in the body of the analysis technique. They can be changed from the user interface of this signal or in the program header. These will be discussed later in this chapter.

```
Buy tomorrow  at  Highest(High,longLength) stop;
```

TABLE 1.8 Tool Menu

Menu Item	Action
Customize	You can change the look/actions of the menus, toolbars, commands, and keyboard with this menu item.
Options	Sets the EasyLanguage Editor's font, background color, syntax coloring, and enables/disables the new Autocomplete.

This line instructs the computer to place a buy stop tomorrow at the highest high of the last 40 days. **Highest** is a function call. Functions are subprograms that are designed for a specific purpose and return a specific value. To communicate with a function, you must give it the information it needs. An automatic teller machine is like a function; you must give it your PIN before it will give you any money. In the case of the Highest function, it needs two bits of information: what to look at and how far to look back. We are instructing this function to look at the highs of the last 40 bars and return the highest of those highs. When an order is instructed through EasyLanguage, you must tell the computer the type of order. In this case, we are using a stop order. Orders that are accepted by EasyLanguage are:

- *Stop.* Requires a price and is placed above the market to buy and below the market to sell.
- *Limit.* Requires a price and is placed below the market to buy and above the market to sell.
- *Market.* Buys/sells at the current market price.

So, if the market trades at a level that is equal to or greater than the highest high of the past 40 days, the signal would enter long at the stop price.

```
Sell Short tomorrow at Lowest(Low,shortLength) stop;
```

The instructions for entering a short position are simply the opposite for entering a long position.

For those of you who are moving from TradeStation 2000i to a new version, you will be pleasantly surprised by the elimination of the StrategyBuilder. Version 6.0 and greater create strategies on the fly by accepting default values for the strategy's properties. For those of you who are not familiar with the StrategyBuilder, it was an additional component that required the user to click through several dialog boxes and change property values before one could create a strategy. Most of the time the default values were sufficient and this was an act of futility. Version 9.0 allows the strategy properties to be changed when needed.

This book assumes the reader knows how to create daily and intraday charts in TradeStation. If you are not sure how to do this, we refer you to your TradeStation manuals. Let's start by creating a daily bar chart of the Japanese Yen going back 500 or more days in the TradeStation Platform. When this chart has been plotted, go under the *Insert* menu and select *Strategy*. A dialog box titled *Insert Analysis Techniques and Strategies* should open and look similar to the one in Figure 1.11.

This dialog box is very informative: It lists the different strategies and also informs the user if the strategy has been verified and if the strategy has a long entry, short entry, long exit, and short exit. Scroll through the list until you find MyStrategy-1. You will notice that the boxes under the Long Entry and Short Entry column headings are checked, but the boxes underneath the Long Exit and Short Exit are not. This tells us

