Introduction to AmiBroker

Second Edition

Advanced Technical Analysis Software for Charting and Trading System Development

Howard B. Bandy

Blue Owl Press

ISBN-10:

ISBN-13:

LCCN:

Published by

Blue Owl Press, Inc.

3700 S. Westport Avenue, #1876

Sioux Falls, SD 57106

Published First edition 2008, Second edition 2012

Disclaimer

This book is an educational document. Nothing in this book is intended as, nor should it be construed to be, investment advice.

The views expressed herein are the personal views of Dr. Howard B. Bandy. Neither the author nor the publisher, Blue Owl Press, Inc., have any commercial interest in any of the products mentioned. All of the products described were purchased by the author at regular retail prices.

Investing and trading is risky and can result in loss of principal. Neither this book in its entirety, nor any portion thereof, nor any follow-on discussion or correspondence related to this book, is intended to be a recommendation to invest or trade mutual funds, exchange traded funds (ETFs), stocks, commodities, options, or any other financial instrument. Neither the author nor the publisher will accept any responsibility for losses which might result from applications of the ideas expressed in the book or from techniques or trading systems described in the book.

The programs used as examples have been tested and are believed to be correct. Even so, this book may contain typographical errors and other inaccuracies. Past performance, whether hypothetical, simulated, backtested, or actual, is no guarantee of future results. Results will depend on the specific data series used. Please verify the accuracy and correctness of all programs before using them to trade.

Acknowledgements

Tomasz Janeczko, author of AmiBroker. Thank you for creating an outstanding program, and for graciously allowing use of materials published in the AmiBroker user documentation.

Bruce Robinson and William Barack. Thank you for the many stimulating discussions, encouraging comments, and conference presentations, all of which helped motivate and shape this book.

Robert Grigg. Thank you for the many stimulating discussions and exchange of ideas, and for raising awareness of AmiBroker in Australia. Particular thanks for permission to use the schematic of AmiBroker that you developed.

Contents

Preface and Introduction

This book is intended to be a tutorial. Topics included and detail covered involved compromises between elementary and advanced, narrow and broad, superficial and in-depth. Those chosen are practical examples of tasks that you will be performing regularly. For additional tutorials, visit the *written tutorials* and *video tutorials* at the AmiBroker website.

For a more complete reference, see the AmiBroker User's Guide and the AmiBroker Reference, both available at *www.amibroker.com*.

This book is task oriented, rather than software oriented. Many operations can be performed in more than one way. When making a choice, clarity was chosen over efficiency. This book is not intended to be read cover-to-cover, as a novel would be read. It is intended to be a series of tutorials and reminders covering operation of AmiBroker.

The examples use a lot of screen capture images, variously annotated with numbered steps, highlights, and arrows.

Intended audience

People wanting:

- Description of AmiBroker.
- Description of the capabilities of AmiBroker.
- Tutorial on installing and setting up AmiBroker.
- Tutorial on setting up databases with free data.
- Tutorial on setting up subscription end-of-day databases.
- Tutorial on setting up real-time data feeds.

- Tutorial on the basic charting.
- Tutorial and reference on the Graphical User Interface - GUI.
- Tutorial on applying formulas and indicators to charts.
- Tutorial on the AmiBroker Formula Language - AFL.
- Introduction to technical analysis.
- Introduction to design of trading systems.
- Tutorial on backtesting trading systems.
- Tutorial on optimizing trading systems.
- Tutorial on validating trading systems.
- Examples of trading systems.

Assumptions

While this book is intended to be a tutorial on AmiBroker, the reader is expected to be reasonably familiar with:

- Computer operations.
- Basic techniques used with the Windows operating system, such as the menu system, drag-and-drop, navigating through the file system.
- A basic text editor, such as Windows Notepad.
- A spreadsheet, such as Excel.
- Basic trading methods and terminology.

What you will find in this book

Section I – Getting Started.

Chapter 1 - An overview of AmiBroker - its features and capabilities.

Chapter 2 - Tutorials describing the installation of AmiBroker, setting up databases using free end-of-day and intraday data, setting up databases using subscription end-of-day data, and setting up databases using subscription real-time data.

Chapter 3 - There are ten examples of useful things you can do with AmiBroker in just a few minutes, even if you are using the trial version. These range from manipulating the charts to applying on-screen indicators to testing and optimizing trading systems. Everything is laid out, click-by-click, with screen captures to illustrate each step.

Section II – Charting.

Chapter 4 - AmiBroker chart structure.

Chapter 5 - Graphical User Interface.

Section III — Analysis.

Chapter 6 - A short introduction to technical analysis.

Chapter 7 - An overview of trading system development.

Chapter 8 - AmiBroker Formula Language, including its structure and syntax. AFL is used to write code to create your own custom charts and trading systems. This section is part tutorial and part programming manual.

Chapter 9 - AmiBroker's Analysis tools, including backtest, optimization, and walk forward validation.

Chapter 10 - Guidelines to help system development and ready-to-use programs you can use as starting points for your own system development.

What you will not find in the this book

AmiBroker has so much to offer that this book could easily have been expanded by several hundred pages. Readers looking for coverage of topics such as the Custom Backtester, Dynamic Data Exchange, low-level graphics, scripting, and the Automated Trading Interface will have to wait for another book. Readers looking for coverage of topics such as different types of trading systems, trading system testing, trading system validation, analysis of system health, determination of position size, and the creation of C++ routines and Dynamic Link Libraries are referred to the author's companion books, *Quantitative Trading Systems*, *Mean Reversion Trading Systems*, *Modeling Trading System Performance*.

Some conventions

AmiBroker can be used with any tradable, including stocks, mutual funds, closed end funds, exchange traded funds, commodities, futures, and Forex. For ease of writing and reading, the terms stock, symbol, issue, or ticker are often used to mean any tradable issue. When it is important to distinguish between different tradables, specific details are given.

AmiBroker is distributed and used world-wide. With the exception of setting up the database to reflect the local tradables and exchanges, most of the operations of AmiBroker are independent of location. The author of this book resides in the United States and most of his experience is with trading in the US, which may shape some of the perspective of this

book. Apologies in advance for any inadvertent mistakes or misleading statements due to his limited background – no discrimination of any kind is intended.

Versions used

AmiBroker 5.50.5.

AmiQuote 3.03.

AmiBroker is regularly expanded and improved. During the life of this edition of the book, the features, capabilities, screens, and commands, will undoubtedly change somewhat. Every attempt was made to write in such a way that the book will be useful for a long time and will not soon become obsolete as new versions of the program are released. But there will be changes in the program that are not reflected in the book.

For the latest and official documentation, please refer to the latest editions of the AmiBroker User's Guide, the on-line help files, and the tutorials, all of which can be found on the AmiBroker web site: *www.amibroker.com*.

The author

Dr. Howard Bandy:

- Has university degrees in mathematics, physics, engineering, and computer science.
- Has specialized in artificial intelligence, applied mathematics, modeling and simulation.
- Was professor of computer science and mathematics, and a university dean.
- Designed and programmed a well-known program for stock selection and timing.
- Was a senior research analyst for a CTA trading firm.

The second edition

The first edition of *Introduction to AmiBroker* was published in 2008 when the current version was 5.10. With the modifications, changes, and improvements to AmiBroker, particularly the new interface for the Automated Analysis features, a revision to this book became necessary.

Publishers who contract for printing and take delivery of the books have a trade-off to consider. In order to obtain a low price for the printing, a large order should be placed; in order to maintain flexibility and not risk wasting too much inventory when a new edition is published, a small inventory should be held. The first edition was about 600 pages and

a press run of several thousand copies was delivered to us. It was expensive to print and expensive to ship.

The audience for Introduction to AmiBroker is specialized, with customers for the book drawn primarily from people who are new customers for the software. Because a large portion of the prospective customer base has already been satisfied and AmiBroker will continue to change, the next press run of printed books would necessarily be smaller, and consequently more expensive. After considering all our alternatives, we decided to reduce the size of the book (primarily by reducing the reference section) and make the second edition available as a free download in the form of a pdf file. It has been formatted so that it will print on either letter size or A4 size paper. Please note that while the download is free for personal use, the book is not being placed into the public domain. Blue Owl Press holds the copyright. Distribution of the pdf file and of printed copies is permitted provided the entire content of each page, including the copyright notice, is retained and there is no cost, not even for materials used, to the recipient.

Readers who appreciate this book are encouraged to visit our websites and learn about the other materials available at *www.BlueOwlPress.com*

Chapter 1

Overview of AmiBroker

AmiBroker is a powerful, comprehensive trading system development platform. It has cutting edge charting and graphics, and fast, flexible and powerful portfolio-level backtesting, optimization, and automated walk forward validation. Its purpose is to help investors and traders identify profitable opportunities to buy and sell. It includes an extensive library of technical indicators that can be plotted along with the price chart as well as tested for profitability in a trading system. It has all the tools needed to chart, test, and trade stocks, exchange traded funds, mutual funds, commodities, and Forex.

AmiBroker has two primary modes of operation – charting and formula evaluation. The data it works with are the price and volume records of buy and sell transactions for stocks, mutual funds, and other tradable issues.

In its charting mode, historical price and volume data are displayed on the computer monitor along with technical indicators. The analyst looks for promising patterns and conditions.

In its formula evaluation mode, patterns, conditions, and rules are described using a programming language and written into a computer program. The program analyzes the price and volume data and reports on the profitability of the rules. When profitable trading systems have been found, it scans the group of stocks that are of interest to the trader

and lists the current buy and sell signals. If desired, AmiBroker can send orders based on these signals directly to a broker for execution.

AmiBroker is unique among trading system development platforms in that the same code displays the indicators, tests the profitability, issues the buy and sell signals and sends the orders. It is an easy transition from chart to analysis to execution.

AmiBroker runs under Windows, including versions 95, 98, Millennium, 2000, NT, XP, Vista, and Windows 7. Both 32 bit and 64 bit operating systems are supported.

Any system that runs Windows efficiently will run AmiBroker efficiently.

Professional and Standard Editions

- Both support intraday as well as end-of-day data
- Standard meets most needs
- Professional is required for
  - Tick charts and bars 15 seconds or shorter
  - MAE / MFE charts
  - 64 bit AmiBroker
  - Unlimited real-time quotes
  - Unlimited time and sales
  - Get real time data
  - Wait for backfill

Data Feeds

- Accepts data from any exchange in the world
- Close integration with major subscription data vendors
  - Norgate Premium Data
  - TC 2000
  - FastTrack
  - CSI
  - eSignal
  - DTN IQFeed
  - Interactive Brokers
  - myTrack
  - Quote Tracker
  - Any DDE-enabled data feed

- AmiQuote downloader provides access to free quotes from major world exchanges
  - Yahoo Finance
  - Google Finance
  - MSN Money Central
- Built-in importer for MetaStock format data
- ASCII import wizard reads any ASCII data format

Databases

- End-of-day data
- Intra-day data
- Any number of databases
- Any number of symbols
- Add or remove issues
- Split and distribution adjustment
- Unlimited history of price quotes
- Quotation editor
- Company information
- Fundamental information

Performance

- Multi-threaded operation
- Fast execution of AFL code
- Fast chart drawing
- Robust and stable

Charting

- Multiple chart panes
- Multiple time frames
- Live updating
- Windows GUI standards
- Intraday, daily, weekly, monthly
- Line, bar, candlestick
- Fast zooming and scrolling
- Custom or automatic scaling

- Built-in indicators
  - Simple moving average
  - Exponential moving average
  - Adaptive moving average
  - Rate of change (ROC)
  - Wilder's Relative Strength Indicator (RSI)
  - Moving Average Convergence-Divergence Oscillator (MACD)
  - On Balance Volume (OBV)
  - Commodity Channel Index (CCI)
  - Money Flow Index (MFI)
  - Bollinger Bands
  - Stochastics
  - Parabolic Stop and Reverse (SAR)
  - Relative strength
- Overlay indicators over price charts
- Overlay indicators over other indicators
- Configurable indicators
- Drag and drop tools
- Chart study tools
  - Trend lines
  - Regression channels
  - Text on chart
  - Fibonacci retracements
  - Gann squares
- Chart studies are saved with the chart
- Charts independently scalable

AFL (AmiBroker Formula Language)

- Designed for charting and trading
- Over 200 built-in building-block functions
  - Pattern-detection
  - Averages
  - Statistical
  - Pre-defined indicators
  - Data manipulation

    - Trade management
- Extensive library of pre-written code
- Create your own indicators, trading systems, and commentaries
- Single code base for indicators, systems, commentaries
- User-defined functions
- Unlimited variables
- Unlimited nesting of function calls
- Local and global variables
- Multiple time frames
- Indicator builder
    - Create and plot indicators
    - Control axes, scales, color, line style
- Show Buy and Sell signals on charts
- Code stored in clear-text
- Built-in formula editor
- Any editor can be used
- Debugging, tracing, and profiling tools
- Extendable through Dynamic Link Libraries (DLL)

Alerts

- Formula-based alerts
- Display to screen
- Play sound
- Send e-mail
- Send order to automated trade execution

Scanning

- Review your database for your buy and sell signals
- Time window can be specified

Exploring

- Search your database for conditions you specify
- Report results
- Multiple-key sorting

Backtesting

- Test profitability of trading system
- Single issue or group of issues you define
- Long, short, or both
- Buy and sell arrows on charts
- Built-in trailing exits and stops
- Realistic slippage and commission
- Portfolio level fully supported
- Position sizing
- Optimization
  - Full search of parameter space
  - Non-exhaustive search
    - Particle Swarm Optimization (PSO)
    - Covariance Matrix Adaptation Evolutionary Strategy (CMAE)
  - Single issue or portfolio
- 3-D presentation of results
- Multiple time frame
- Equity curve creation
- Equity curve input to trading system
- Walk forward testing
- Over 20 built-in metrics
- User-definable metrics
- Fast execution
- Detailed reporting
- Report explorer
- Export results for further analysis

Automated Trading

- Interfaces to on-line brokers

Scripting

- Jscript (JavaScript)
- VBScript (Visual Basic Scripting)
- Scripts embeddable in AFL programs
- OLE (Object Linking and Embedding) Automation

- Call COM (Component Object Model) objects from AFL
- Create and automate database management tools

Configurability

- Almost everything is configurable and customizable
- Standard Windows operations
- Charts can be arranged as you wish them to be
- Not tied to a specific exchange or data provider
- Indicators are parameter driven

Account Manager

- Track your investments
- Calculate commissions, dividends, deposits, withdrawals

Internet Integration

- Built-in web browser for company research
- Configurable settings

Pricing (as of August 2012)

- Fully functional, no cost, 30 day trial
- New one-time license:
  - Standard Edition: $199
  - Professional Edition: $279
- Free upgrades for one year from purchase
- Renewal license (to continue upgrades): half price
- Install on all your personal computers
- Access to members only websites

Ordering

Visit *www.amibroker.com/order.php*. This secure site accepts your payment using any major credit card, check, or bank transfer.

Delivery

After paying the registration fee, you will receive a personalized keyfile by e-mail. Installing the keyfile converts your trialware version to a registered version. No other downloads are required. If desired, a backup CD ROM can be ordered.

Support

- Unlimited support for installation issues via e-mail
- Unlimited support for basic usage issues via e-mail
- Interactive help anywhere in AmiBroker (F1 key)
- AmiBroker website support page
  www.amibroker.com/support.html
- On-line video tutorials (the URL is for one example - there are many)
  www.amibroker.com/video/uicustomize.html
- On-line Users Guide
  www.amibroker.com/guide/
- Knowledge base – provided by AmiBroker
  www.amibroker.com/kb/
- Users Knowledge Base – provided by user community
  www.amibroker.org/userkb/
- Yahoo discussion forum – over 12,000 members – about 1000 messages a month
  finance.groups.yahoo.com/group/amibroker/

Chapter 2

Installation

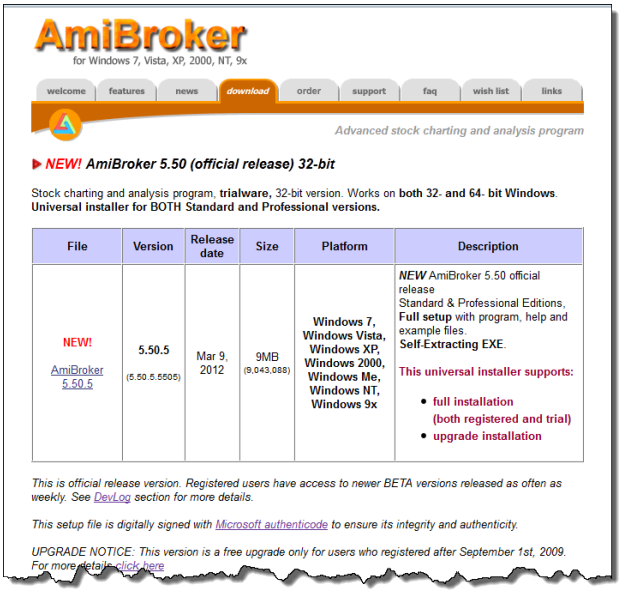
AmiBroker runs under Windows, including versions 95, 98, Millennium, 2000, NT, XP, Vista, and Windows 7. Both 32 bit and 64 bit operating systems are supported.

All versions of AmiBroker – trial and registered, standard and professional, end-of-day and real-time – begin with a visit to the AmiBroker web site and downloading the installation file.

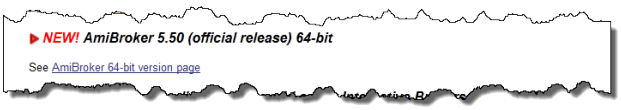
Using your Internet browser, visit *www.amibroker.com*, then select the download tab. The universal installer for the 32 bit version of AmiBroker supports:

- the trial version
- full and upgrade registered versions
- standard and professional version of AmiBroker
- both 32 bit and 64 bit versions of Windows

Download the file, about 9 MB, to your computer. When the download is complete, double-click the installation file (AmiBroker550.exe, or the latest version) to run the AmiBroker Setup Wizard. Follow the directions and answer the prompts.



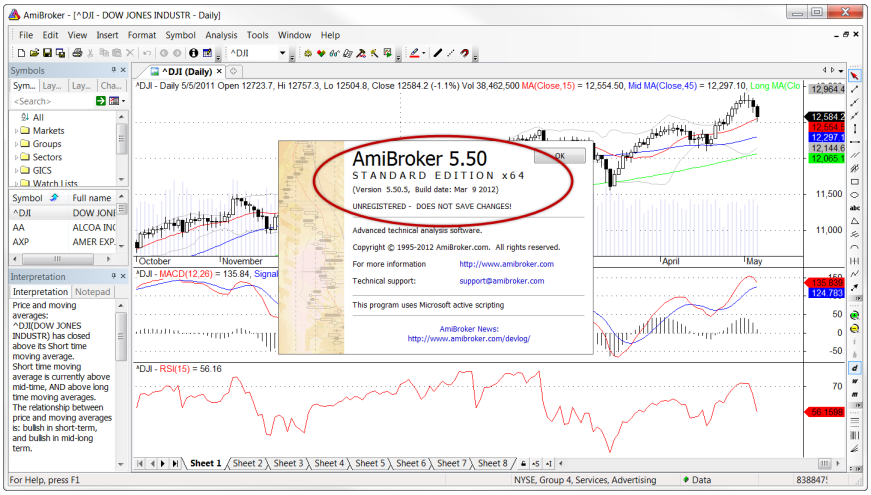
If you want the 64 bit version of AmiBroker, you will need the 64 bit version of Windows and the Professional version of AmiBroker. Click the AmiBroker 64 bit installer link to take you to the page with information and the downloader for the 64 bit program.



Both AmiBroker and this author recommend accepting the default options and file locations during installation.

When the installation is complete, launch AmiBroker. The installation will have created an AmiBroker icon on your desktop. Just double-click that. When AmiBroker starts, it displays a message that it is Standard Edition and Unregistered. This is the trial version, but will be instantly converted to Registered (and Professional Edition, if you requested that) when your AmiBroker Registration file is processed. In the mean time, AmiBroker is ready for your use in Trial mode.

AmiBroker 32 bit trial version



AmiBroker 64 bit trial version

AmiBroker in Trial Mode

The trial version of AmiBroker is fully functional with a few exceptions.

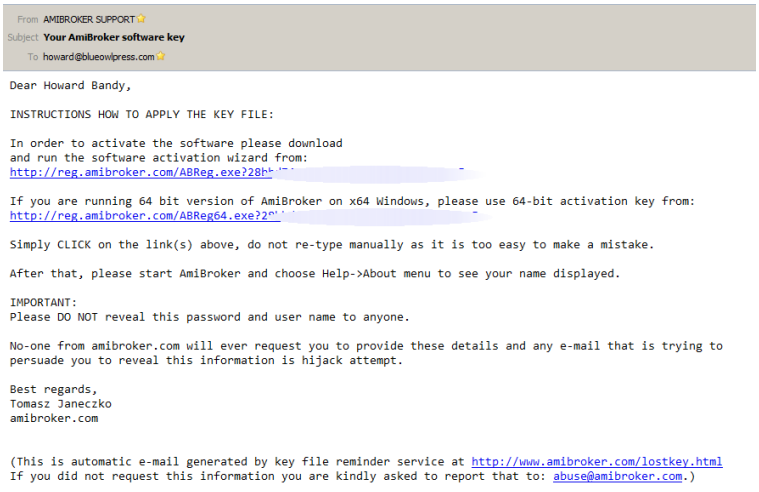
- The trial version will run only 30 days. To run longer than that, it must be registered.
- When processing multiple symbols, the maximum in the trial version is five; the registered version handles an unlimited number.
- Changes, particularly to the database, will not be saved. Also, if you modify any of the chart panes, the changes you made will not be saved when AmiBroker closes.

To perform most of the operations described from this point on, you will need a registered version of AmiBroker. But you can do useful work and evaluate AmiBroker using the default database and the trial version. Chapter 3 - 30 Minutes to Useful Results, has some suggestions.

Registering AmiBroker

To purchase a license for AmiBroker, go to the AmiBroker website *www.amibroker.com* and click the BuyNow button. After selecting the products and versions you want, and pay using your credit card, you will be sent an email that contains links to the files that unlock the full capability of AmiBroker.

You can have both the 32 bit and 64 bit versions of AmiBroker installed on your computer. But unless you have a specific need for both, I recommend that you pick one of them - the 64 bit version is preferable if your system will support it.



From AMIBROKER SUPPORT
Subject **Your AmiBroker software key**
To howard@blueowlpress.com

Dear Howard Bandy,

INSTRUCTIONS HOW TO APPLY THE KEY FILE:

In order to activate the software please download
and run the software activation wizard from:
http://reg.amibroker.com/ABReg.exe?28h

If you are running 64 bit version of AmiBroker on x64 Windows, please use 64-bit activation key from:
http://reg.amibroker.com/ABReg64.exe?2

Simply CLICK on the link(s) above, do not re-type manually as it is too easy to make a mistake.

After that, please start AmiBroker and choose Help->About menu to see your name displayed.

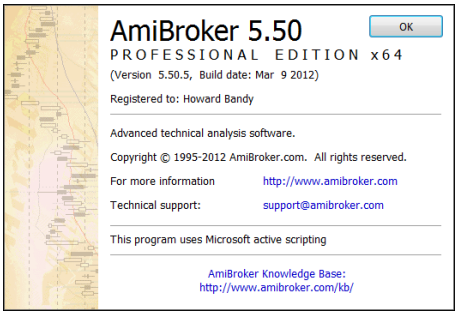
IMPORTANT:
Please DO NOT reveal this password and user name to anyone.

No-one from amibroker.com will ever request you to provide these details and any e-mail that is trying to
persuade you to reveal this information is hijack attempt.

Best regards,
Tomasz Janeczko
amibroker.com

(This is automatic e-mail generated by key file reminder service at http://www.amibroker.com/lostkey.html
If you did not request this information you are kindly asked to report that to: abuse@amibroker.com.)

Download the registration file -- ABReg.exe for 32 bit, ABReg64.exe for 64 bit. With AmiBroker closed, run it. Reopen AmiBroker and click Help > About AmiBroker. The screen will show that you have successfully registered.



Default Database

When you install AmiBroker, a small database is also installed. Its name is Data and it contains end-of-day price history for the stocks in the Dow Jones Industrial Average and for the average itself. The date range is short and the quotes are not up-to-date. But it is useful for verifying that the installation was successful and for investigating AmiBroker.

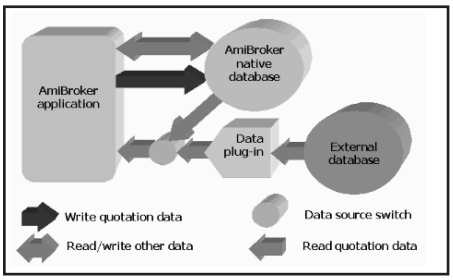
You will definitely want a longer history and more issues. You can have as many databases as you wish. Data is one. You can expand it or add other databases.

I recommend creating a copy of the original Data database as it was distributed. To do that, follow these steps. You will need a registered version to do this.

1. Using Windows Explorer, navigate to the AmiBroker directory. If you accepted all the defaults during installation, it is C:\Program Files\AmiBroker.
2. With Explorer, using the File menu, click New, then Folder. A folder named New Folder will be created.
3. Using Explorer, rename that to the name you want for the original database, for example DataAsDistributed.
4. Using AmiBroker and its File menu, click SaveDataBaseAs, select the folder you just created, DataAsDistributed, click OK. AmiBroker will make a copy of the Data database in the folder DataAsDistributed.

Block Diagram of Database

As the block diagram shows, AmiBroker has two levels of database files - native and external.



The native database is maintained by AmiBroker and AmiQuote, and can be modified by you using the Quote Editor within AmiBroker. The Data database is an AmiBroker native database.

When you subscribe to a data provider, that provider will set up an external database that they exclusively maintain. As AmiBroker needs quotes to place on a chart or process in a backtest, the Data Plug-in reads the external database and passes the data to AmiBroker.

Database Setup - End-of-Day

Setting up a database is a two step operation. Step one is establishing the database within AmiBroker and specifying which tickers will be stored in it. Step two is filling the database with historical price quotations. Depending on the processes being used, you may be aware of the two separate steps, or they may be combined and appear to you as a single step.

If you will be subscribing to a data service, such as Norgate Premium Data, the installation procedure for that service will set up a database specifically for quotes from that service and will be maintained by that service. We will discuss setting up a Premium Data database in a few pages.

If you want to use one of the free data providers, such as Yahoo, you can either expand Data or create a new database.

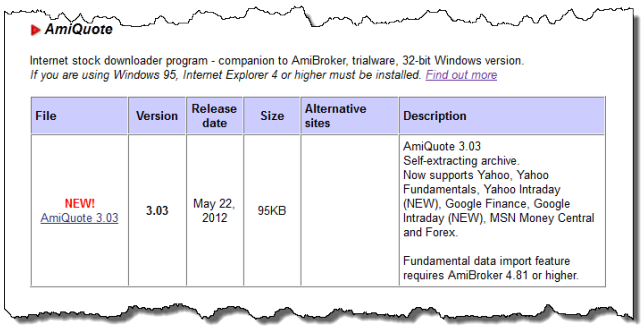
Free data may have no monetary cost, but consider the time and effort you will spend maintaining the free data. Subscription data services provide not only the price and volume data, but also manage tasks such as splits and correcting data errors. I recommend subscribing to a high quality data provider. To my mind, the approximately $40 per month they charge is well worth while. Visit the AmiBroker website for a list of

supported data vendors, their products, and their fees. Choose one that provides the data you need - stocks, mutual funds, ETFs, futures, Forex, end-of-day, real-time. *www.amibroker.com/guide/h_quotes.html*

Free Database using AmiQuote

AmiQuote is a companion program to AmiBroker that manages downloading of price quotes, from either Yahoo, Google, or msn, and storing those quotes in the AmiBroker database.

Return to *www.amibroker.com/download.html* and download the setup file for AmiQuote. The current version is 3.03.



AmiQuote

Internet stock downloader program - companion to AmiBroker, trialware, 32-bit Windows version.
If you are using Windows 95, Internet Explorer 4 or higher must be installed. Find out more

File	Version	Release date	Size	Alternative sites	Description
NEW! AmiQuote 3.03	3.03	May 22, 2012	95KB		AmiQuote 3.03 Self-extracting archive. Now supports Yahoo, Yahoo Fundamentals, Yahoo Intraday (NEW), Google Finance, Google Intraday (NEW), MSN Money Central and Forex. Fundamental data import feature requires AmiBroker 4.81 or higher.

Double-click aq3030.exe and follow the installation instructions, accepting the defaults. A readme file will open and describe the latest features.

The procedure described here uses only the ticker symbol. If you are serious enough to want the full name and industry category stored with the symbol, I assume you will be using a subscription data service where all of that is provided for you automatically for the majority of your data. The procedures explained here will let you create data files for those few issues that your primary vendor does not supply.

Begin by deciding which issues you want in your database. Find or create a list of their tickers. This does not need to be the final list – you will be able to add additional tickers and historical data at any time in the future. For example, you might want all the stocks in the Russell 1000 index – the 1000 largest capitalization stocks listed on US exchanges. (According to the Russell Investments website, *www.russell.com*, the Russell 1000 represents approximately 92% of the US equities market. That website also has a list of the current components of all the Russell indices.) Other lists you might want are the stocks in the S&P 100, S&P 500, and Nasdaq 100. Visit *www.blueowlpress.com/WordPress/* and click Resources for links to lists of members of various categories.

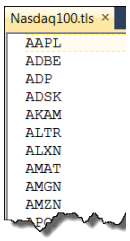
Many stocks are constituents of several indexes or lists. You only need one copy of the historic price data for each issue. AmiBroker manages the lists and associations.

Ticker Only

This example updates the database named Data with data for the tickers in the Nasdaq 100 list.

1. Get the current list of constituents of the Nasdaq 100 from the London Stock Exchange webpage.
   www.londonstockexchange.com/exchange/prices-and-markets/international-markets/indices/home/nasdaq-100.html

2. Using a simple text editor, such as NotePad, or a spreadsheet, create a file with one ticker per line. The tickers should be in all capital letters. Each ticker must be spelled the same as your data provider spells it. Save that file in the AmiQuote directory with a meaningful file name, such as Nasdaq100.tls. If you accepted all the defaults during installation, that directory is C:\Program Files\AmiBroker\AmiQuote. Tls files are ordinary text files that are recognized by AmiBroker and AmiQuote as containing lists of ticker symbols.



3. Open AmiBroker.

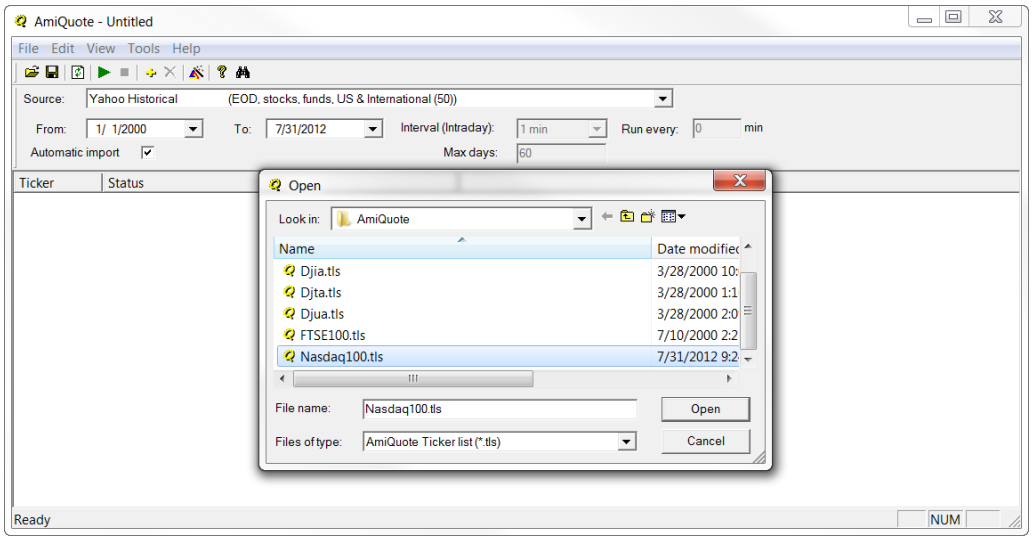
4. Using the AmiBroker File menu, click Open Database, select the Data database, click OK.

5. Leave AmiBroker open as the next steps are performed.

6. Decide the date you want the historical data to begin. If you plan to do extensive trading system development, I suggest having at least ten years of data available. The characteristics of the markets changed after the October 1987 crash, so data before that date has limited value in developing systems for current conditions. A starting date (From Date) of 1/1/1995 or 1/1/2000 might be reasonable. Use today's date for the end date (the To Date).

7. Using the Windows Start menu, select the AmiQuote program and run it. You will find it under All Programs > AmiBroker > AmiQuote > AmiQuote.

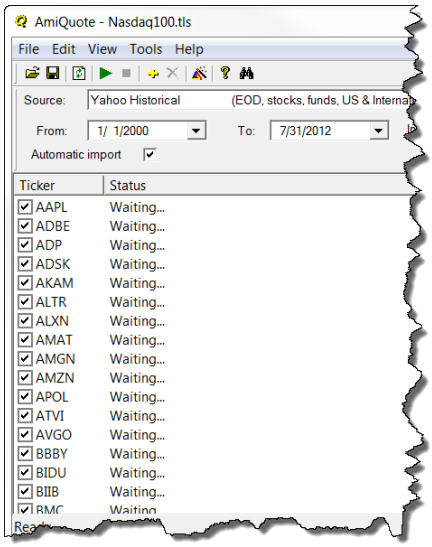
8. Using AmiQuote's File menu, select Open. Then select the file with the tickers you want to get historical data for, Nasdaq100.tls, and click Open.



Be sure that AmiBroker is running and that the database you want updated is the current database. The name of the database in current use is displayed in the status bar in the bottom right-hand corner of the AmiBroker window. If necessary, use the AmiBroker File menu and open the desired database.

9. Be sure that the From and To dates are set as you want them, and that AmiQuote's Automatic Import box is Checked.

10. Click the Green Triangle to begin the download.



You have asked for over 1,200 years of data, each with 252 sets of Open, High, Low, Close, and Volume. Depending on the speed of your Internet connection, it will take a few minutes to a few hours for the download to complete.

11. AmiQuote will inform you of the progress of the download. If the amount of data stretches the capacity of your communications or your patience, divide the Nasdaq100.tls file into several smaller files and work with each part in turn. As soon as the data for any individual stock has been downloaded, it is immediately available for use within AmiBroker. You may continue to work with AmiBroker while downloading proceeds.

    When the download is complete, scroll through the Ticker List in AmiQuote. If there were problems with any of the tickers, there will be messages displayed. A common problem is that the spelling of the ticker in the tls file is not the same as the spelling the data provider uses. For example, the list supplied by the exchange used a period when specifying the Class A stock, while Yahoo uses a dash character. Make a list of all the tickers that did not download properly, spelling them as the data provider spells them, and create a file named NewTickers.tls Add the tickers of any other stocks you want. Using the same procedure, have AmiQuote download the new tickers. A good source for researching companies and verifying tickers that will be downloaded from Yahoo is the Yahoo financial site. *finance.yahoo.com/*

12. Close AmiQuote.
13. In AmiBroker, using the File menu, click Save Database.

Ticker, Full Name, Other Data

Rather than do this using free data, subscribe to one of the commercial data provider. In the process of setting up their database, they will load the full name of the issue and assign each stock to its proper market, industry, and group. Most of the services will store additional information, such as number of shares outstanding, dividend, and so forth, that you would have difficulty finding yourself.

Additional Symbols

Whenever you want to add issues to your free data database, use this procedure:

1. Enter their ticker symbols, one per line, in the file NewTickers.tls. (It is convenient to keep all tls files in the AmiQuote directory.)
2. With AmiBroker running, open the database you want to add to.
3. Run AmiQuote.
4. Using AmiQuote's File menu, open NewTickers.tls.
5. Set the From Date and To Date.
6. Check the Automatic Import box.
7. Click the Green Arrow to begin the download.
8. When the download is finished, close AmiQuote.
9. In AmiBroker, Save database.

Adding Full Name

If you want to record the full name of the issue:

1. In AmiBroker, using the Symbol menu, select Information. The information window for this stock will open.
2. Edit the Full Name field.

Assigning Categories

If you want to assign categories, in AmiBroker either:

1. Using the Symbol menu, select Information.
2. In the Categories section, make whatever assignments you wish for the one ticker you are working with.

Or:

1. Using the Symbol menu, select Organize Assignments.
2. Make whatever assignments you wish for the list of tickers you select.

Data Updates

Once the database has been established and the historical data loaded, maintaining the data by adding the latest quotes is very easy. In AmiBroker, using the Tools menu, select AutoUpdateQuotes. AmiBroker will examine the database that is open, determine the date range needed to bring the quotes for all the issues in this database up to current date, and call AmiQuote to do the download. Since only a few days data is needed for each ticker, the process takes only a short time.

Sources of Free Data

The examples above illustrate using Yahoo as the source of historical data. There are other sources. For US markets, msn and Google also provide free end-of-day historical data. Google recently began offering free intra-day data. If you wish to use either of those sources, use the pull down menu in AmiQuote to select one of them instead of Yahoo. Keep in mind that consistency is important. Different data suppliers will have different ways of preparing and presenting the data. For example, the volume multiplier may be different, or the ticker spelling may be different. After you load your database with quotes from one supplier, keep the database up-to-date using that same supplier.

ASCII Import

There are two methods of importing ASCII data: the ASCII Import Wizard (file menu, Import Wizard), and the full ASCII Importer (file menu, Import ASCII). The wizard is good for one-off imports. Its features are a subset of the full importer. If you will be importing the same files regularly, set up the full importer. (Note: the ASCII Import Wizard can create a format definition file for later use with the ASCII Importer.)

Before you start, decide which database you want the imported data to go into. If you are just practicing, create a test database and open it. Do not import into your high quality database until you are confident the import procedure is working correctly.

Download the data files and, if necessary, unzip or expand them. Open the files using a text editor such as Notepad. There should be one quote per line, fields separated by space, semicolon, or comma.

There may be a header line describing the fields. If not, identify the fields yourself. If necessary, go to the site from which the data was obtained and read their documentation.

The date field is the trickiest. The wizard understands many formats. If your data is in one of them, all you need to do is identify the order of month, day, and year. The codes are DMY, MDY, YMD. If your data does not follow one of the recognized date formats, you will need to reformat the date (for example, by using a spreadsheet) and rewrite the data file before proceeding.

The year can be either four digits or two (the final two of the year). The month can be either two digits or three characters (such as Dec or Jan). The day is two digits. The separator can be / (slash), \ (backslash), - (minus), or not separated at all.

Assume the date for one quote is December 31, 2000. YMD formats that are recognized include:

20001231
2000-12-31
2000/12/31
2000-Dec-31
001231
00-12-31
00/12/31
00\12\31

If your data has a ticker symbol as a field in each line, the data will be stored under that symbol. If there is not an explicit ticker, the file name will be used. If your data file is downloaded as, for example, IBM.csv, the data will be stored as IBM. If your data file is downloaded with a generic name, such as Table.csv (Yahoo uses this file name for all its individual downloads), rename it before proceeding.

Yahoo Historical Data Download

The Yahoo site is a good place to get stock and index historical data.
finance.yahoo.com

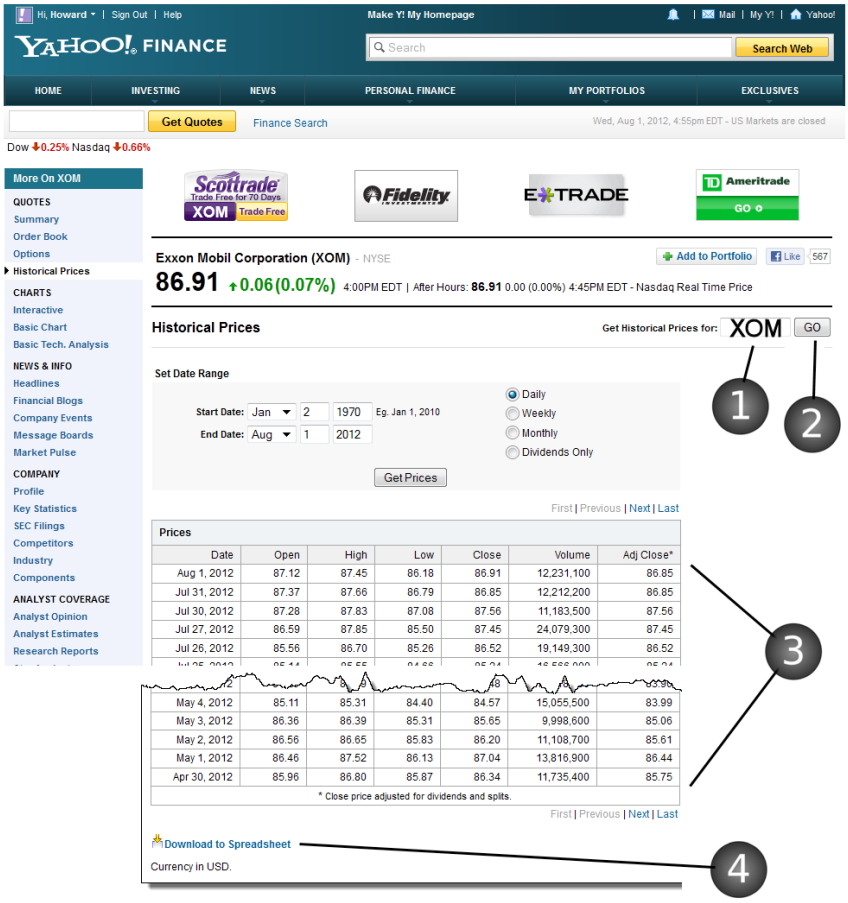
Using the investing tab, select stocks. Under research tools, click historical quotes to get to the page with historical data.

Follow these steps to download the historical data for a stock or index:

1. Enter the ticker (say it is XOM)
2. Click GO
3. The screen will display the most recent data
4. Click Download to Spreadsheet.

   On your desktop, or wherever your default directory to receive downloads is, you will find a file named Table.csv.