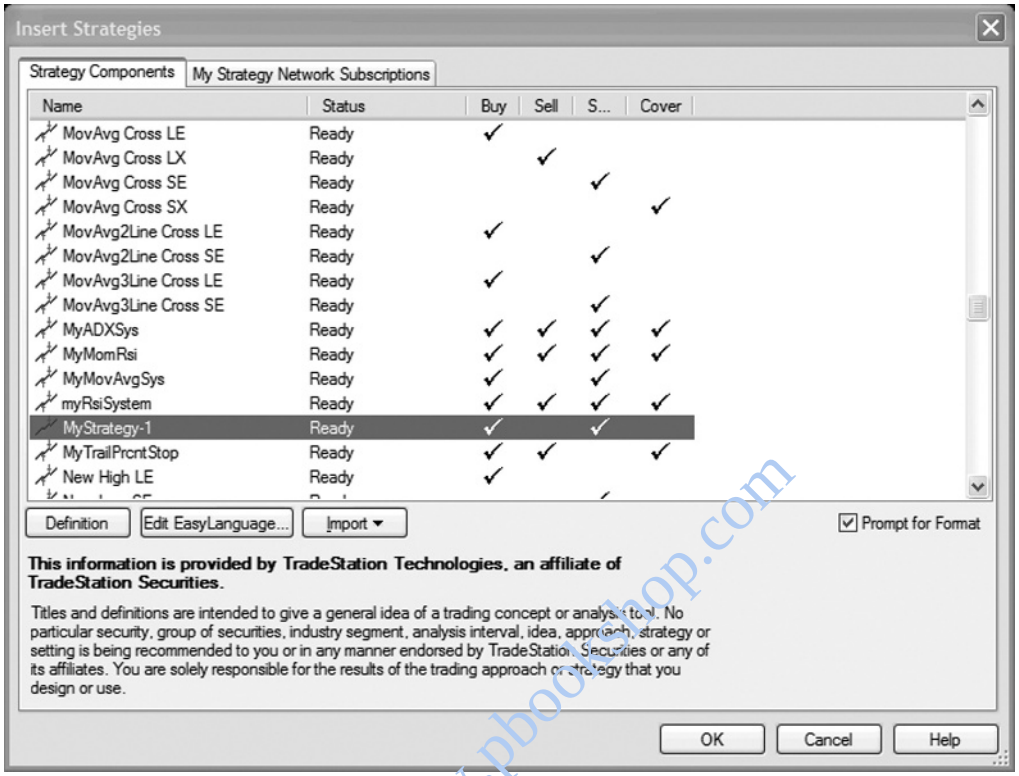

FIGURE 1.11 Insert Analysis Techniques and Strategies

that the system only enters the market; it is never flat—the system is either long or short. Long positions are liquidated when a short position is initiated, and vice versa. Our simple Donchian Break Out is a pure stop-and-reverse system. You will also notice a small check box titled *Prompt for Format*. Make sure this box is checked, and then select *MyStrategy-1* from the list of available strategies and click on *OK*. Another dialog box titled *Format Strategy* appears and should look similar to Figure 1.12.

Click on the *Inputs* button. You will see the two inputs that we have coded in our Donchian Break Out strategy. By using the following line of code, we have given the user of the strategy, be it ourselves or someone else, the ability to change the **longLength** and **shortLength** inputs of MyStrategy-1.

```
Inputs:  longLength(40), shortLength(40);
```

You do not need to change the code to change the system from a 40- to a 50-day Donchian Break Out. These inputs can be edited at any time—before the analysis technique is inserted or after. You will notice that the values of these inputs default to those values that we had initiated. If you change these inputs from this dialog, they do not

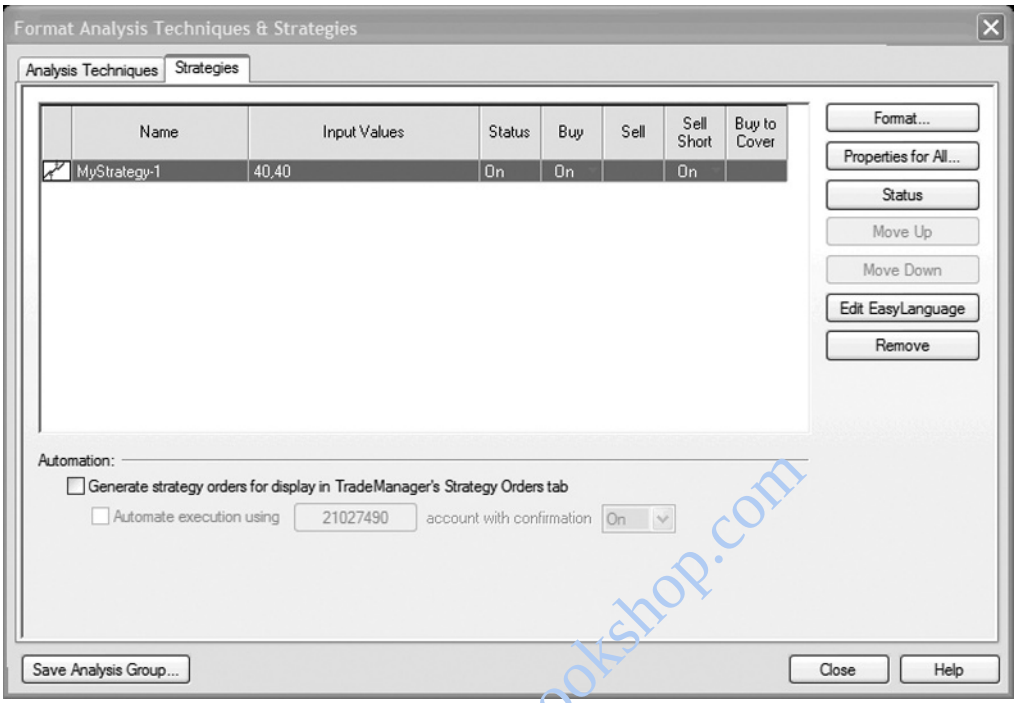

FIGURE 1.12 Format Strategy

permanently change; they will change only during this session. You can permanently change the default input values for MyStrategy-1 by changing the values and then clicking on the *Set Default* button. If you do change these inputs, always make sure you change the maximum number of bars the analysis technique will reference parameter. We will show you how to reference this parameter when we discuss the *Format* dialog box. You will notice that there are several other buttons in this dialog box; however, we will ignore these for now and accept our inputs as they are by clicking on the *OK* button. Click on the *Format* button. A dialog box should open titled *Format Strategy*. It should look like Figure 1.13.

You will notice several different parameters that TradeStation takes into account when testing a strategy. Let's examine each option and its purpose.

Commission per share/contract $. The dollar value that will be deducted from each trade as a commission charge. Let's set this to $0.

Slippage per share/contract $. The dollar value that you expect the strategy will be slipped on each trade. Slippage is the actual fill price minus the calculated fill price. Slippage is either positive or negative; it is positive when you get in at a better price and negative when you get in at a worse price. Most of the time, the slippage is negative. Let's set this to $0.