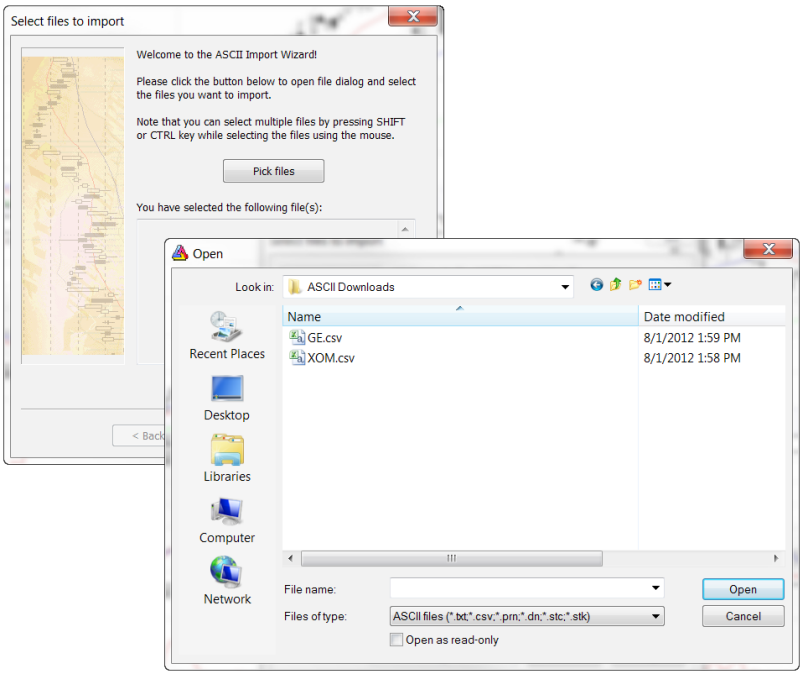
5. Rename Table.csv to XOM.csv. Use capital letters if you want your symbol to be in capitals.
6. Proceed to import XOM.csv into AmiBroker using either the ASCII Import Wizard (see next section) or Import ASCII.



ASCII Import Wizard

The following steps illustrate how to use the ASCII Import Wizard. Before beginning, a database named Test has been created and opened. It already has a few symbols and data for them. Data has been down loaded from Yahoo for two stocks, GE and XOM, but not yet imported. Since Yahoo downloads all data using the file name Table.csv, the downloaded file was renamed between downloads, so the files are GE.csv and XOM.csv.

1. Using the File menu, select Import Wizard.
2. When the dialog box opens, click Pick Files.
3. Navigate to the directory holding the files you have downloaded.

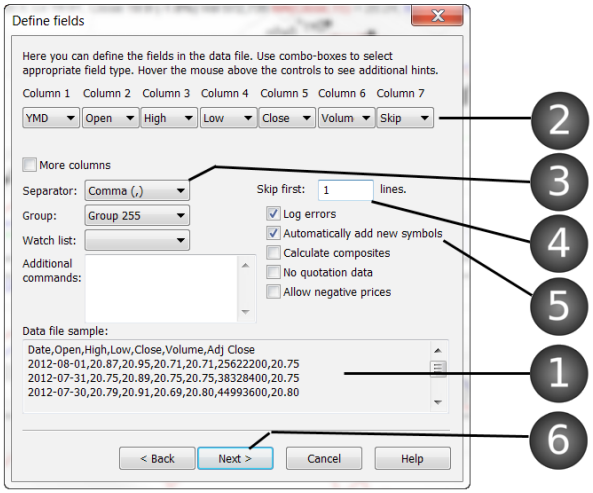


4. Since the files have identical formats, you can process multiple files in one pass, so select both GE.csv and XOM.csv. (Click one file, press the Control key and click the other.)
5. Click Open.

The Define Fields dialog box will open.

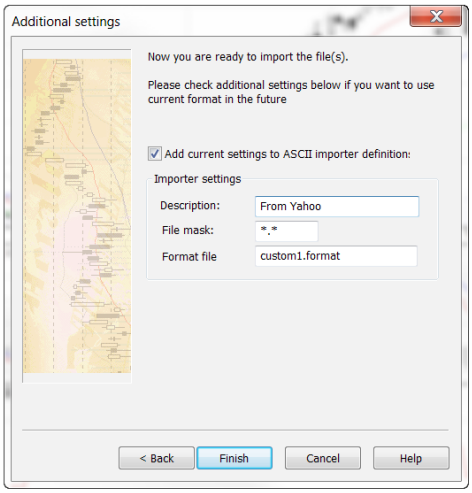
1. Note the format of the data in the .csv files.

2. Using the pull-down menus, define the fields for each element of data. Each element in the entire row of data must be defined. Select SKIP to ignore data in the csv file that you do not want in your AmiBroker database.
3. Select comma as the separator.
4. Skip 1 line - the header.
5. Check Automatically add new symbols.
6. Click Next.



The Additional Settings screen will come up. It is on this screen that you can save the format settings that you just used. They will not help for future uses of the ASCII Import Wizard, but they will be available in the full ASCII importer. To do this, Check Add current settings to ASCII importer definitions, and give meaningful Description and File Name - such as From Yahoo. You will see your format as we look at the full ASCII Importer in the next section.

7. Click Finish.



The data will be converted from ASCII, imported into the open database, and be immediately ready for use in AmiBroker.

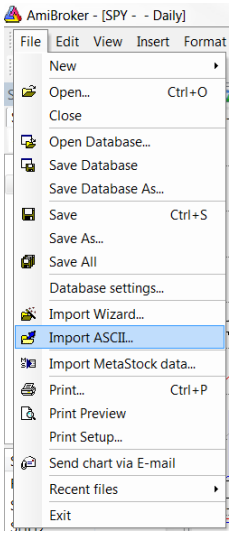
ASCII Importer

The full ASCII Importer is much more efficient, has more capabilities, and is more complex. Rather than having to fill in a form each time you want to import an ASCII file, you can set up a format definition file, or use one of the pre-defined files. The file that was created for the import done manually using the wizard in the previous section is:

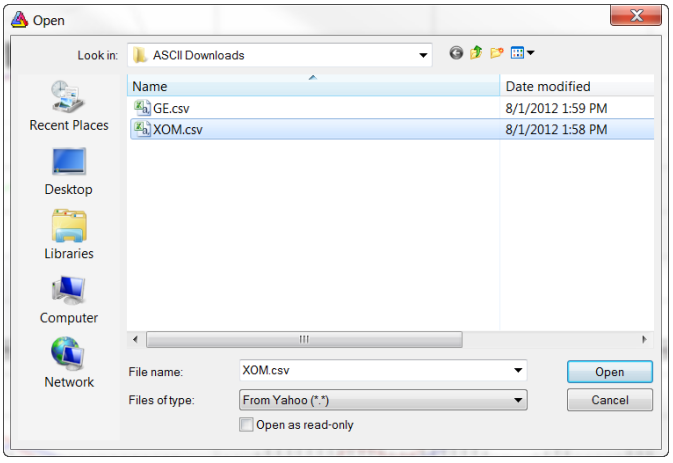
```
# Format definition file generated automatically
# by AmiBroker's ASCII Import Wizard
$FORMAT Date_YMD, Open, High, Low, Close, Volume, Skip
$SKIPLINES 1
$SEPARATOR ,
$CONT 1
$GROUP 255
$AUTOADD 1
$DEBUG 1
```

The definition files are stored in the C:\Program Files\AmiBroker\Formats directory. By copying the format file that you will be using regularly to be Default.Format, all that is necessary to import ASCII files is:

1. Using the File Menu, select Import ASCII.



2. On the Open screen, select the file you want to import.
3. Enter the file type, such as From Yahoo.
4. Click Open.



The file will be converted, imported, and made available within AmiBroker for immediate use.

The options available for controlling the processing of ASCII data are extensive. Read about them all in the AmiBroker User's Guide.

Subscription Data Providers

Free data has a cost to you – the cost of checking the data for missing quotes, bad values, unadjusted splits, and so forth, and correcting those problems. Subscription data services, such as Premium Data, CSI, and others, provide that maintenance for you and deliver cleaner, more consistent data with less effort on your part. End-of-day stock data costs about $40 per month – more for some services, less for others. Some end-of-day vendors include mutual funds at no charge; others charge extra. Commodities and futures data is an additional cost, depending in part on the fees the exchanges that clear those trades and report that data charge. Some subscription vendors provide custom indexes, custom indicators, and have a wider selection of issues – and often charge an extra amount for those. Some services charge a one-time fee for loading the historical data, others do not.

If you plan to use a subscription service, be aware that each service has its own database format. Once you begin with one subscription service, they will be the only service that can keep your data in that database up-to-date.

The installation procedure for each service will be unique to that service, but it is generally simple and well documented. Typically, the installation does three things for you:

1. Establishes the external database and the communications between the database and AmiBroker.
2. Loads the definitions of the tickers, including the ticker symbol, the full name, and the industry assignment.
3. Loads the historical price data.

After the close of trading each day, you initiate communication with the data vendor. The database is brought up-to-date, and errors and inconsistencies are corrected and re-downloaded, all automatically. Some vendors provide intra-day updating – treating the latest price as though it is the closing price. At subsequent updatings, the earlier prices are replaced by the more recent ones.

Norgate Premium Data - End-of-Day

Norgate Investor Service provides quality end-of-day data for stock markets in Australia (ASX), Asia (SGX), and the United States (NYSE, NASDAQ, AMEX, OTC-BB, PinkSheets). Extensive historical data is available, including data for delisted stocks (an important feature for those interested in studying survivorship bias).

Each data provider will have their own specific details and methods, but the end result is the same – an end-of-day database that is established and loaded with historical data by the data vendor. Tickers, full names, industry associations, and other information related to that stock are all provided by the vendor, stored in the external database for that subscription, and made available to AmiBroker through an interface that is transparent to you as a user.

The data integrates very well with the symbol lists, and includes over 900 pre-built and automatically maintained watchlists for the Australian and US databases. Hourly snapshot data is available for the ASX and SGX.

Data is provided in MetaStock data format. (If you are installing Premium Data as an AmiBroker database, you will not be aware of the format of the external database.)

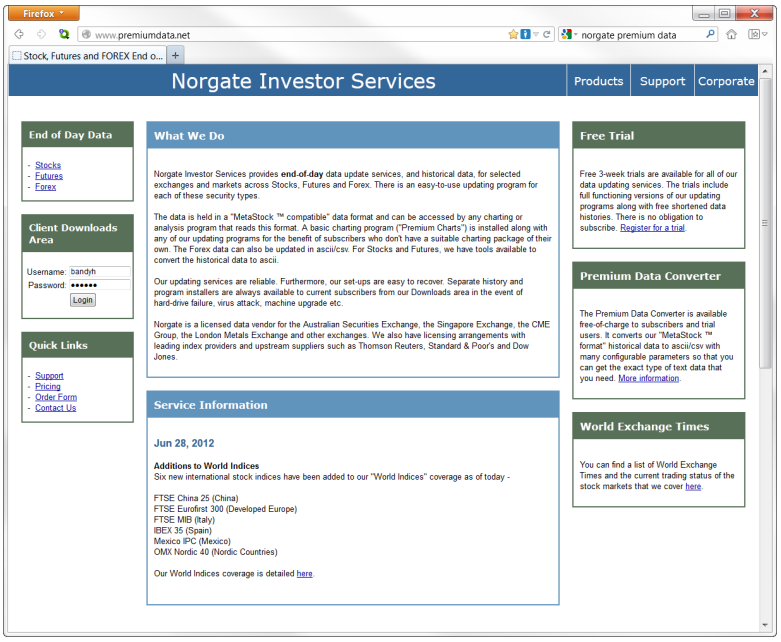
Norgate also provides end-of-day data for 113 liquid futures markets drawn from 15 exchanges. Futures data is available in individual, spliced continuous, and backadjusted continuous contract forms. All the fields required for serious backtesting (margin, point value, tick size, etc.) are populated. Data is provided in ASCII and MetaStock data formats.

Norgate is in the process of improving their integration with AmiBroker. A new native AmiBroker data plugin that will provide additional speed and features is under development. Visit *www.premiumdata.net/support/amibroker.php* for the latest details.

Free trials are available.

The next pages discuss setting up Norgate end-of-day data for the US stock market.

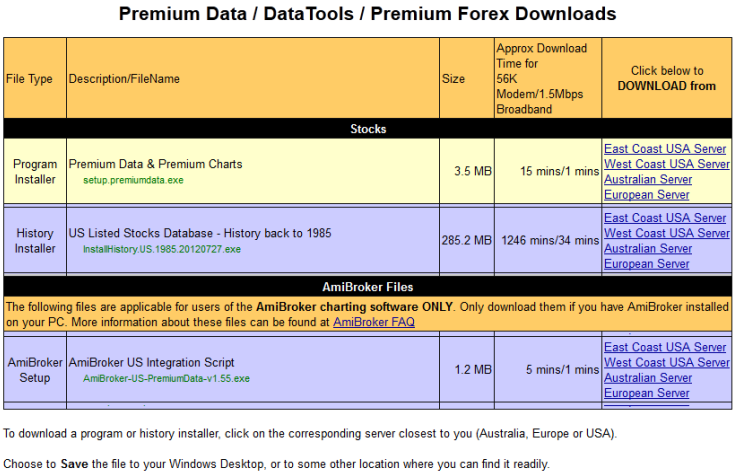
1. Sign up for a subscription or free trial at their web site: *www.premiumdata.net*.



2. You will receive an e-mail with your user name and password, and a link to the download page.

3. Go to that web page and login.

4. The services you have subscribed to will be noted in the download screen. You will need to download at least three files:
   A. Program Installer
   B. History Installer
   C. AmiBroker Setup.



Premium Data / DataTools / Premium Forex Downloads

File Type	Description/FileName	Size	Approx Download Time for 56K Modem/1.5Mbps Broadband	Click below to **DOWNLOAD from**
Stocks				
Program Installer	Premium Data & Premium Charts setup.premiumdata.exe	3.5 MB	15 mins/1 mins	East Coast USA Server West Coast USA Server Australian Server European Server
History Installer	US Listed Stocks Database - History back to 1985 InstallHistory.US.1985.20120727.exe	285.2 MB	1246 mins/34 mins	East Coast USA Server West Coast USA Server Australian Server European Server
AmiBroker Files				
The following files are applicable for users of the **AmiBroker charting software ONLY**. Only download them if you have AmiBroker installed on your PC. More information about these files can be found at AmiBroker FAQ				
AmiBroker Setup	AmiBroker US Integration Script AmiBroker-US-PremiumData-v1.55.exe	1.2 MB	5 mins/1 mins	East Coast USA Server West Coast USA Server Australian Server European Server

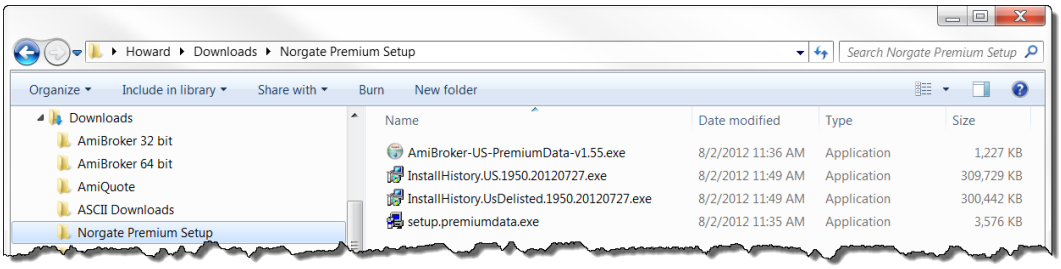
To download a program or history installer, click on the corresponding server closest to you (Australia, Europe or USA).

Choose to **Save** the file to your Windows Desktop, or to some other location where you can find it readily.

Click on the server nearest to you (Australia, Europe or USA).

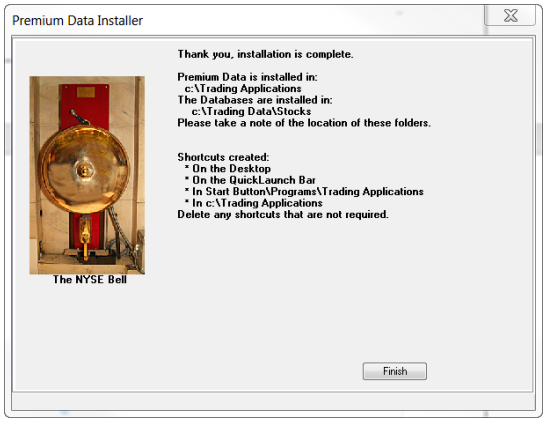
5. Save the files and accept all the defaults for file locations.
6. The Program Installer and History Installer must be in the same subdirectory on your computer.
7. Insure that AmiBroker is not running.
8. Locate the program installer, run it, and follow the prompts. The program installer will recognise the history installer and offer to install the history for you as part of the process. If the files are too large for your communications, you may prefer to order a data history CD which will be sent out to you by ordinary mail.

9. Execute the files in this order:
   1) Program Installer. (setup.premiumdata.exe)
   2) Historical Data Installer. (InstallHistory.US.xxx.exe)
   3) AmiBroker Setup. (AmiBroker-US-PremiumData-v1.55.exe)



The program installer will ask if you want to install the history at this time. If you agree, it automatically runs the second step for you.

When it finishes, the installer will tell you where it put the files.



Then run the AmiBroker Setup. When the AmiBroker Setup runs, it will open a browser-based FAQ page explaining how to install the interface necessary for AmiBroker to use the Premium Data files.
AmiBroker Setup will establish a database in the AmiBroker directory, but it will not be usable yet. Here is how to connect the two:

10. Start AmiBroker.
11. Using the File menu, select Open Database, select US-PremiumData, click OK.

12. Using the Tools menu, click US-Premium Data. This will run a script with seven stages that takes several minutes.
    If it is not already open, the program will instruct you to have the Premium Data database open.
    When the script finishes, use the File menu and Save Database.
    AmiBroker is ready to run with Norgate Premium Data.

13. Here is AmiBroker with SPY in the chart window.

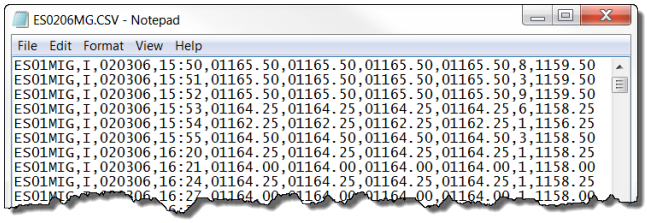
Database Setup - Intra-day and Real-Time Data

Intra-day data is data with bars shorter than a full trading day. Taking great care and using a data vendor that supports it, it is possible to combine intra-day and end-of-day data in the same database. But most AmiBroker users who use intra-day data maintain a separate intra-day database. The next few pages describe setting up intra-day and realtime data.

Free Intra-day Historical Data

If you are interested in developing systems with intra-day data, but are not ready to commit to a real-time data subscription, you can download intra-day bars into an AmiBroker database and work with them. If your systems test out to be profitable, you will eventually need real-time data to execute them.

There are several sources of free intra-day historical data. Visit *www.blueowlpress.com/WordPress/*, click Resources, then scroll down to the Data area for links to some of them. AN Futures sells intra-day data for the e-mini S&P 500, e-mini NASDAQ 100, e-mini Russell 2000, Euro, and Yen in 1 minute bar and tick resolution. Visit them at: *www.anfutures.com*. They have a demo file, ES0206MG.csv with 1 minute bars for the e-mini S&P June 2002 contract – 3/6/2002 through 6/12/2002. The third column is the date, with 020306 meant to be 2002, March, 6.

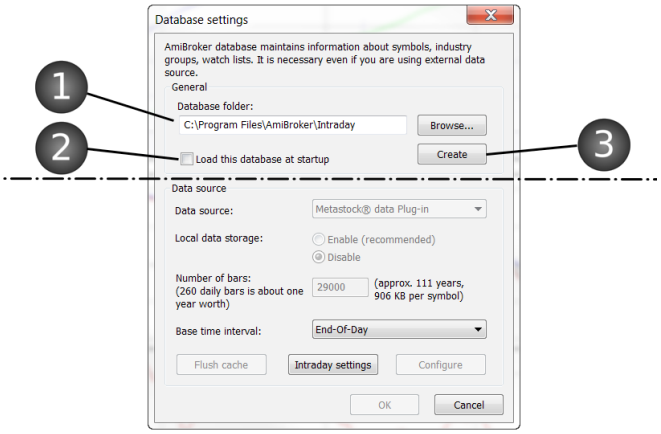


The following procedure will set up a database into which we will load the e-mini S&P intra-day data. This is the same procedure used to set up any manually installed native database, adjusting parameters such as the base time interval as necessary.

In AmiBroker, from the File menu, select New, then Database.

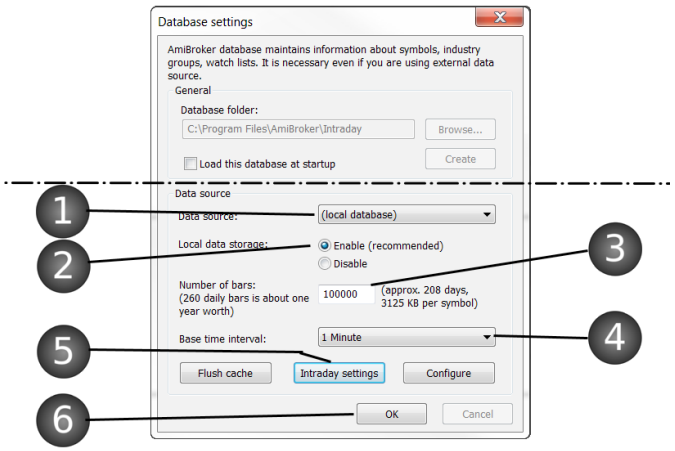
Working first in the top half of the window:

1. In Destination Folder, choose a meaningful name for the database – such as Intra-day.
2. Uncheck Load this database at startup.
3. Click Create.



Now working in the bottom half of the window:

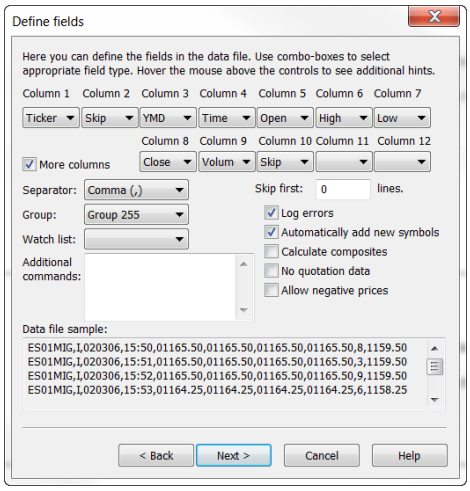
1. Data Source: (local database)
2. Local data storage: Enable
3. Number of bars: 100000
4. Base time interval: 1 minute
5. Accept the defaults for Intraday settings
6. Click OK.



The database is established, but it has no data in it. The ASCII importer will load the data from the csv file using the ASCII Import Wizard in the following procedure:

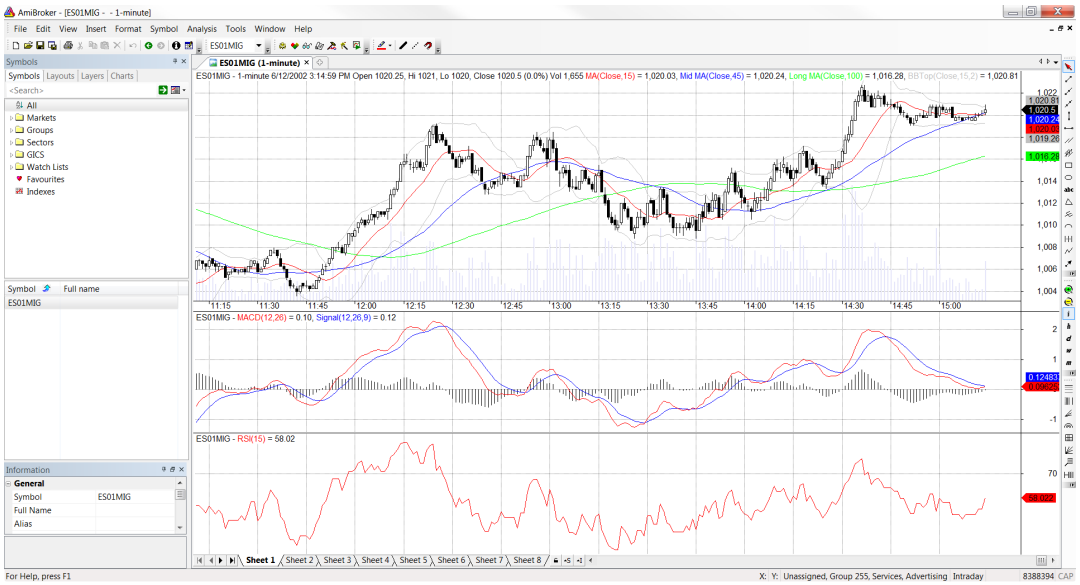
1. In AmiBroker, from the File menu, Open Database. Select the database you just created.
2. From the File menu, Import Wizard.
3. Click Pick Files.
4. Navigate to the location of your modified file and select it.
5. Click Open.
6. Click Next.

7. Fill in the fields.



8. Click Next.

9. Click Finish. The data will be imported and the active chart will display the intra-day data you just loaded.



10. In AmiBroker, Save Database.

If you were going to import data from this vendor regularly, you could set up a custom ASCII format in AmiBroker specifically for that file format.

But, eventually, you will need a real-time data feed to create 1 minute bars throughout the trading day.

Delayed Real-Time Data

Some data vendors offer real-time data that has been delayed 15, 20, or 30 minutes. It is priced lower than non-delayed real-time data, and often is free. From the perspective of a developer of trading systems, about the only reasons to use delayed real-time data are to try out a data service or to build up a tick-based database. And, since most of the high quality data services have a trial period, you can try the real thing for the same amount of effort. Trying to place trades using delayed real-time data is dangerous to your wealth — a lot can happen in 15 minutes.

Real-Time Data

Real-time data can be expensive. Prices range from about $50 to several hundred per month for the basic service which includes a limited number of tickers. There are additional charges for extra services, such as charting packages, Level II quotes, or getting data for a larger number of issues. Exchange fees, which are set by the exchange that clears trades for the tickers listed by them, are also added. There are different levels of exchange fees for professional and non-professional traders. Exchange fees can add up to several hundred dollars a month. Most real-time data vendors have trial periods, but the exchange fees charged during the trial are usually not refundable.

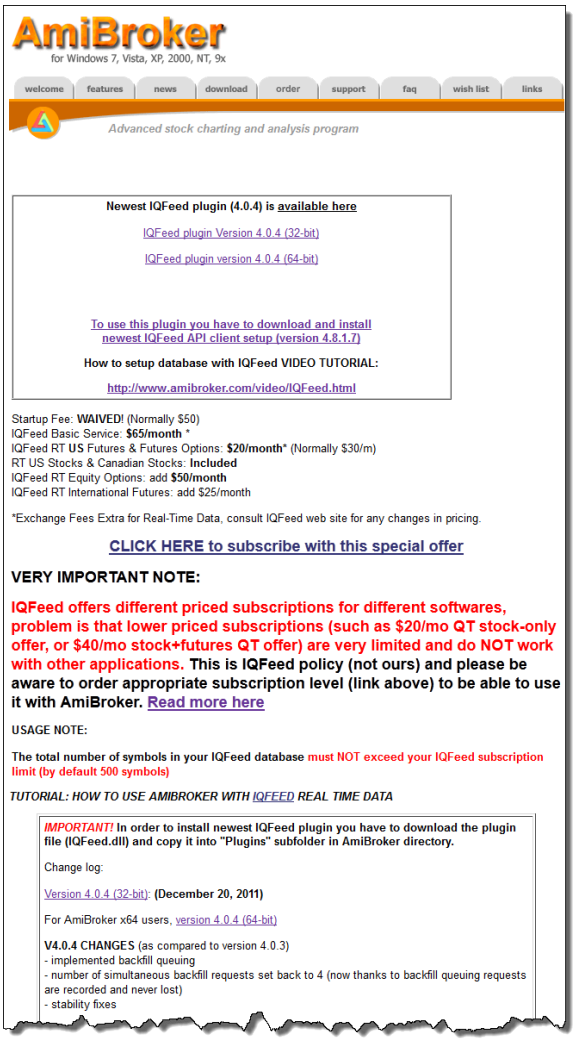
Also, check your Internet communications capacity and your communication vendor's fair use policy. Monitoring about ten symbols in the real-time quote window causes traffic of about 50 MB per hour. Done every trading day of the month, this will exceed the fair use limits of some vendors.

As mentioned earlier, some vendors support combining end-of-day and intra-day data in the same database. These steps describe separate intra-day databases.

DTN IQ Real-Time Data

There is an AmiBroker web page that describes use of IQFeed with AmiBroker: *www.amibroker.com/iqfeed.html*

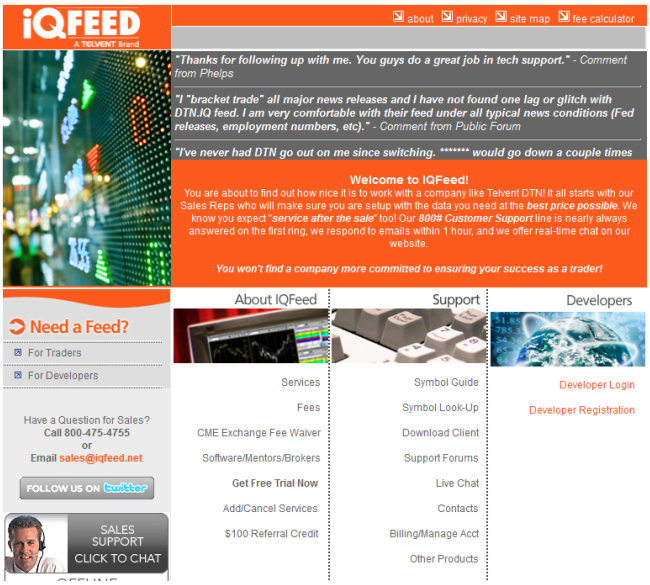
This is a long scrolling page. Only the top portion is captured here. Visit the site and read the page.



You will need the IQFeed plugin. There are links to the 32 bit and 64 bit versions from the page shown above. This plugin works in conjunction with the IQFeed API, which you get from the link on the next web page.

There is also a link to a video describing how to set up an AmiBroker database with IQFeed. The link to the video is on the page shown above. Watch the video. The procedure to set up a real-time database using IQFeed is:

1. Visit the DTN IQ Feed web site at: *www.iqfeed.net*.



2. Click Get Free Trial Now. Complete the application forms. The trial is exactly the same as the subscription version. The trial turns into a subscription at the end of the trial period, unless you call to cancel it. DTN will send an e-mail with your User ID and PIN.

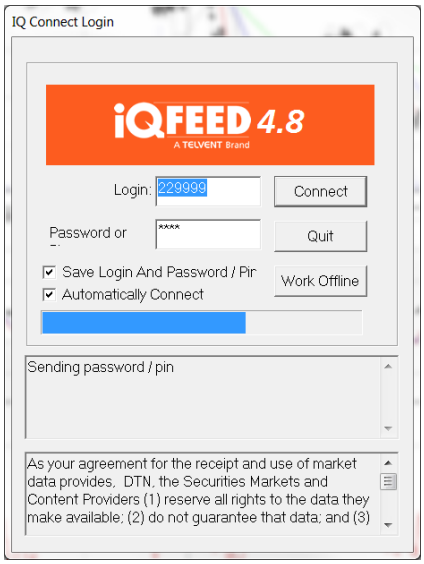
3. Note that there is a credit to a current DTN subscriber for referrals to new subscribers. If you are planning to subscribe to DTN IQ, I would appreciate receiving the referral credit. My DTN IQ user ID is howard@blueowlpress.com, account 229999. It does not cost you anything, and it will help support the effort of writing this book.

4. Download the IQFeed Client. It can be found at this website:
www.iqfeed.net/index.cfm?displayaction=support&section=download
At the time this is being written, the current version is 4.8.1.7.

5. Run the IQFeed Client program.

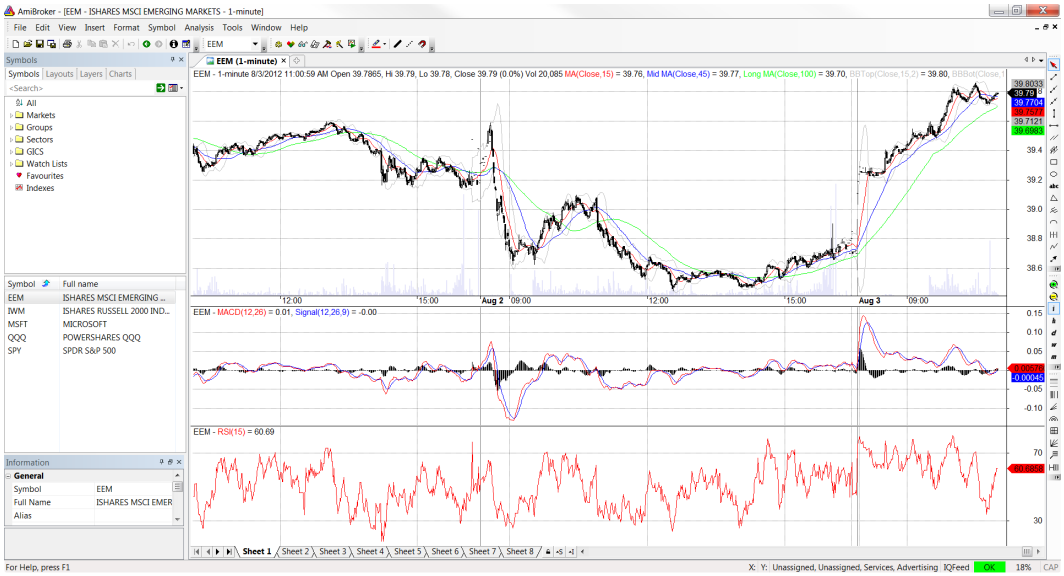
6. Download the IQFeed plugin file (IQFeed.dll) and copy it to the plugins subdirectory of the AmiBroker directory. If you recently installed AmiBroker, the IQFeed dll file may already be in place.

7. Set up the database.
   The procedure is the same as described above for the Free Intra-day Historical Data.
   - Use a meaningful name for the database, such as IQFeed.
   - The Data Source will be DTN IQFeed.
   - Number of bars should be 100000.
   - Base time interval 1 minute.
8. Click Configure. The configuration menu will appear.
9. Note that DTN IQ has a bad tick filter. Accept the defaults.
   Click OK.
   Return to the Database settings screen.
10. Click OK. The IQ Feed Connect Login will appear.
11. Enter your Login.
12. Enter your Password or PIN.
13. Click Connect. Your computer will communicate with the IQ Feed server.



To bring up a chart:

1. Using the AmiBroker Symbol menu, select New.
2. Type in a symbol, say SPY.
3. The historical prices will be backfilled, the chart will be displayed, and current activity will be added to the chart as ticks are received in real-time.



4. Add other symbols.
5. Open the Realtime Quote window.
6. Before you change to another database or close AmiBroker, remember to Save Database or Save All.

Tick Data

Tick data refers to market data that represents the time and transaction price, and perhaps the volume, of every trade. Depending on the exchange and the data vendor, the tick data stream may also include other information, such as changes in the bid price, the asked price, the volume on each side. Just as it is possible to make display bars that represent a given amount of time, say 1 minute, it is possible to display bars that represent a given number of ticks, say 15 ticks.

In the opinion of this author, tick data is not appropriate for use by individuals and small trading companies. The highest quality tick data includes an information packet for every transaction. Some of the difficulties with tick data include: high cost; very high volume of transactions; erroneous transactions (called bad ticks); dropping of transactions when exchange activity increases (fast market conditions) or when communications capacity is reached.

Tick data is best used by larger organizations that have high capacity equipment, redundant data feeds, and staff devoted to monitoring and correcting the data.

To use tick data in AmiBroker, the Professional version is required.

If you want to experiment with very frequent data, try working with 1 minute bars. If you are successful, then move on to tick data.

This book will not discuss tick data further.

Chapter 3

30 Minutes to Useful Results

You have installed AmiBroker and the Data database. This chapter describes ten things you can do with AmiBroker - even while it is in Trial Mode - each in only a few minutes.

Example 1 - Chart a Stock

Example 2 - Apply a Trendline

Example 3 - Plot a Moving Average

Example 4 - Make a Watchlist

Example 5 - Run an Exploration

Example 6 - Run a Single Stock Backtest

Example 7 - Run a Portfolio Backtest

Example 8 - Optimize a Trading System

Example 9 - Perform Walk Forward Validation

Example 10 - Scan for Buy and Sell Signals

None of these is intended to be a complete description of the features used in the example. But working through them should give you a better feeling for the capabilities and ease of use of AmiBroker, and provide some experience as you read about other features.

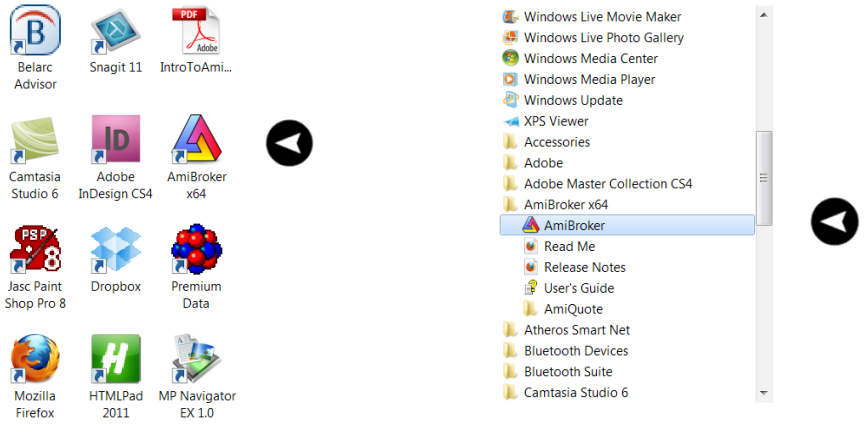
Examples build. Later examples assume you can perform tasks described in earlier examples.

Example 1 - Chart a Stock

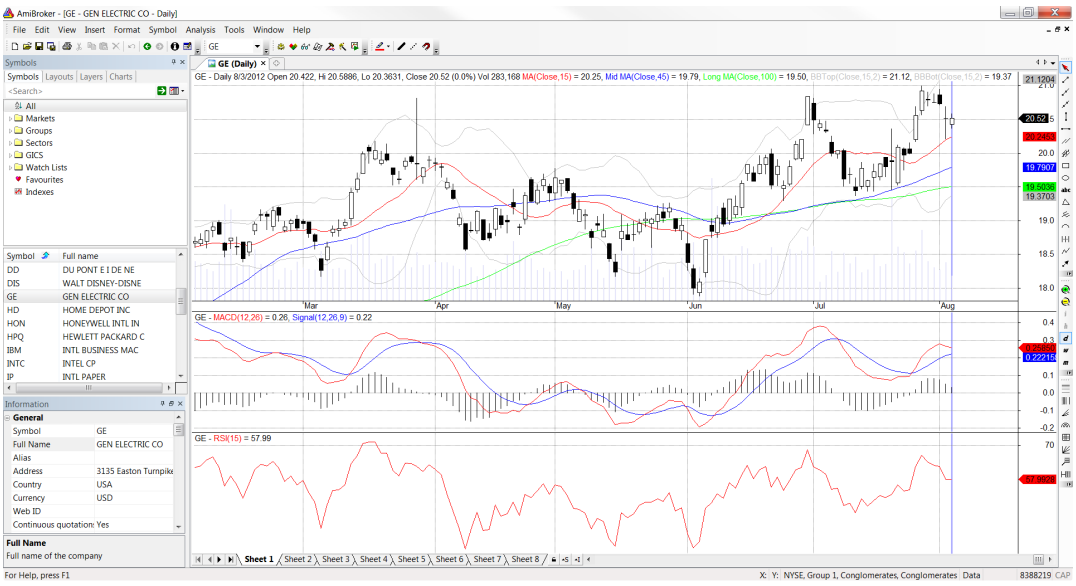
In this example, you will chart a stock, change its periodicity, chart a new stock, scroll, and zoom.

To start AmiBroker, either:

- Double-click the AmiBroker icon that was placed on your desktop during installation.
- Using the Windows Start menu, select AmiBroker, then AmiBroker.



AmiBroker will display the default chart window:



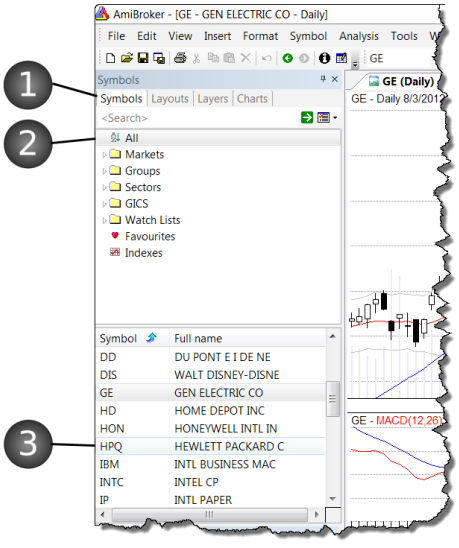
The chart window has three panes. The top pane has the price of the General Electric Corporation, GE, in candlestick format. Applied to those prices are several indicators - some moving averages and a set of Bollinger Bands. The middle pane has an MACD indicator. The bottom pane has an RSI indicator.

The prices are from the data that is installed in the default database named Data. The chart is displaying Daily data, one bar for each trading day.

To display the data in weekly format, click the Weekly view icon in the View toolbar.

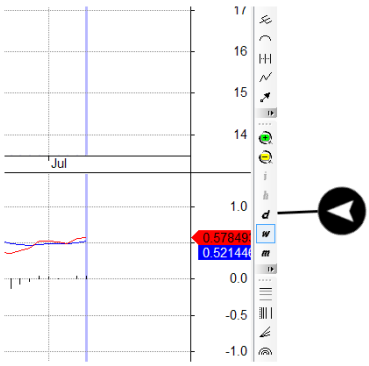
To choose a new symbol to plot:

1. Click the Symbols tabbed menu.
2. Click the All category. The Symbol dialog box will fill with members of that category.
3. Click one of the entries, say HPQ.