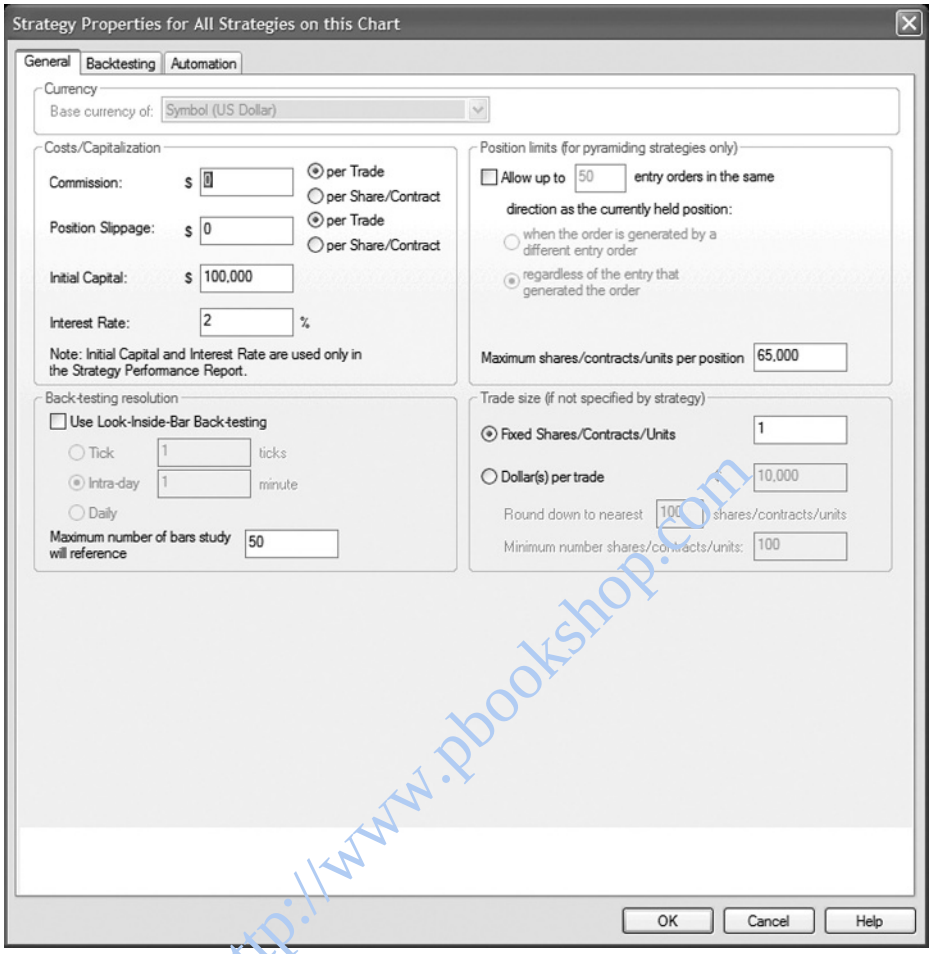

FIGURE 1.13 Format Strategy after Clicking the *Format* Button

Trade size (if not specified by strategy).

Fixed unit. The number of shares or contracts that are put on at the initiation of a trade. Change this value to 1, if it isn't already so.

Dollars per transaction. The fixed dollar value used to determine the number of shares/contracts. Trade size is calculated by dividing the price of the instrument into this value.

Go ahead and click *OK*, and you return to the *Format Strategy* dialog box. Before we close this dialog box and proceed with the application of the strategy to the chart, make sure the box that says *Generate strategy orders for display in Account Manager's Strategy Order Tab* is checked. Since MyStrategy-1 is a pure stop-and-reverse system, you will get a *Strategy New Order* or a *Strategy Active Order* dialog box that states that

you should buy or sell short at a certain price. If one of these dialog boxes does indeed open, just close it by clicking on the *Close* button.

We need to learn more about the TradeManager. Go to the *File* menu and select *New* and then select *Window*. A dialog window will open. Select the *Tools* tab. Click on the *TradeManager* icon and then click the *OK* button. We could have accomplished this by using the *Shortcut* bar and clicking on the *TradeManager* icon. We have simply fallen in love with the *Shortcut* bar. Once you click the icon, a window like the one in Figure 1.14 will open.

This window keeps track of all real orders and positions (if you have an actual trading account with TradeStation securities) and all simulated orders and positions. Since we are working with simulated trades that were generated by our strategy, click on the *Strategy Positions* tab. Your window will change and should look like the one in Figure 1.15.

This spreadsheet shows the symbol we are currently testing, current strategy position, entry price, entry time, and various other statistics. Click on the *Strategy Orders* tab, and your window will change to look like the one in Figure 1.16.

Since our strategy is in the market all of the time, you will have two rows (possibly more if you are currently tracking more than one system of information.

The top row will be the order to initiate a new position and the second row will be the order to cover the existing position. If the order is a stop order, the stop price will be located under the Stop header. If it is a limit order, the limit price will be under the Limit header. If today's market action causes a fill to occur, the fill price will be under the