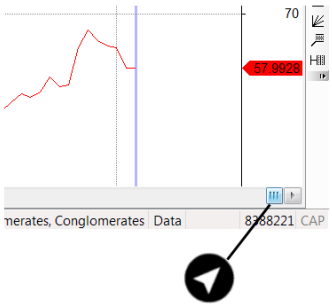


Since the charts were last displayed using weekly bars, they remain in weekly bars when the new symbol is plotted.

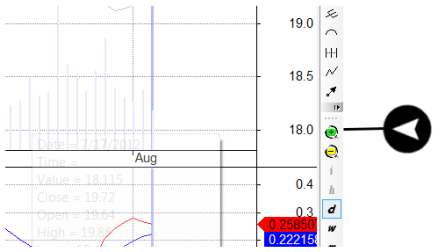
Return the chart to daily bars by clicking on the Daily view icon in View toolbar.



Using the Scroll Bar in the lower right corner of the window, scroll back in time (to the left) to display earlier data.

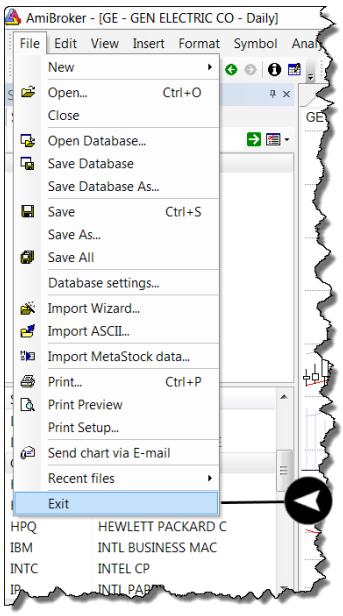


Using the Zoom In icon on the View toolbar, zoom in to see fewer bars, but in more detail.



Zoom back out using the Zoom Out icon on the View toolbar. (There is no graphic for this step.)

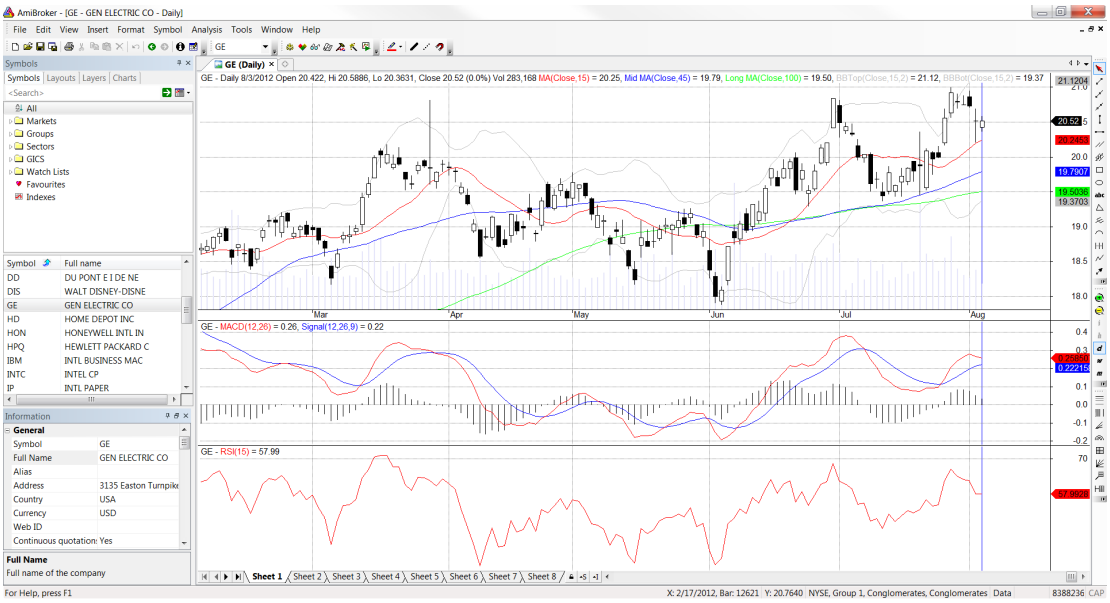
To exit AmiBroker, using the File pull-down menu, select Exit.



Example 2 - Apply a Trendline

In this example, you will apply a trendline, resize panes, and move a trendline.

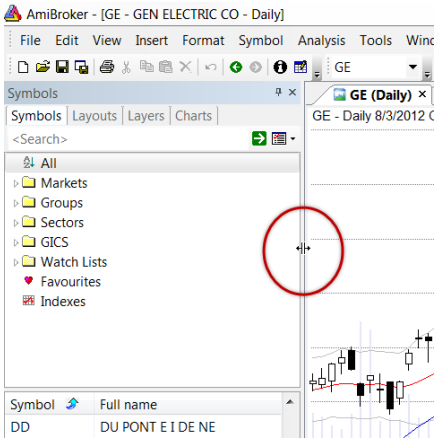
Start AmiBroker and plot the data for GE (General Electric) using daily bars.



Resize the panes to give the chart more real estate.

Move your mouse slowly across the vertical bar between the top pane and the Symbols window. Watch for the cursor to change from a single large arrow to two small arrows on either side of to vertical lines.

When you have the new cursor, hold the mouse button down, drag the vertical divider to the left, release the mouse.



Pick the points you want to use for your trendline. One technique is to connect two or more bottoms. For this example, we will use the low on June 5, 2012, and the low on July 23, 2012.

Use your mouse to select the Extended Trendline Tool from the Draw toolbar.

The cursor will change to be a small crosshairs. Position the cursor at the low on June 5, 2012. The ToolTip box will display the information about the bar your cursor is hovering over. When it is in the correct place, left-click your mouse.

Move your mouse, with no buttons held down, to the low on July 23. The ToolTip will tell you when you are there.

When you are satified with the position of your second point, left-click your mouse. The trendline will be drawn between the two points you selected and extended to both the right and left. You will see two small red squares at the points you used to define the trendline.

If you want to move one of the end points, hover your mouse over the red square until it becomes a two-headed arrow. Then left-click, drag the square to the position you want, and release the mouse button.

If you want to move the whole trendline, hover your mouse over the trendline between the two red squares until it becomes a four-headed arrow. Then left-click, drag the trendline to the position you want it, and release the mouse button. The trendline will maintain its angle and be parallel to its original position.

Scroll, zoom, change to weekly - do whatever you want to with the chart - the trendline will remain anchored at the two red squares and change with the chart.

If you have a registerd version of AmiBroker, the trendline will be saved when you exit. If you are unregistered, no changes are saved.

The techniques for placing and moving a trendline apply to most of the tools on the Draw toolbar.

Exit AmiBroker.

Example 3 - Plot a Moving Average

In this example, you will insert a new pane with a new price series, overlay it with a moving average, and modify the moving average.

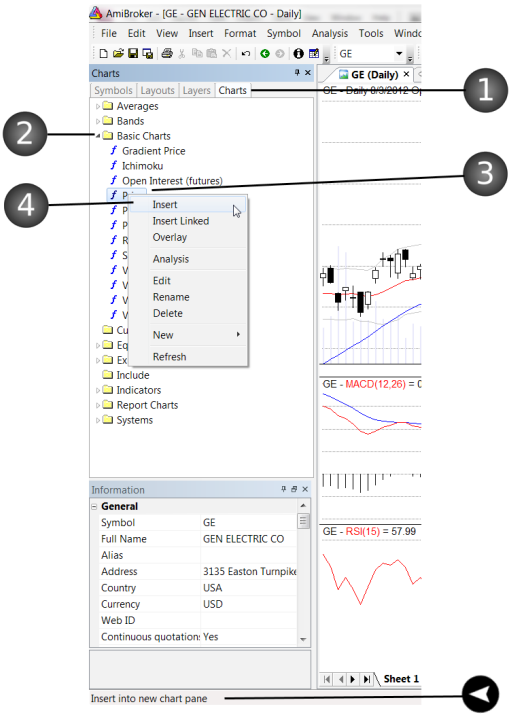
Start AmiBroker. You will see the normal chart window with three panes.

If necessary, resize the panes so that you can see the Charts tabbed menu.

Follow these steps to insert a price series in a new pane into the window:

1. Click the Charts tab.
2. Expand the Basic Charts category by clicking the triangle.
3. Right-click Price, which brings up the context menu.
4. Left-click Insert.

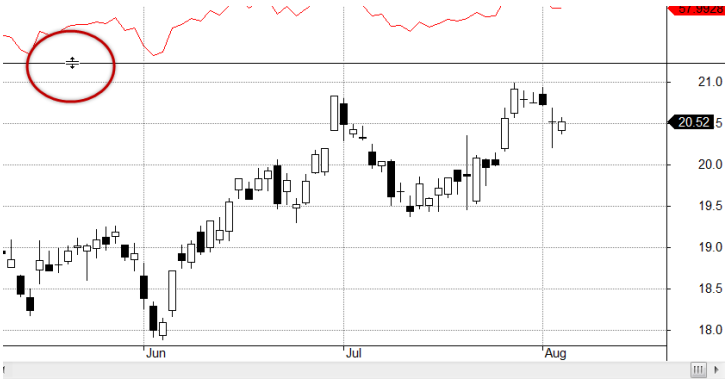
Note the tip at the bottom of the window that tells you what this command will do.



A new pane is opened as the bottom pane and the price series is plotted in candlestick format.

A properties dialog box is also opened. Click OK to close it.

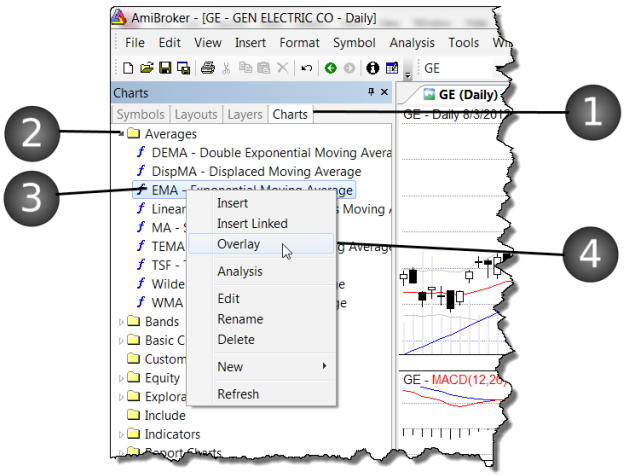
Move your mouse slowly over the divider between the new pane and the one above it until the cursor changes to two parallel lines with two small arrows. Use this cursor to resize the new pane to make it bigger.



Left-click in the new pane to be certain it is active (or selected, or has the focus).

Follow these steps to apply an Exponential Moving Average to the Price series:

1. Click the Charts tab.
2. Expand the Averages category.
3. Right-click EMA - Exponential Moving Average, which brings up the context menu.
4. Left-click Overlay.

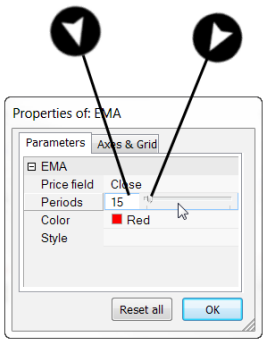


A properties dialog box is opened.

If you are satisfied with the parameter values, Click OK. If you want to change one or more of them, follow these steps.

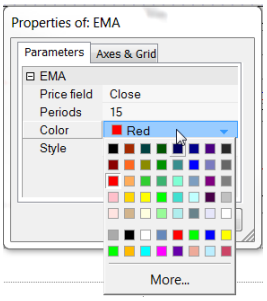
To change the length of the moving average, click in the area to the right of the Periods field. Then either:

- Type a numeric value for the length.
- Use the slider to adjust the length.

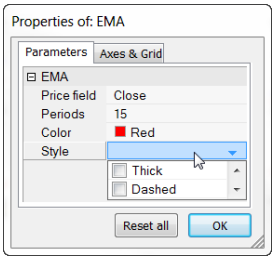


As you change the length of the moving average, the line that plots the moving average changes immediately to reflect the new length parameter. Move the slider and watch the red line. As the length is shorter, the moving average responds quickly and hugs the prices more closely. As the length increases, the moving average is smoother, but lags behind when the price series changes.

To change the color of the line used to plot the moving average, click in the area to the right of the Color field. A color chart will appear and you can choose the color you prefer.



To change the type of line, click in the area to the right of the Style field. A pull-down menu will give you choices such as dashed, dotted, and thick.



The Axes and Grid Tab also has some parameters that can be changed, including show or hide dates, linear or log scale.

If you have a registered version of AmiBroker, the window, including the new pane with the price series and moving average, will be saved when you exit. If you are unregistered, no changes are saved.

The techniques used to modify the moving average are also applicable to many of the indicators on the Charts tabbed menu.

Exit AmiBroker.

Example 4 - Make a Watchlist

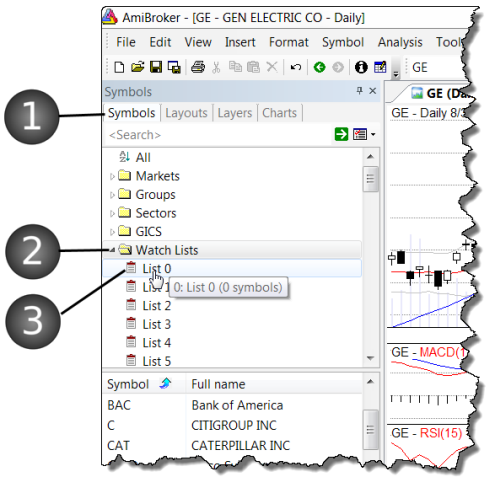
In this example, you will make a watchlist, modify it, and use it to display the charts of each member.

Start AmiBroker.

Follow these steps to learn about Watchlist Number 0:

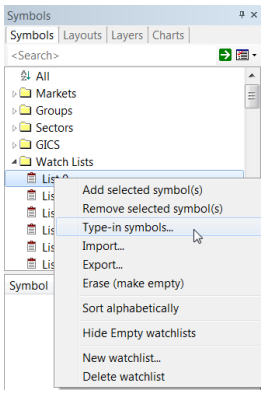
1. Click on the Symbols tabbed menu.
2. Expand the Watchlist category.
3. Hover your mouse over List 0.

A message informs you that there are 0 symbols in List 0.

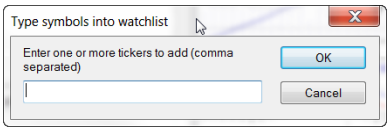


Follow these steps to add three ticker symbols to Watchlist 0:

1. Right-click List 0. The context menu will open.
2. Left-click Type-in symbols.

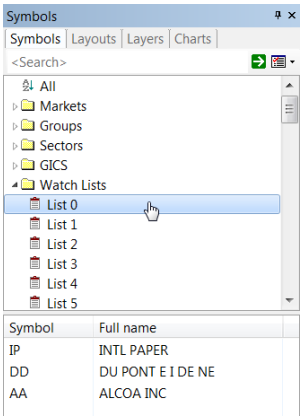


A dialog box will open, inviting you to Type symbols into the watchlist.



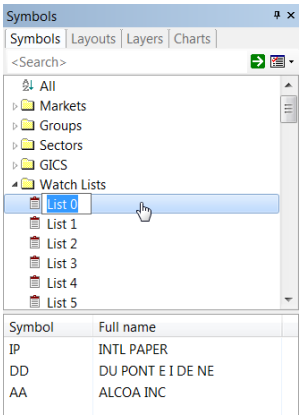
1. Type IP, DD, AA. Use all capital letters and separate the symbols with commas.
2. Click OK.

Click on List 0 and note, in the Symbol section, that it now has three members.



Rename Watchlist 0 to something more meaningful, say Basic Materials:

1. Click the name, List 0.
2. Wait a few seconds.
3. Click List 0 again. The name field will be selected.
4. Type the new name, Basic Materials.
5. Press Enter. The watchlist has been renamed.



Add a symbol to the watchlist:

1. Right-click the name of the watchlist, Basic Materials. A context menu opens.
2. Left-click Type in symbols.. A dialog box opens.
3. Type in the additional symbol(s), say XOM. The watchlist has four members.

To use the watchlist to rapidly review the charts of its members:

1. Left-click the first member of the list, IP. The chart for International Paper will be displayed.
2. Press the keyboard cursor-down-arrow key. The chart of the next member, Du Pont, will be displayed.
3. Continue through the watchlist.

Exit AmiBroker.

Exercise 5 - Run an Exploration

In this example, you will use the Formula Editor, and write and run an Exploration. For this exploration, the AFL program checks a watchlist to see how many stocks are at new 10 day highs or new 10 day lows. There is also a short introduction to Analysis.

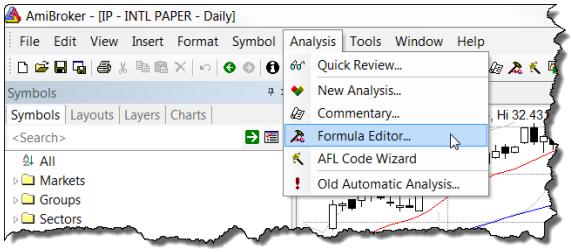
Start AmiBroker.

Formula Editor

First, use the Formula Editor to write and save an AFL program.

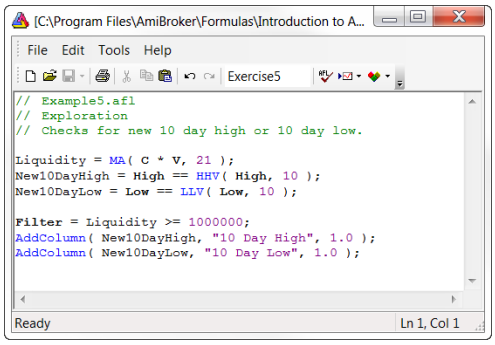
Open the Formula Editor:

1. Click the Analysis pull-down menu.
2. Click Formula Editor. The Editor window will open.



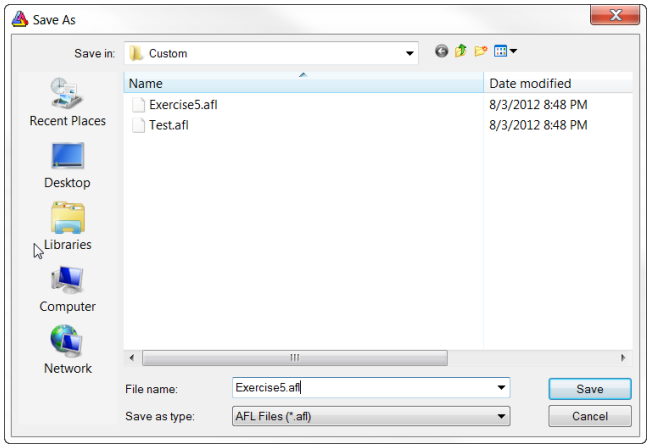
3. Expand the Editor window enough to give yourself room to work.
4. Type the AFL code into the window so that it matches the example below.

```
// Example5.afl
// Exploration
// Checks for new 10 day high or 10 day low.

Liquidity = MA( C * V, 21 );
New10DayHigh = High == HHV( High, 10 );
New10DayLow = Low == LLV( Low, 10 );

Filter = Liquidity >= 1000000;
AddColumn( New10DayHigh, “10 Day High”, 1.0 );
AddColumn( New10DayLow, “10 Day Low”, 1.0 );
```



5. Using the editor's pull-down menu, select Save As.
6. The Save In directory should already be Custom. If it is not, navigate to: C:\Program Files\AmiBroker\Formulas\Custom.
7. For the File Name, type Example5.afl.

Click Save. (This works even if you are using an unregistered version of AmiBroker.)



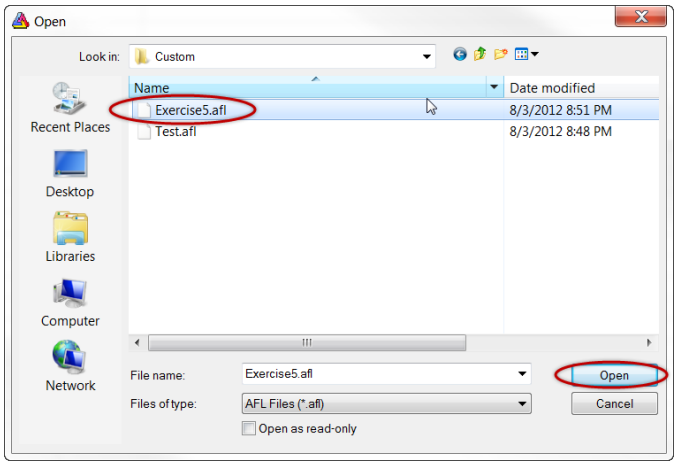
Any time you want to modify or edit this program:

1. Use the Formula Editor's File menu, select Open.
2. Choose Example5.afl.
3. Click Open.

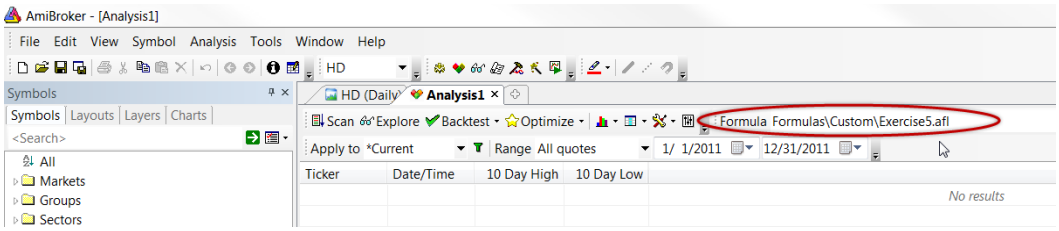
Exploration

Next, use Example5.afl as an Exploration.

1. Using the Analysis tab, Click the Pick a File icon. A dialog box opens, showing afl programs in the Custom directory (or whichever directory was most recently used).
2. Select Example5.afl
3. Click Open.



4. As you can tell from the Formula File box, Example5.afl has been loaded in the Analysis module.



Analysis is the module from which you will run the formula-based analysis techniques. These are the quantitative techniques, as compared with the visual and graphical techniques of applying trendlines and looking for chart patterns. There are four major components of Analysis:

- Explore - used to search for conditions that are interesting, and produce reports with columns of data. For example, search for all instances of the stochastic oscil-

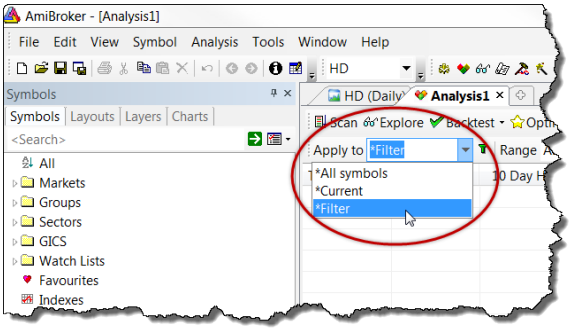
lator being below 20 for two days. Compute and report the price change over the next three days.

- Backtest - used to test a trading system using historical data. Backtests can be performed on individual stocks or on portfolios of stocks. You have complete control over the account size, how many positions are held at any time, how much to invest in each position, when to buy, and when to sell. The output is a report, a plot showing arrows at the the buy and sell points, and a plot of the account equity.
- Optimize - used to select the best values for the parameters in the trading system. In a moving average crossover trading system, optimization could be used to select the best values for the lengths of the moving averages.
- Scan - after the system has been designed and is ready to be traded, scan through the symbols that are in the universe of possible holdings to see if there are new buy or sell signals.

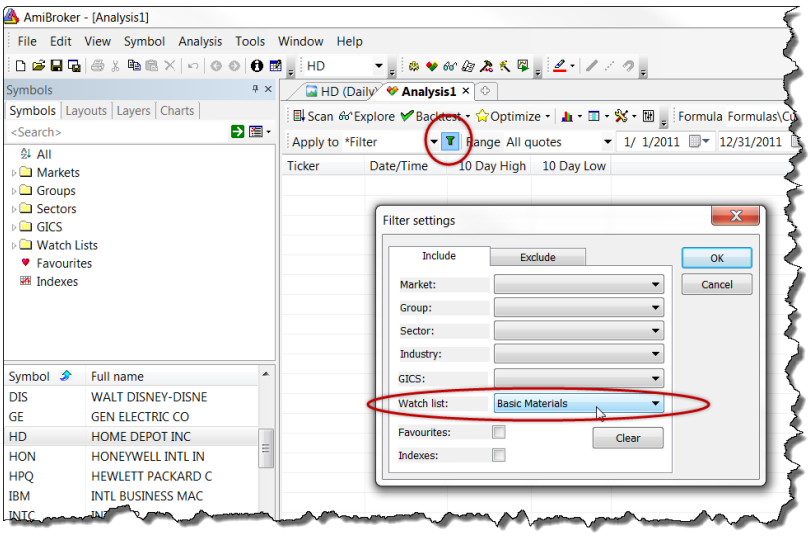
Depending on what instructions you have written in your afl program, you may be able to run exactly the same program in all of the Analysis components. Chapter 9, Analysis, goes into each of these in more detail.

For now, fill in just what is necessary to complete the Exploration.

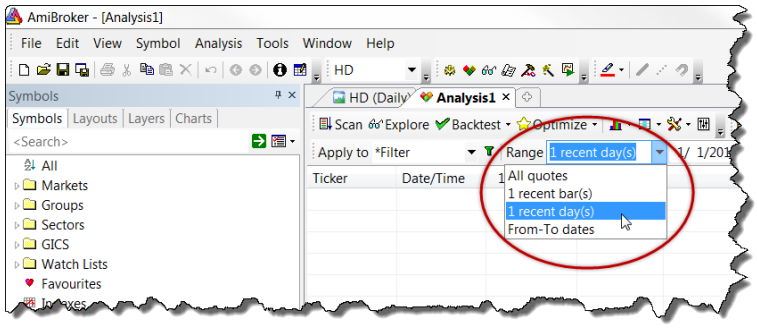
In the Apply to area, use the drop-down menu and select Filter.



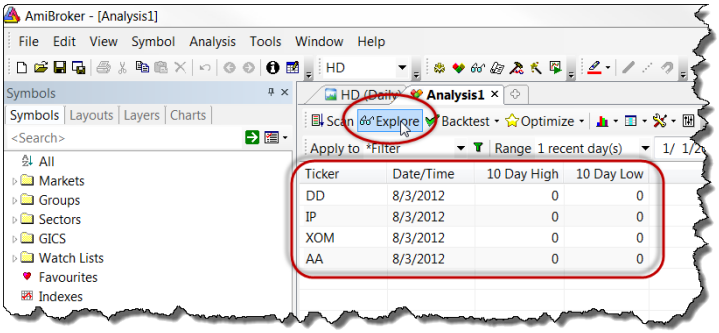
Click the Define Filter icon. The Filter Settings dialog box opens. Use the drop-down menu to select the watchlist named Basic Materials. Click OK.



In the Range area, select 1 Recent Days.



Click Explore. A report will be displayed. There is one line for each ticker for each day. A value of 1, which means True, indicates that issue had a 10 day High or 10 day Low on that day. A value of 0, which means False, indicates that it did not.



None of the issues had either a 10 day high or 10 day low when this example was run.

On your own -- Change the Range to From-To Dates, enter a date range using the calendar icons, and rerun the exploration to see if there were 10 day highs or lows in the period you chose.

Exit AmiBroker.

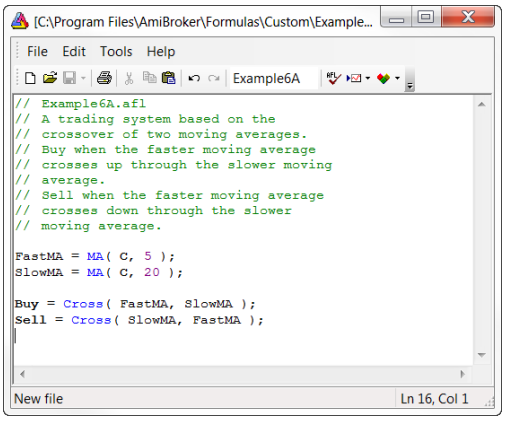
Example 6 - Run a Single Stock Backtest

In this example, you will use the Formula Editor to write the AFL code for a simple trading system, run the code as a Backtest on the historical data of one stock, review the results, plot the Buy and Sell arrows, plot the Equity Curve.

Start AmiBroker.

Using the Analysis pull-down menu, select the Formula Editor. The Formula Editor will open, ready to accept your new AFL program. Type it in so it matches this one:

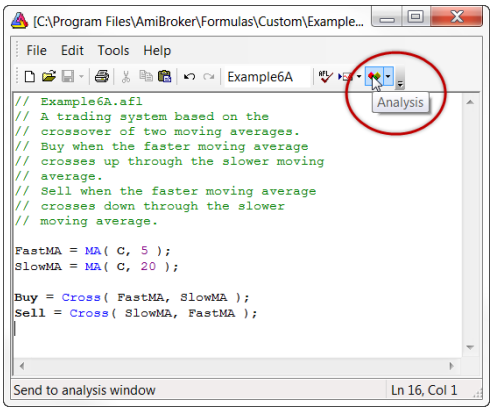
```
// Example6A.afl
// A trading system based on the
// crossover of two moving averages.
// Buy when the faster moving average
// crosses up through the slower moving
// average.
// Sell when the faster moving average
// crosses down through the slower
// moving average.

FastMA = MA( C, 5 );
SlowMA = MA( C, 20 );

Buy = Cross( FastMA, SlowMA );
Sell = Cross( SlowMA, FastMA );
```



Save it in the Custom directory as Example6A.afl.

In setting up Analysis, you need to load the file you want processed. Example 5 showed one way to do that. Here is another - a shortcut that coordinates Formula Editor with Analysis.

Formula Editor has a toolbar. The rightmost icon in the toolbar is three squares in the shape of a V. If you hover your mouse over it, the tooltip says Analysis. When you click that icon, the file that is open in Formula Editor is loaded into Analysis. This makes for an easy cycle of editing and analyzing. After the edit, click the Analysis icon and go directly on with the analysis.



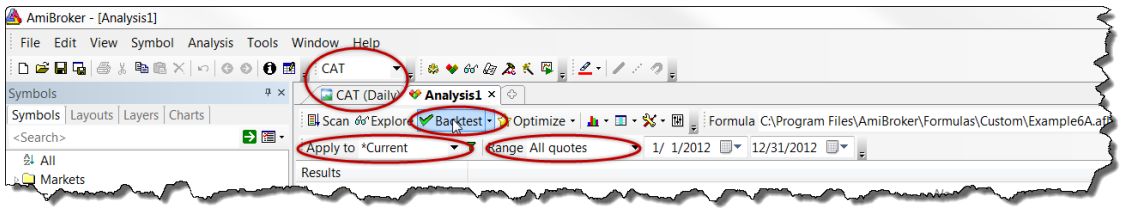
Backtest

In the Formula file area, load Example6A.afl.

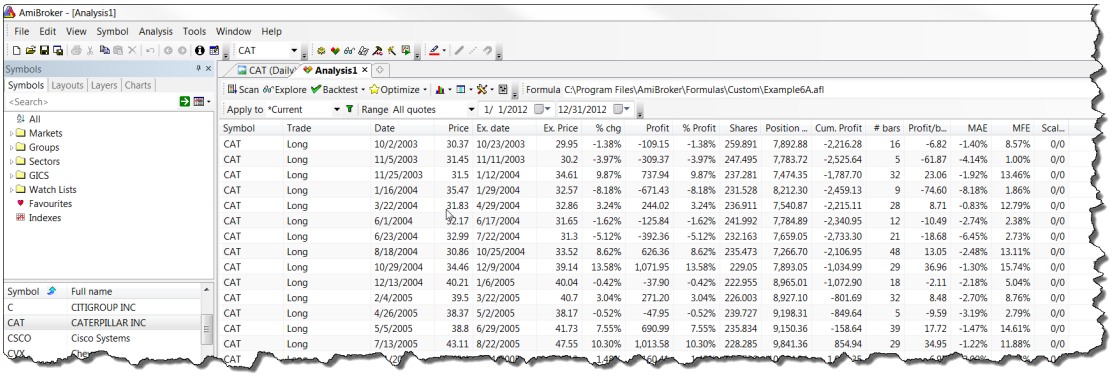
Make CAT, Caterpillar Corporation, the active symbol and display it in the chart window with daily bars.

To run the backtest:

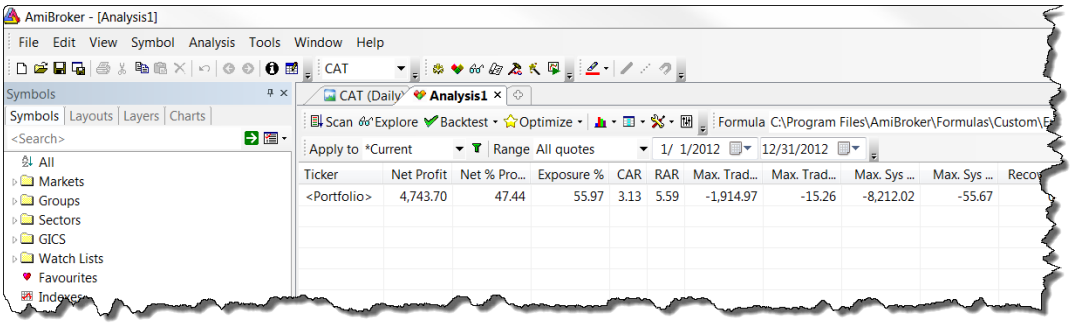
1. In the Apply to area, select Current symbol.
2. In the Range area, select All quotes.
3. Click Backtest (ignore the pull-down menu for now).



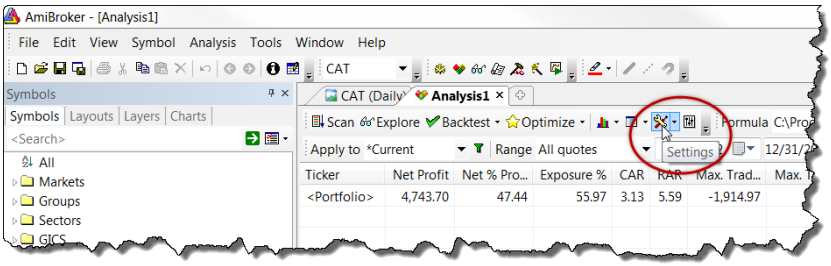
The trade-by-trade results appear in the Results List on the Analysis tab.



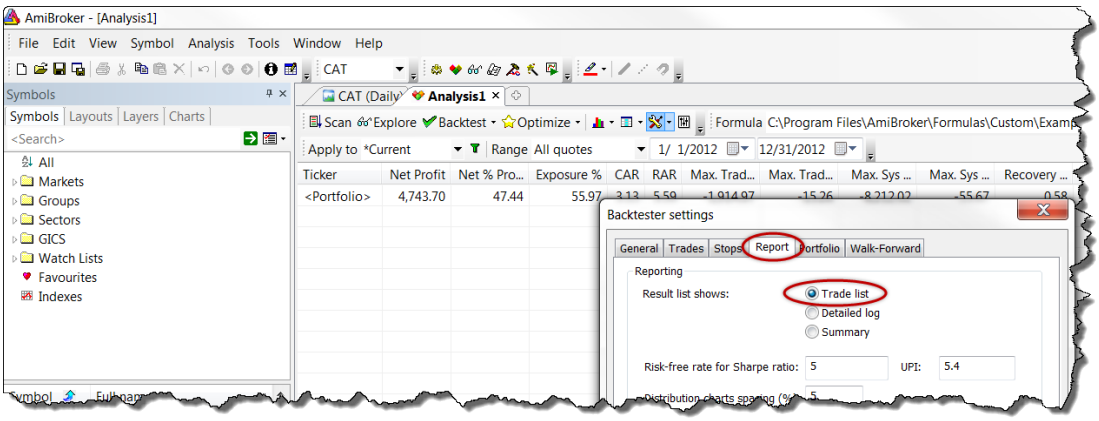
If, instead of the trade list, you see only a single line summary:



you can control that using the Settings. Click the Settings icon. It looks like a little wrench.



A Backtester settings dialog box will open. Select the Report tab. Select the Trade List radio button. Click OK. Click Backtest again.

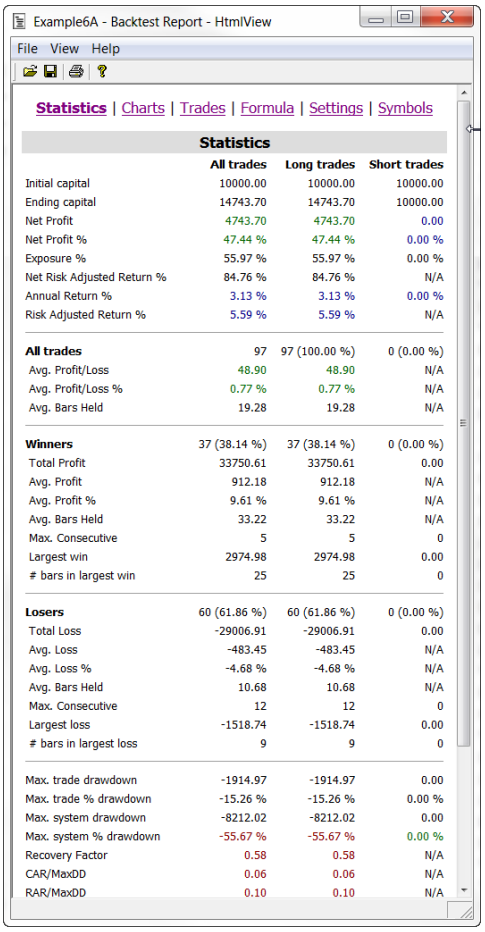


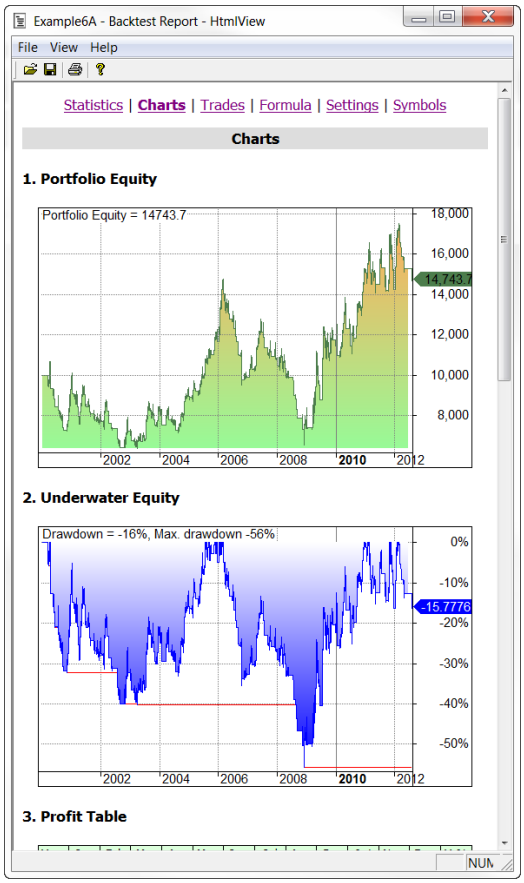
Results

In addition to the trade-by-trade results, there are three ways you might like to review the results - the report, buy and sell arrows, and equity curve. Each will be discussed in turn.

Report

Click the Report icon that looks like a small spreadsheet. A report window will open. Click the Maximize button to make it full screen, then examine each of the six report sections - Statistics, Charts, Trades, Formula, Settings, Symbols. The statistics and charts show that this system is awful.

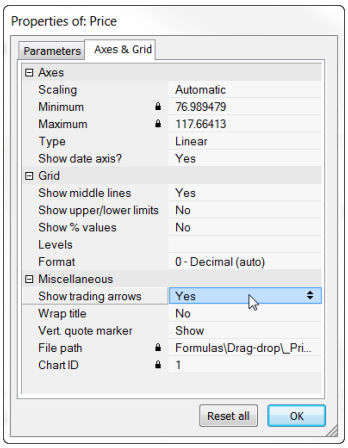




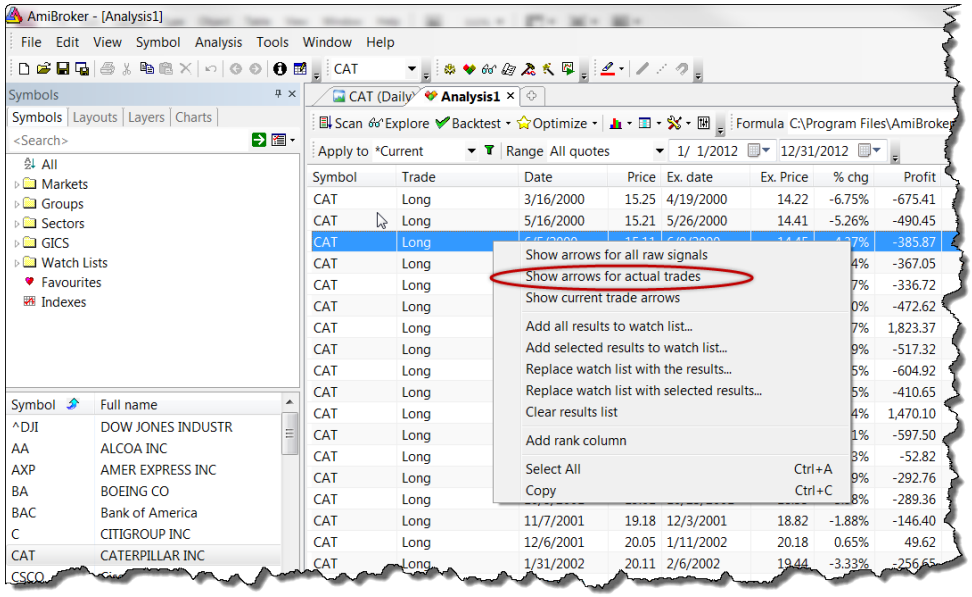
Buy and Sell Arrows

Click the Chart tab. Right-click anywhere in the top pane where the price series is plotted. A context menu will open. Left-click Parameters.

A Properties dialog box will open. Click the Axes & Grid tab. Scroll down until you find Show trading arrows. Click in the area to the right until it is set to Yes. Click OK.



Click the Analysis tab. Right click any single line in the trade list. A dialog box will open. Click Show arrows for actual trades.