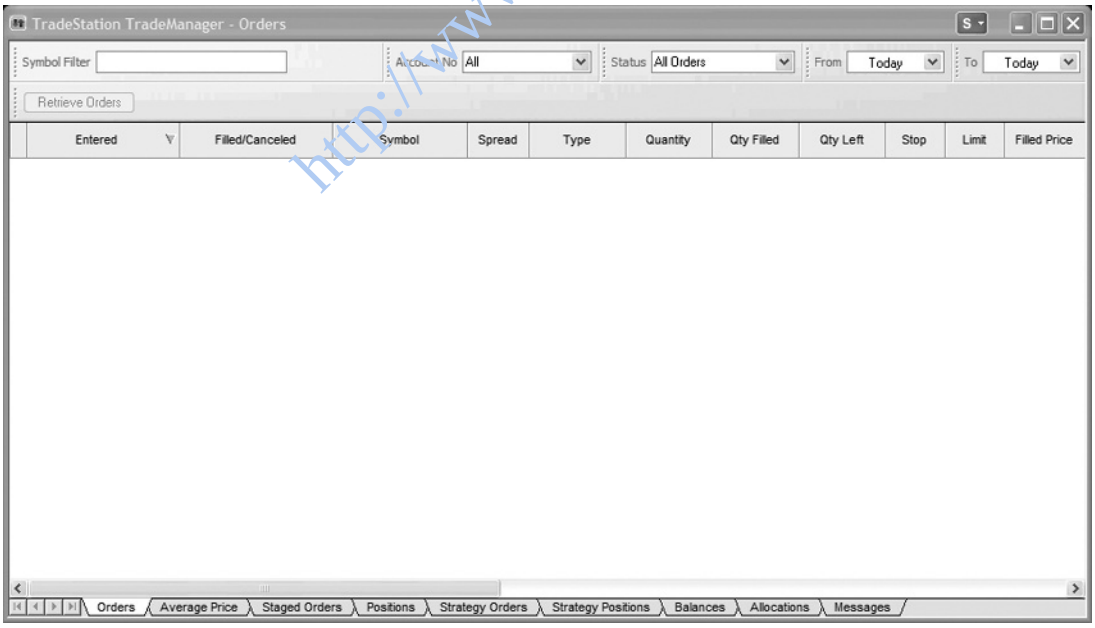

FIGURE 1.14 Account Manager and Strategy Tracking—Today's Orders

FIGURE 1.15 Account Manager and Strategy Tracking—Strategy Positions

Filled header. You will notice other tabs in the *TradeManager* window. These tabs are used only if you have an account set up at TradeStation's brokerage company. Basically, these other tabs give the same information as the Strategy tabs, but with real execution statistics. You can use TradeStation 9.0 without a trading account, but you will need to keep track of your real live positions and fills yourself. This in no way takes away from the back-testing and research capabilities of this product.