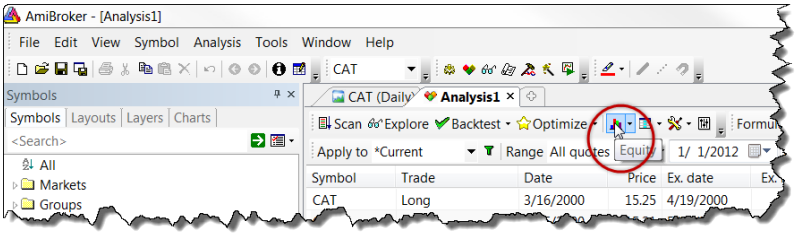


Click the Chart tab. You will see green upward-pointing arrows at the bars where Buy signals occurred and red downward-pointing arrows at the bars where Sell signals occurred.

Equity Curve

Method 1

On the Analysis tab, click the Equity icon that looks like a small bar chart.



Click the Chart tab. A new pane has been created as the bottom pane. It contains a bar chart of equity. The green represents cash; blue holding a position; brown drawdown.

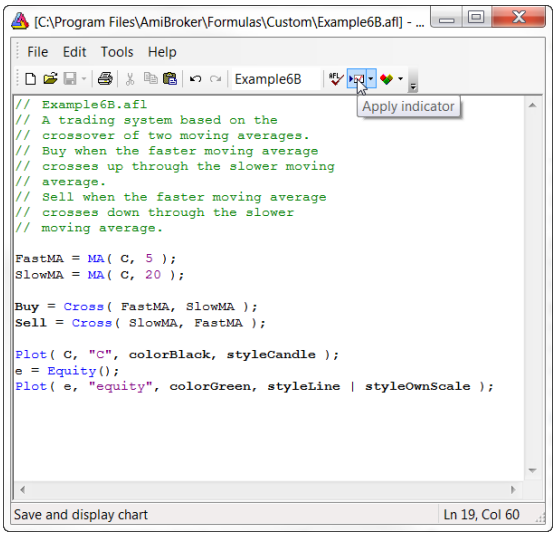
Method 2

Click the Chart tab.

In Formula Editor, open Example6A.afl.

Add the three lines at the bottom to make it like the code shown here.

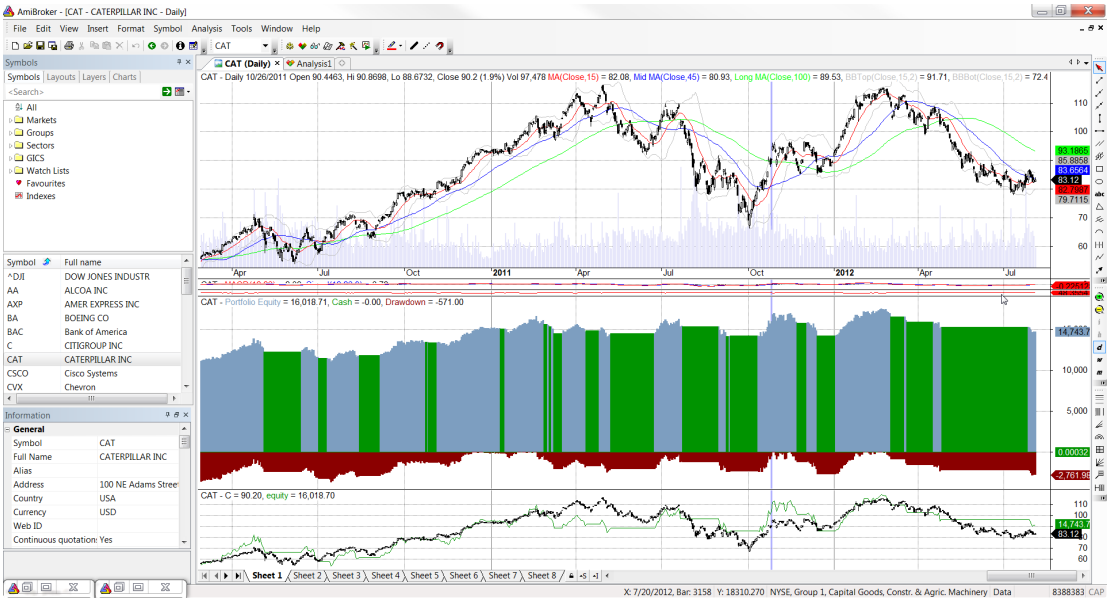
```
// Example6B.afl
// A trading system based on the
// crossover of two moving averages.
// Buy when the faster moving average
// crosses up through the slower moving
// average.
// Sell when the faster moving average
// crosses down through the slower
// moving average.

FastMA = MA( C, 5 );
SlowMA = MA( C, 20 );

Buy = Cross( FastMA, SlowMA );
Sell = Cross( SlowMA, FastMA );

Plot( C, “C”, colorBlack, styleCandle );
e = Equity();
Plot( e, “equity”, colorGreen, styleLine | styleOwnScale );
```



Save it as Example6B.afl

Click the Apply indicator icon in the toolbar.

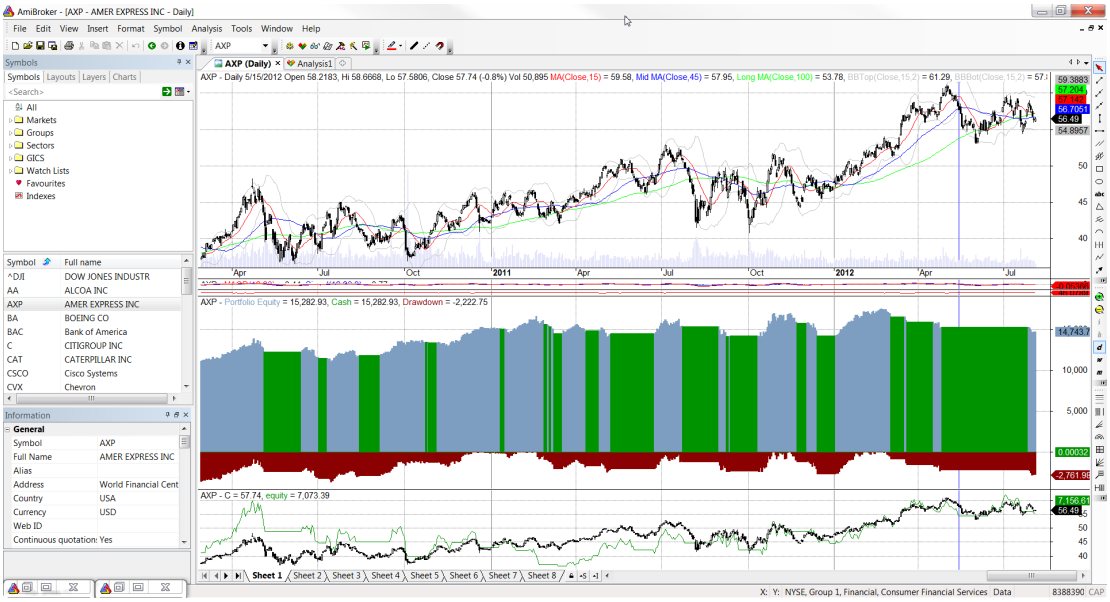
A new pane will appear in the chart window. It has the price series displayed as black candlesticks and the equity curve displayed as a green line. These came from the two Plot statements in the revised program. Note that the changes in the equity as plotted in Method 1 and the changes in equity as plotted using this method agree. Click the cursor at any point in any pane and the value of the equity at that bar will be displayed. The final equity is in the green field at the right side of the chart -- $14,743 for CAT for the period tested. Note that this figure agrees with the statistics and chart produced by the Report.



There is an added benefit to using the second method of displaying equity. Using the Symbols menu, click on any symbol, say AXP. All of the charts, except the green and blue equity chart, change to reflect American Express, including the equity resulting from applying the trading system of Example6B, showing a final equity of $7,156.

The green and blue chart did not change -- it was plotted from the analysis tab. In order to refresh it, click the Analysis tab, click Backtest. The green and blue equity chart will be refreshed to reflect applying the system to AXP.

Visual inspection of the equity curve is an excellent way to get a feeling for the behaviour of a trading system. Cycling through a list of issues by pressing the down-arrow key shows the equity curve for each without needing to explicitly run a backtest.



Save all.

For a more detailed discussion of the use of backtesting and the reliability of in-sample results, read my paper, *Backtesting and Cognitive Dissonance*.

Exit AmiBroker.

Example 7 - Run a Portfolio Backtest

This example will apply a trading system to a portfolio and view the portfolio equity.

Start AmiBroker.

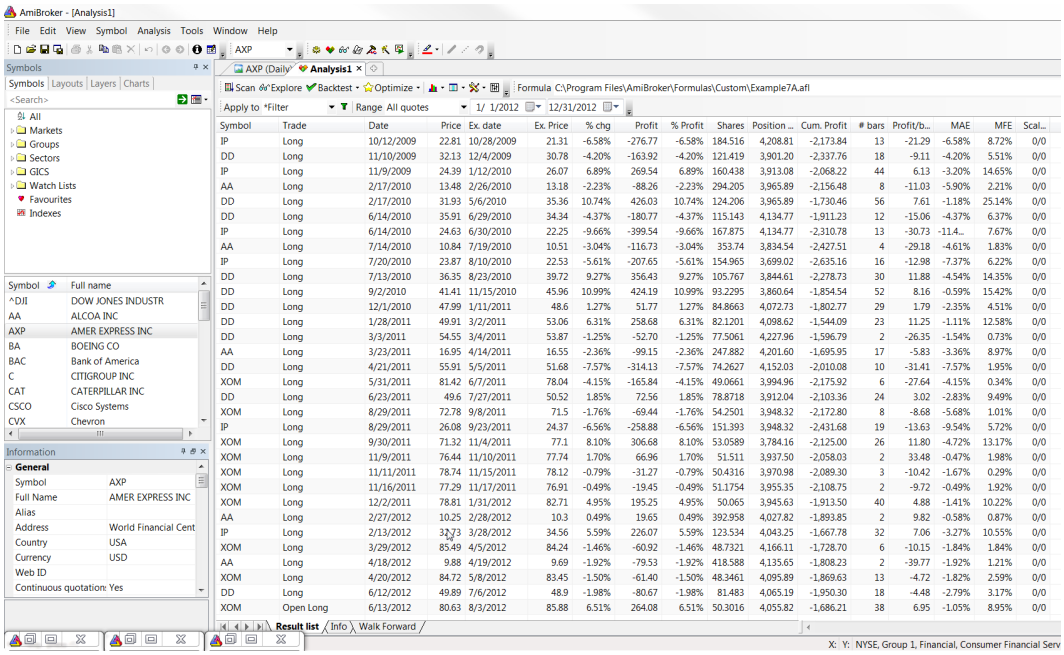
Using Formula Editor:

1. Open Example6A.afl.
2. Change it to make it agree with the program below.

```
// Example7A.afl
// A trading system based on the
// crossover of two moving averages.
// Buy when the faster moving average
// crosses up through the slower moving
// average.
// Sell when the faster moving average
// crosses down through the slower
// moving average.
// This version adds multiple positions
// for use with a portfolio.

SetOption( “MaxOpenPositions”, 2 );
SetPositionSize( 50, spsPercentOfEquity );

FastMA = MA( C, 5 );
SlowMA = MA( C, 20 );

Buy = Cross( FastMA, SlowMA );
Sell = Cross( SlowMA, FastMA );
```

3. Save the new program as Example7A.afl.
4. Click the Analysis icon. Example7A is loaded into Analysis as the Formula File.
5. Click the Analysis tab.
6. Apply to: Filter.
7. Define filter: Watchlist > Basic Materials.
8. Range: All quotes.
9. Settings > Report tab > Trade List.

10. Click Backtest. The Results List shows the individual trades.



Symbol	Trade	Date	Price	Ex. date	Ex. Price	% chg	Profit	% Profit	Shares	Position ...	Cum. Profit	# bars	Profit/b...	MAE	MFE	Scal...
IP	Long	10/12/2009	22.81	10/28/2009	21.31	-6.58%	-276.77	-6.58%	184.516	4,208.81	-2,173.84	13	-21.29	-6.58%	8.72%	0/0
DD	Long	11/10/2009	32.13	12/4/2009	30.78	-4.20%	-163.92	-4.20%	121.419	3,901.20	-2,337.76	18	-9.11	-4.20%	5.51%	0/0
IP	Long	11/9/2009	24.39	1/12/2010	26.07	6.89%	269.54	6.89%	160.438	3,913.08	-2,068.22	44	6.13	-3.20%	14.65%	0/0
AA	Long	2/17/2010	13.48	2/26/2010	13.18	-2.23%	-88.26	-2.23%	294.205	3,965.89	-2,156.48	8	-11.03	-5.90%	2.21%	0/0
DD	Long	2/17/2010	31.93	5/6/2010	35.36	10.74%	426.03	10.74%	124.206	3,965.89	-1,730.46	56	7.61	-1.18%	25.14%	0/0
DD	Long	6/14/2010	35.91	6/29/2010	34.34	-4.37%	-180.77	-4.37%	115.143	4,134.77	-1,911.23	12	-15.06	-4.37%	6.37%	0/0
IP	Long	6/14/2010	24.63	6/30/2010	22.25	-9.66%	-399.54	-9.66%	167.875	4,134.77	-2,310.78	13	-30.73	-11.4...	7.67%	0/0
AA	Long	7/14/2010	10.84	7/19/2010	10.51	-3.04%	-116.73	-3.04%	353.74	3,834.54	-2,427.51	4	-29.18	-4.61%	1.83%	0/0
IP	Long	7/20/2010	23.87	8/10/2010	22.53	-5.61%	-207.65	-5.61%	154.965	3,699.02	-2,635.16	16	-12.98	-7.37%	6.22%	0/0
DD	Long	7/13/2010	36.35	8/23/2010	39.72	9.27%	356.43	9.27%	105.767	3,844.61	-2,278.73	30	11.88	-4.54%	14.35%	0/0
DD	Long	9/2/2010	41.41	11/15/2010	45.96	10.99%	424.19	10.99%	93.2295	3,860.64	-1,854.54	52	8.16	-0.59%	15.42%	0/0
DD	Long	12/1/2010	47.99	1/11/2011	48.6	1.27%	51.77	1.27%	84.8663	4,072.73	-1,802.77	29	1.79	-2.35%	4.51%	0/0
DD	Long	1/28/2011	49.91	3/2/2011	53.06	6.31%	258.68	6.31%	82.1201	4,098.62	-1,544.09	23	11.25	-1.11%	12.58%	0/0
DD	Long	3/3/2011	54.55	3/4/2011	53.87	-1.25%	-52.70	-1.25%	77.5061	4,227.96	-1,596.79	2	-26.35	-1.54%	0.73%	0/0
AA	Long	3/23/2011	16.95	4/14/2011	16.55	-2.36%	-99.15	-2.36%	247.882	4,201.60	-1,695.95	17	-5.83	-3.36%	8.97%	0/0
DD	Long	4/21/2011	55.91	5/5/2011	51.68	-7.57%	-314.13	-7.57%	74.2627	4,152.03	-2,010.08	10	-31.41	-7.57%	1.95%	0/0
XOM	Long	5/31/2011	81.42	6/7/2011	78.04	-4.15%	-165.84	-4.15%	49.0661	3,994.96	-2,175.92	6	-27.64	-4.15%	0.34%	0/0
DD	Long	6/23/2011	49.6	7/27/2011	50.52	1.85%	72.56	1.85%	78.8718	3,912.04	-2,103.36	24	3.02	-2.83%	9.49%	0/0
XOM	Long	8/29/2011	72.78	9/8/2011	71.5	-1.76%	-69.44	-1.76%	54.2501	3,948.32	-2,172.80	8	-8.68	-5.68%	1.01%	0/0
IP	Long	8/29/2011	26.08	9/23/2011	24.37	-6.56%	-258.88	-6.56%	151.393	3,948.32	-2,431.68	19	-13.63	-9.54%	5.72%	0/0
XOM	Long	9/30/2011	71.32	11/4/2011	77.1	8.10%	306.68	8.10%	53.0589	3,784.16	-2,125.00	26	11.80	-4.72%	13.17%	0/0
XOM	Long	11/9/2011	76.44	11/10/2011	77.74	1.70%	66.96	1.70%	51.511	3,937.50	-2,058.03	2	33.48	-0.47%	1.98%	0/0
XOM	Long	11/11/2011	78.74	11/15/2011	78.12	-0.79%	-31.27	-0.79%	50.4316	3,970.98	-2,089.30	3	-10.42	-1.67%	0.29%	0/0
XOM	Long	11/16/2011	77.29	11/17/2011	76.91	-0.49%	-19.45	-0.49%	51.1754	3,955.35	-2,108.75	2	-9.72	-0.49%	1.92%	0/0
XOM	Long	12/2/2011	78.81	1/31/2012	82.71	4.95%	195.25	4.95%	50.065	3,945.63	-1,913.50	40	4.88	-1.41%	10.22%	0/0
AA	Long	2/27/2012	10.25	2/28/2012	10.3	0.49%	19.65	0.49%	392.958	4,027.82	-1,893.85	2	9.82	-0.58%	0.87%	0/0
IP	Long	2/13/2012	32.73	3/28/2012	34.56	5.59%	226.07	5.59%	123.534	4,043.25	-1,667.78	32	7.06	-3.27%	10.55%	0/0
XOM	Long	3/29/2012	85.49	4/5/2012	84.24	-1.46%	-60.92	-1.46%	48.7321	4,166.11	-1,728.70	6	-10.15	-1.84%	1.84%	0/0
AA	Long	4/18/2012	9.88	4/19/2012	9.69	-1.92%	-79.53	-1.92%	418.588	4,135.65	-1,808.23	2	-39.77	-1.92%	1.21%	0/0
XOM	Long	4/20/2012	84.72	5/8/2012	83.45	-1.50%	-61.40	-1.50%	48.3461	4,095.89	-1,869.63	13	-4.72	-1.82%	2.59%	0/0
DD	Long	6/12/2012	49.89	7/6/2012	48.9	-1.98%	-80.67	-1.98%	81.483	4,065.19	-1,950.30	18	-4.48	-2.79%	3.17%	0/0
XOM	Open Long	6/13/2012	80.63	8/3/2012	85.88	6.51%	264.08	6.51%	50.3016	4,055.82	-1,686.21	38	6.95	-1.05%	8.95%	0/0

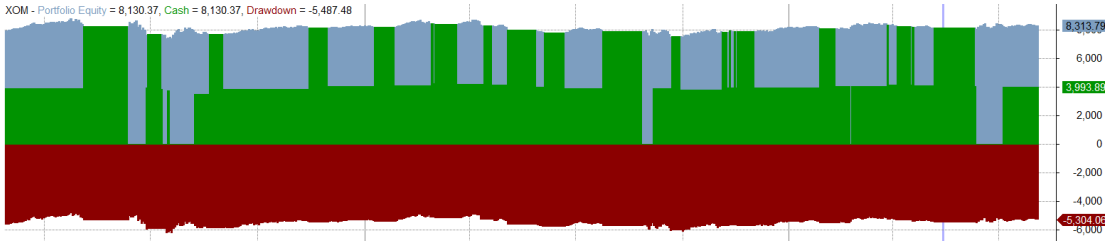
Looking carefully at the results, there are times when there are two positions, times when there is only one, and times when there are none. When processing a portfolio backtest, AmiBroker starts out with 100% cash and knows the maximum number of positions it can hold simultaneously (2 in this case). The code in Example7A tells AmiBroker to divide the available funds into two, giving each 50%. The trading system is applied to each of the ticker symbols in the watchlist. When the first (in chronological order) buy signal occurs, one position is taken and as many shares as possible are purchased with 50% of the funds. Whenever there is a sell signal for that symbol, those shares are sold and the cash made available for another purchase. In the mean time, when the next buy signal occurs, the remaining 50% is used to buy shares in that stock. The buying and selling, using funds and returning funds, continues for the length of time specified.

Plotting the portfolio equity gives a graphic view of the process.

Click the Equity icon.

A new pane opens at the bottom of the chart window.

The two positions are evident, one above the other. When a stock position is held, the plot is blue. When cash is held, the plot is green. The chart shows the ebb and flow of positions - sometimes two, other times one, other times none. The dark brown area at the bottom shows the drawdown. For this run, the final equity was $8,313. The drawdown when the test period ended was $5,304. Drawdown is the decrease in equity from the highest high equity value.



On the Analysis tab, Click the Report icon.

Read the Statistics. Examine the Charts.

This run was made allowing a maximum of two simultaneous positions. The code can be modified to study the effect of allowing fewer or more positions. Using Formula Editor, modify Example7A to agree with the code that follows, and save it as Example7B.afl.

```
// Example7B.afl
// A trading system based on the
// crossover of two moving averages.
// Buy when the faster moving average
// crosses up through the slower moving
// average.
// Sell when the faster moving average
// crosses down through the slower
// moving average.
// This version adds multiple positions
// for use with a portfolio.

NumPos = 3;
SetOption( “MaxOpenPositions”, NumPos );
SetPositionSize( 100 / NumPos, spsPercentOfEquity );

FastMA = MA( C, 5 );
SlowMA = MA( C, 20 );

Buy = Cross( FastMA, SlowMA );
Sell = Cross( SlowMA, FastMA );
```

Rerun the portfolio backtest. Plot the portfolio equity. Read the report.

Exit AmiBroker.