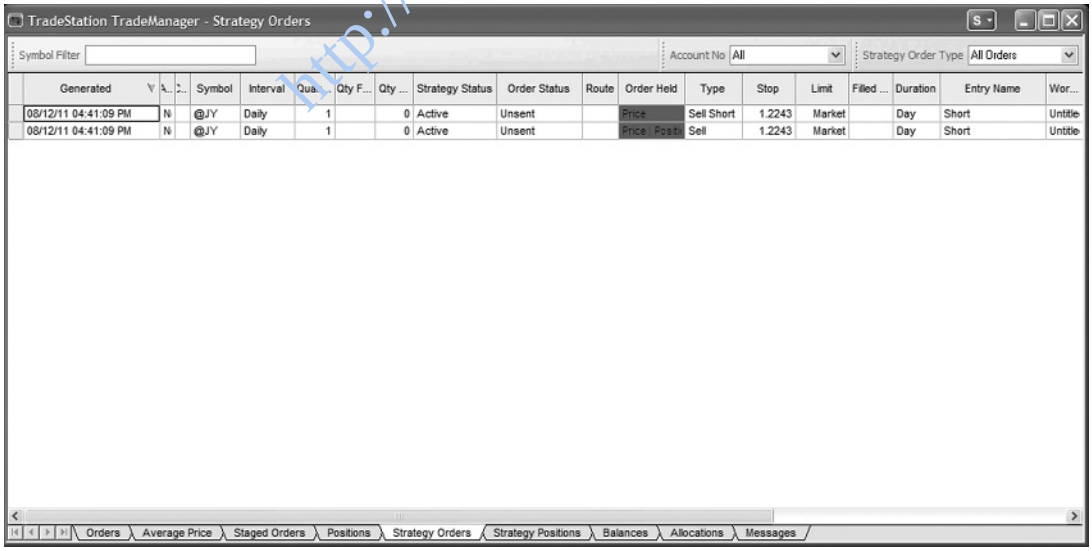

FIGURE 1.16 Account Manager and Strategy Tracking—Strategy Orders

Reduce this window, and the chart of the Japanese Yen with buy and sell signals will now be the only window on the screen. If you like, you can go under the *View* menu and select *Strategy Performance Report* and look at how well the system performed over the test period. We will go into much further detail concerning the reports that TradeStation creates for the analysis of Trading Strategies in Chapter 5.

TradeStation 9.0 EasyLanguage Editor With version 8.8, TradeStation finally developed a powerful IDE (Integrated Development Environment). It is now similar to other professional program authoring tools. It is so much better that we felt a brief tutorial on the EasyLanguage Editor should be included in this introductory chapter. If you are experienced with the new editor, then you might want to skip this section and go on to Chapter 2.

An impressive feature of this editor is the ability to quickly and easily move from the source code of one program (the word "program" will be interchangeable with "strategy and analysis technique" throughout the rest of this book) to another. When you open to edit more than one analysis technique, each program or file is opened underneath its own tab very similar to the way Microsoft Excel uses tabs for different worksheets. This is a nice feature because it makes copying existing code from one strategy, function, indicator, or paintbar to another very easy. You will find out that programming is like building with building blocks. After programming for a while, you will develop a library of ideas that you use and reuse again and again. So it's nice to be able to program a new idea with bits and pieces of old ideas. The multiple tab concept makes this much simpler than the way the old TradeStation cobbled the program source code windows along with any other window.

The uncoupling of the editor and the charting/trading platform has made all of this possible. Now you can launch the EL Editor independent of the TradeStation platform. If you simply want to make some modifications to existing programs or begin new ones that do not need to be immediately applied to a chart, then you can do so without the overhead of the testing/trading platform.

Outlining is a feature where blocks of source code can be grouped together to make readability much easier. This also helps make programs much more modular. We discuss the concept of modular programming in the next chapter. A portion of the strategy that we will work on later in the book is shown in Figure 1.15. You can easily see how the blocks of code are outlined. There's a small box with a "-" symbol and a vertical line connecting the related blocks of code together. Outlining also allows you to hide the blocks of code if you wish to do so, again adding to the readability of the code.

EasyLanguage has a huge library of reserved words and it is very difficult if not impossible to remember them all. The new editor has a really cool feature called Autocompleting. Let's say you want to code an idea that incorporates an average true range calculation and you can't remember the name of the function that does the calculation. You could, of course, stop what you are doing and go to *Help* and look up the function. However, this is time-consuming, so the new EL Editor monitors what you are typing and provides all the possible reserved words that might match what has thus been typed.

It does this by creating a drop-down menu with all of the reserved words and placing you in the general area of words that begin with what you have typed. In the case of our example, all you need to do is type what you might think is the name of the function, and the list appears. By typing "av" the list pops up and places you at the word "average." You can simply scroll down to find the function AvgTrueRange.

Let's play around with the EL Editor a bit before moving on to Chapter 2. If you haven't downloaded the companion code for this book, it would be advisable to do so now. The editor should still be open from our previous exercise. If not, then go ahead and launch it. From the *File* menu select *Open* and when the *Open Easy Language Document* dialog box appears, select *Strategy* in the *Select Analysis Type* drop-down menu. All of the strategies in your library are now presented to you. Scroll down and select *Thermostat* and click on the *Open* button. The source code for Thermostat should now be in your editor window and look similar to Figure 1.17.

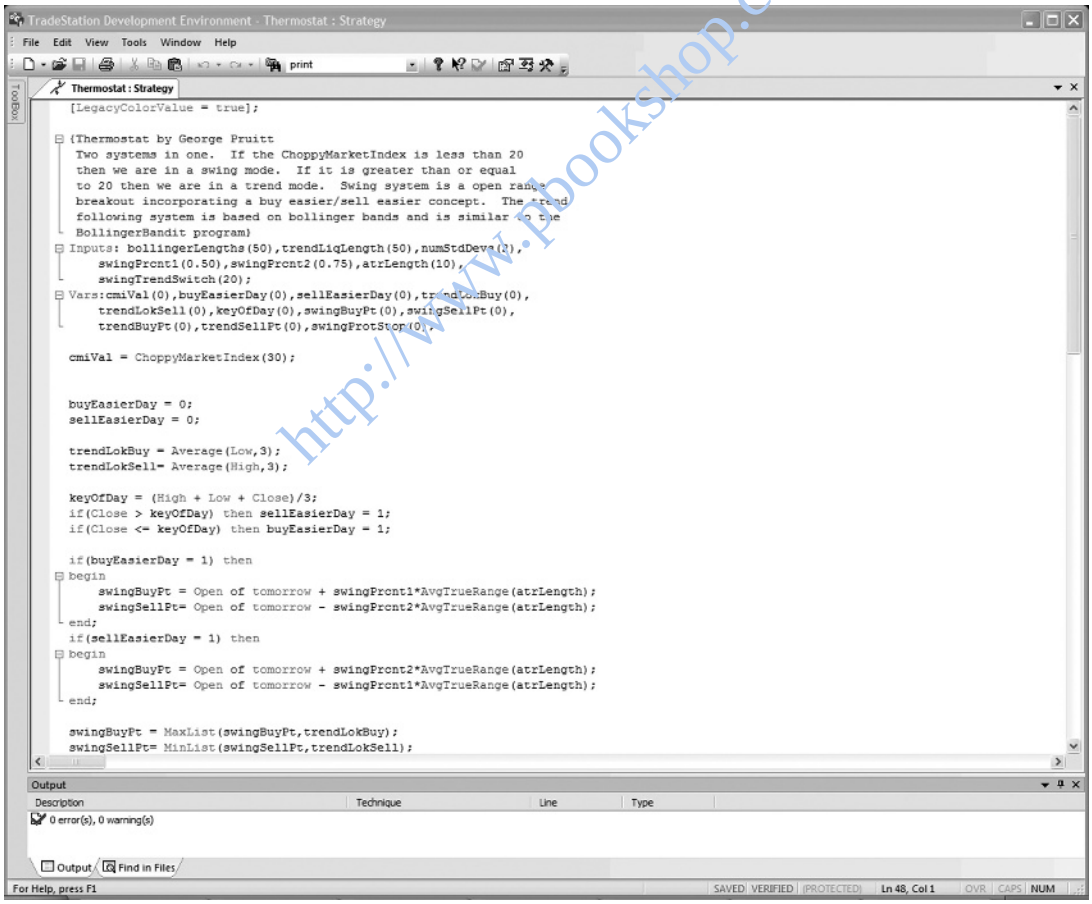

FIGURE 1.17 The Thermostat Strategy Opened in the EL Editor

```
if(buyEasierDay = 1) then
begin
     swingBuyPt = Open of tomorrow + swingPrcnt1*AvgTrueRange(atrLength);
     swingSellPt= Open of tomorrow - swingPrcnt2*AvgTrueRange(atrLength);
end;
```

FIGURE 1.18 An Example of How Code Is Outlined

Scroll down until you find this line of code:

```
if(buyEasierDay = 1) then
```

You should see a little box with a "-" at the start of the following line of code. You will also see a line running vertically to the left that signifies a group of related code. Figure 1.18 shows how this block of code is outlined. If you don't see the box or the vertical line, then make sure your outlining is turned on. You do this by going up under *Edit* in the menu bar and selecting *Enable Outlining* in the *Outlining* submenu.

Click on the "-" in the box and you will notice the code that was included in this particular outline is collapsed and the "-" in the box is replaced with a "+" symbol. It should look like Figure 1.19.

You can toggle back and forth by simply clicking in the small box with either the "-" or "+" symbol. This is a neat feature when you are dealing with larger programs and want to hide blocks of code that are not being currently modified. It helps with the readability of the code. The outline also helps you see blocks of related information. Don't worry about how the editor selects the outline at this time because we will discuss this in Chapters 2 and 3.

Refer back to the source code of the Thermostat strategy and take a look at the beginning of the code. You will see the following snippet:

```
{Thermostat by George Pruitt
 Two systems in one. If the ChoppyMarketIndex is less than 20,
 then we are in a swing mode. If it is greater than or equal
 to 20, then we are in a trend mode. Swing system is an open range
 breakout incorporating a buy easier/sell easier concept. The trend
 following system is based on bollinger bands and is similar to the
 BollingerBandit program}
```

This information briefly describes how the strategy works. This description is totally optional, but coming from a professional programmer, you should explain as much as possible in English what you're trying to accomplish. Again this helps to remind or explain to the reader of the code what it does. You will notice the "{" bracket at the beginning of the text and the "}" bracket at the end of text. The "{}" brackets are

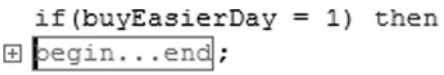

FIGURE 1.19 An Example of How Code Is Collapsed

telling the computer to ignore the text that is encapsulated. This is known as a comment block and we discuss how this works in Chapter 2. But for right now we want to show yet another cool feature of this editor. Find the following snippet of code in the Thermostat strategy:

```
buyEasierDay = 0;
sellEasierDay = 0;
trendLokBuy = Average(Low,3);
trendLokSell= Average(High,3);
```

Go ahead and highlight these four lines by holding down your left mouse button and dragging to capture just these lines. Once they are highlighted, go up under the *Edit* menu and select *Comment Selection* from the *Advanced* submenu. You will notice the text will be enclosed by curly brackets.

```
{buyEasierDay = 0;
 sellEasierDay = 0;
 trendLokBuy = Average(Low,3);
 trendLokSell= Average(High,3);}
```


This is known as "commenting out" a piece of code from the entire program. Again, don't worry about what this means; we just want you to get familiar with some of the neat features of the EasyLanguage Editor.

The last feature we want to highlight is the editor's dictionary capabilities. Go back to the EasyLanguage editor and scroll down until you see these two lines of code:

```
trendBuyPt = BollingerBand(Close,bollingerLengths,numStdDevs);
trendSellPt= BollingerBand(Close,bollingerLengths,-numStdDevs);
```

Click on the word "BollingerBand" with the right mouse button and select *Definition of BollingerBand* from the pop-up menu. The dictionary will quickly open and the exact definition of BollingerBand will be presented. If you select a word that is not recognized by the dictionary, it will tell you, "Requested topic not found."

Continue navigating around the editor and play with some of the features we have mentioned. There are a lot of tools and options we haven't discussed, so just ignore them. Some of these tools will be covered in later chapters. When you are done, simply close out the editor but do not save changes to the Thermostat strategy.

CONCLUSIONS

The objective of this chapter is to introduce the fundamentals necessary to become a productive EasyLanguage programmer. We discussed data types, expressions, and statements. We reviewed how to declare variables and inputs, call built-in functions, and verify (compile) our code. In addition, we can create a strategy from a simple signal and insert that strategy into a chart. In the next chapter, we will build on this foundation and create much more complex programs ("analysis techniques," in EasyLanguage vernacular).