Exercise 8 – Optimize a Trading System

In this example, the moving average crossover system introduced in Example 6 is studied in more detail. The values for the two moving average lengths were guesses. In this example they will be chosen through an optimization process. An optimization is an organized search, done in such a way that a large number of alternative values of parameters are considered. For each set of values, the trading system is tested. The report produced at the conclusion of the optimization lists the profitability and other statistics for each of the sets of values tested. The hope is that the best set of values will continue to be profitable in the future.

Start AmiBroker.

Using Formula Editor:

Open Example6B.afl.

Change it to agree with the code shown below.

```
// Example8A.afl
// A trading system based on the
// crossover of two moving averages.
// Buy when the faster moving average
// crosses up through the slower moving
// average.
// Sell when the faster moving average
// crosses down through the slower
// moving average.
// Parameters are introduced.

FastMALength = 5;
SlowMALength = 20;

FastMA = MA( C, FastMALength );
SlowMA = MA( C, SlowMALength );

Buy = Cross( FastMA, SlowMA );
Sell = Cross( SlowMA, FastMA );

Plot( C, “C”, colorBlack, styleCandle );
e = Equity();
Plot( e, “equity”, colorGreen, styleLine | styleOwnScale );
```

Save the new program as Example8A.afl.

Compare the statements that were changed.

Example6B: `FastMA = MA ( C, 5 );`

Example8A:
```
FastMALength = 5;
FastMA = MA ( C, FastMALength);
```

A numeric value is assigned to a variable FastMALength, then that variable is used to calculate the moving average. Computationally, Example6B and Example8A are the same. But using the variable gives some flexibility.

Optimize Function

AmiBroker has a built-in function, Optimize, that has this format:

```
V = Optimize (“Name”, Default, Initial, Final, Increment);
```

- V is a variable that will be assigned one value at a time, and a series of values as the search takes place.
- Name is the name of the variable as it will appear in the reports.
- Default is the value that will be assigned to V if the run is a Backtest, Explore, or Scan -- anything except an optimization.
- Initial is the first value to be tested during the optimization.
- Final is the last value to be tested.
- Increment is the step taken from one iteration to the next.

For example, `V = Optimize (“Length”, 5, 1, 20, 1);`

If only a backtest is being run, V will be assigned the value 5.

If an optimization is being run, V will be assigned 20 values, one for each optimization pass. The first will be 1, then 2, then 3, and so forth, with the final value assigned to V being 20. From 1 to 20 in steps of 1.

EXAMPLE8B.AFL

Using the Formula Editor, modify Example8A so that it agrees with the code shown below.

```
// Example8B.afl
// A trading system based on the
// crossover of two moving averages.
// Buy when the faster moving average
// crosses up through the slower moving
// average.
// Sell when the faster moving average
// crosses down through the slower
// moving average.
//
// This version can be optimized to
// find the best values for the two
// moving average lengths.

FastMALength = Optimize( “FMA”, 5, 1, 31, 2 );
SlowMALength = Optimize( “SMA”, 20, 2, 30, 2 );

FastMA = MA( C, FastMALength );
SlowMA = MA( C, SlowMALength );

Buy = Cross( FastMA, SlowMA );
Sell = Cross( SlowMA, FastMA );

Plot( C, “C”, colorBlack, styleCandle );
e = Equity();
Plot( e, “equity”, colorGreen, styleLine | styleOwnScale );
```

Save it as Example8B.afl.

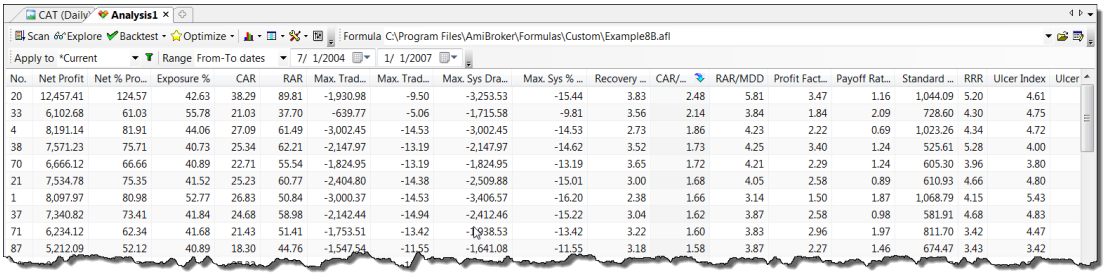
Note there are two Optimize statements. The first will assign values from 1 to 31 in steps of 2 to the variable named FastMALength. The second will assign values of 2 through 30 by steps of 2 to the variable named SlowMALength. There are 16 different values in the first statement, 15 different values in the second statement. To do the complete search, FastMALength will be set to 1, then all 15 values of SlowMALength tried; following that, FastMALength will be set to 3, then all 15 values of SlowMALength will be tried; and so forth until all 240 combinations have been tested.

To perform the Optimization test, using the Analysis tab:

1. In Formula File, load Example8B.afl.
2. Load CAT as the current ticker symbol.
3. In Apply to, click Current symbol.

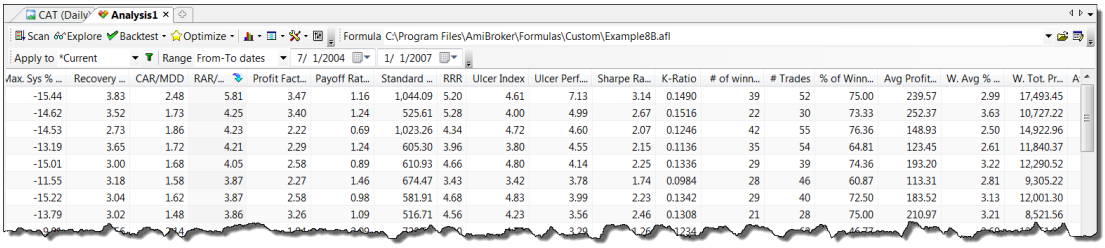
4. In Range, set the From date to 7/1/2004.
5. In Range, set the To Date to 1/1/2007.
6. Click Optimize.

The Results List will fill with 240 rows, one for each combination of FastMALength and SlowMALength.



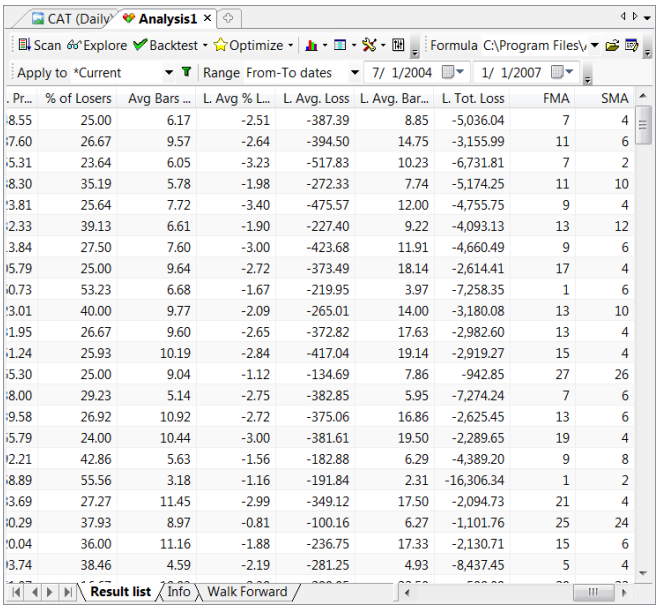
No.	Net Profit	Net % Pro...	Exposure %	CAR	RAR	Max. Trad...	Max. Trad...	Max. Sys Dra...	Max. Sys % ...	Recovery ...	CAR/...	RAR/MDD	Profit Fact...	Payoff Rat...	Standard ...	RRR	Ulcer Index	Ulcer
20	12,457.41	124.57	42.63	38.29	89.81	-1,930.98	-9.50	-3,253.53	-15.44	3.83	2.48	5.81	3.47	1.16	1,044.09	5.20	4.61	
33	6,102.68	61.03	55.78	21.03	37.70	-639.77	-5.06	-1,715.58	-9.81	3.56	2.14	3.84	1.84	2.09	728.60	4.30	4.75	
4	8,191.14	81.91	44.06	27.09	61.49	-3,002.45	-14.53	-3,002.45	-14.53	2.73	1.86	4.23	2.22	0.69	1,023.26	4.34	4.72	
38	7,571.23	75.71	40.73	25.34	62.21	-2,147.97	-13.19	-2,147.97	-14.62	3.52	1.73	4.25	3.40	1.24	525.61	5.28	4.00	
70	6,666.12	66.66	40.89	22.71	55.54	-1,824.95	-13.19	-1,824.95	-13.19	3.65	1.72	4.21	2.29	1.24	605.30	3.96	3.80	
21	7,534.78	75.35	41.52	25.23	60.77	-2,404.80	-14.38	-2,509.88	-15.01	3.00	1.68	4.05	2.58	0.89	610.93	4.66	4.80	
1	8,097.97	80.98	52.77	26.83	50.84	-3,000.37	-14.53	-3,406.57	-16.20	2.38	1.66	3.14	1.50	1.87	1,068.79	4.15	5.43	
37	7,340.82	73.41	41.84	24.68	58.98	-2,142.44	-14.94	-2,412.46	-15.22	3.04	1.62	3.87	2.58	0.98	581.91	4.68	4.83	
71	6,234.12	62.34	41.68	21.43	51.41	-1,753.51	-13.42	-1,938.53	-13.42	3.22	1.60	3.83	2.96	1.97	811.70	3.42	4.47	
87	5,212.09	52.12	40.89	18.30	44.76	-1,547.54	-11.55	-1,641.08	-11.55	3.18	1.58	3.87	2.27	1.46	674.47	3.43	3.42	

Sort the results according to some measure of the performance that you like. (Click the heading of any column to sort using that column.) In general, the measure of performance is called a fitness function or an objective function. There are several fitness functions in the results that both reward equity growth and penalize drawdown. These include CAR/MDD, RAR/MDD, RRR, Ulcer Performance Index, and K-ratio. The results, sorted by RAR/MDD, are shown in the next image.



Max. Sys % ...	Recovery ...	CAR/MDD	RAR/...	Profit Fact...	Payoff Rat...	Standard ...	RRR	Ulcer Index	Ulcer Perf...	Sharpe Ra...	K-Ratio	# of winn...	# Trades	% of Winn...	Avg Profit...	W. Avg % ...	W. Tot. Pr...	A
-15.44	3.83	2.48	5.81	3.47	1.16	1,044.09	5.20	4.61	7.13	3.14	0.1490	39	52	75.00	239.57	2.99	17,493.45	
-14.62	3.52	1.73	4.25	3.40	1.24	525.61	5.28	4.00	4.99	2.67	0.1516	22	30	73.33	252.37	3.63	10,727.22	
-14.53	2.73	1.86	4.23	2.22	0.69	1,023.26	4.34	4.72	4.60	2.07	0.1246	42	55	76.36	148.93	2.50	14,922.96	
-13.19	3.65	1.72	4.21	2.29	1.24	605.30	3.96	3.80	4.55	2.15	0.1136	35	54	64.81	123.45	2.61	11,840.37	
-15.01	3.00	1.68	4.05	2.58	0.89	610.93	4.66	4.80	4.14	2.25	0.1336	29	39	74.36	193.20	3.22	12,290.52	
-11.55	3.18	1.58	3.87	2.27	1.46	674.47	3.43	3.42	3.78	1.74	0.0984	28	46	60.87	113.31	2.81	9,305.22	
-15.22	3.04	1.62	3.87	2.58	0.98	581.91	4.68	4.83	3.99	2.23	0.1342	29	40	72.50	183.52	3.13	12,001.30	
-13.79	3.02	1.48	3.86	3.26	1.09	516.71	4.56	4.23	3.56	2.46	0.1308	21	28	75.00	210.97	3.21	8,521.56	

Scroll over to the far right and note the values of FMA and SMA that correspond to the best fitness value. They are FMA = 7 and SMA = 4.



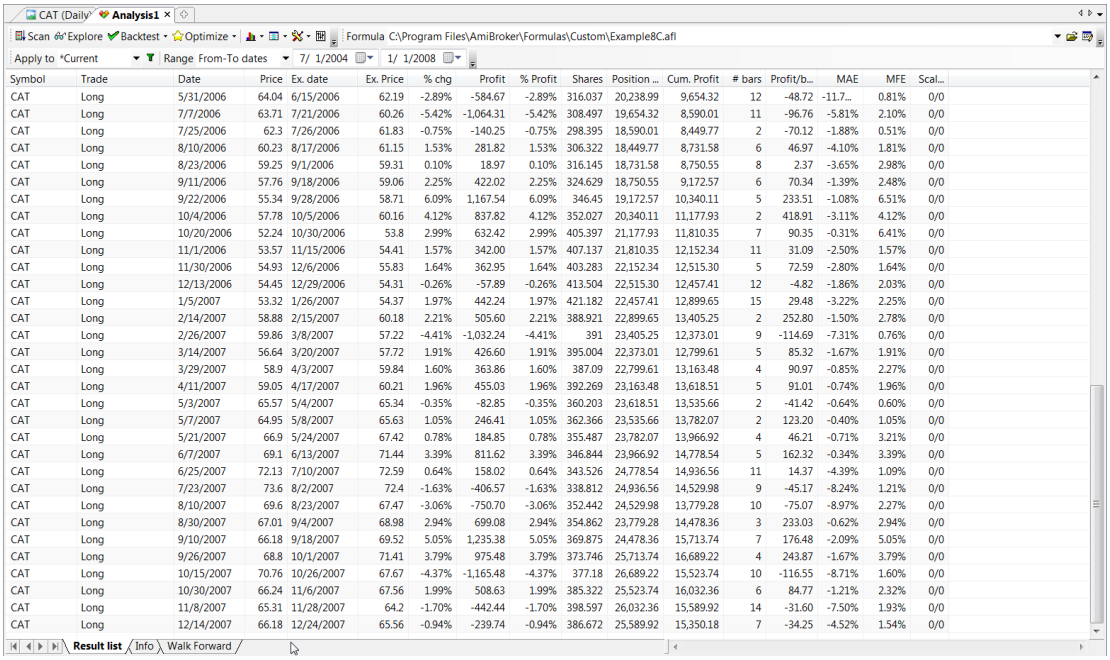
. Pr...	% of Losers	Avg Bars ...	L. Avg % L...	L. Avg. Loss	L. Avg. Bar...	L. Tot. Loss	FMA	SMA
8.55	25.00	6.17	-2.51	-387.39	8.85	-5,036.04	7	4
7.60	26.67	9.57	-2.64	-394.50	14.75	-3,155.99	11	6
5.31	23.64	6.05	-3.23	-517.83	10.23	-6,731.81	7	2
8.30	35.19	5.78	-1.98	-272.33	7.74	-5,174.25	11	10
3.81	25.64	7.72	-3.40	-475.57	12.00	-4,755.75	9	4
2.33	39.13	6.61	-1.90	-227.40	9.22	-4,093.13	13	12
.3.84	27.50	7.60	-3.00	-423.68	11.91	-4,660.49	9	6
5.79	25.00	9.64	-2.72	-373.49	18.14	-2,614.41	17	4
0.73	53.23	6.68	-1.67	-219.95	3.97	-7,258.35	1	6
3.01	40.00	9.77	-2.09	-265.01	14.00	-3,180.08	13	10
1.95	26.67	9.60	-2.65	-372.82	17.63	-2,982.60	13	4
1.24	25.93	10.19	-2.84	-417.04	19.14	-2,919.27	15	4
5.30	25.00	9.04	-1.12	-134.69	7.86	-942.85	27	26
8.00	29.23	5.14	-2.75	-382.85	5.95	-7,274.24	7	6
9.58	26.92	10.92	-2.72	-375.06	16.86	-2,625.45	13	6
5.79	24.00	10.44	-3.00	-381.61	19.50	-2,289.65	19	4
2.21	42.86	5.63	-1.56	-182.88	6.29	-4,389.20	9	8
8.89	55.56	3.18	-1.16	-191.84	2.31	-16,306.34	1	2
3.69	27.27	11.45	-2.99	-349.12	17.50	-2,094.73	21	4
0.29	37.93	8.97	-0.81	-100.16	6.27	-1,101.76	25	24
0.04	36.00	11.16	-1.88	-236.75	17.33	-2,130.71	15	6
3.74	38.46	4.59	-2.19	-281.25	4.93	-8,437.45	5	4

Return to Example8B.afl and enter those values as the defaults in the two optimize statements.

```
FastMALength = Optimize( “FMA”, 7, 1, 31, 2 );
SlowMALength = Optimize( “SMA”, 4, 2, 30, 2 );
```

Save the program as Example8C.afl.

On the Analysis tab, change the To Date to 1/1/2008. Click Backtest. On the Result List, scroll down so you can see the trades for the period 1/1/2007 through 1/1/2008. The optimization that was run used the period 7/1/2004 through 1/1/2007. This is called the in-sample period. In-sample results are always good, but they have little value in estimating future performance. The backtest using the period 7/1/2004 through 1/1/2008 includes the year 2008, which was not used in determination of the best parameter values. This is called an out-of-sample period. The account balance increased in 2008, suggesting that the system might be worth further study. See my companion books, including *Quantitative Trading Systems*, for much more discussion of in-sample, out-of-sample, and system validation.



Symbol	Trade	Date	Price	Ex. date	Ex. Price	% chg	Profit	% Profit	Shares	Position ...	Cum. Profit	# bars	Profit/b...	MAE	MFE	Scal...
CAT	Long	5/31/2006	64.04	6/15/2006	62.19	-2.89%	-584.67	-2.89%	316.037	20,238.99	9,654.32	12	-48.72	-11.7...	0.81%	0/0
CAT	Long	7/7/2006	63.71	7/21/2006	60.26	-5.42%	-1,064.31	-5.42%	308.497	19,654.32	8,590.01	11	-96.76	-5.81%	2.10%	0/0
CAT	Long	7/25/2006	62.3	7/26/2006	61.83	-0.75%	-140.25	-0.75%	298.395	18,590.01	8,449.77	2	-70.12	-1.88%	0.51%	0/0
CAT	Long	8/10/2006	60.23	8/17/2006	61.15	1.53%	281.82	1.53%	306.322	18,449.77	8,731.58	6	46.97	-4.10%	1.81%	0/0
CAT	Long	8/23/2006	59.25	9/1/2006	59.31	0.10%	18.97	0.10%	316.145	18,731.58	8,750.55	8	2.37	-3.65%	2.98%	0/0
CAT	Long	9/11/2006	57.76	9/18/2006	59.06	2.25%	422.02	2.25%	324.629	18,750.55	9,172.57	6	70.34	-1.39%	2.48%	0/0
CAT	Long	9/22/2006	55.34	9/28/2006	58.71	6.09%	1,167.54	6.09%	346.45	19,172.57	10,340.11	5	233.51	-1.08%	6.51%	0/0
CAT	Long	10/4/2006	57.78	10/5/2006	60.16	4.12%	837.82	4.12%	352.027	20,340.11	11,177.93	2	418.91	-3.11%	4.12%	0/0
CAT	Long	10/20/2006	52.24	10/30/2006	53.8	2.99%	632.42	2.99%	405.397	21,177.93	11,810.35	7	90.35	-0.31%	6.41%	0/0
CAT	Long	11/1/2006	53.57	11/15/2006	54.41	1.57%	342.00	1.57%	407.137	21,810.35	12,152.34	11	31.09	-2.50%	1.57%	0/0
CAT	Long	11/30/2006	54.93	12/6/2006	55.83	1.64%	362.95	1.64%	403.283	22,152.34	12,515.30	5	72.59	-2.80%	1.64%	0/0
CAT	Long	12/13/2006	54.45	12/29/2006	54.31	-0.26%	-57.89	-0.26%	413.504	22,515.30	12,457.41	12	-4.82	-1.86%	2.03%	0/0
CAT	Long	1/5/2007	53.32	1/26/2007	54.37	1.97%	442.24	1.97%	421.182	22,457.41	12,899.65	15	29.48	-3.22%	2.25%	0/0
CAT	Long	2/14/2007	58.88	2/15/2007	60.18	2.21%	505.60	2.21%	388.921	22,899.65	13,405.25	2	252.80	-1.50%	2.78%	0/0
CAT	Long	2/26/2007	59.86	3/8/2007	57.22	-4.41%	-1,032.24	-4.41%	391	23,405.25	12,373.01	9	-114.69	-7.31%	0.76%	0/0
CAT	Long	3/14/2007	56.64	3/20/2007	57.72	1.91%	426.60	1.91%	395.004	22,373.01	12,799.61	5	85.32	-1.67%	1.91%	0/0
CAT	Long	3/29/2007	58.9	4/3/2007	59.84	1.60%	363.86	1.60%	387.09	22,799.61	13,163.48	4	90.97	-0.85%	2.27%	0/0
CAT	Long	4/11/2007	59.05	4/17/2007	60.21	1.96%	455.03	1.96%	392.269	23,163.48	13,618.51	5	91.01	-0.74%	1.96%	0/0
CAT	Long	5/3/2007	65.57	5/4/2007	65.34	-0.35%	-82.85	-0.35%	360.203	23,618.51	13,535.66	2	-41.42	-0.64%	0.60%	0/0
CAT	Long	5/7/2007	64.95	5/8/2007	65.63	1.05%	246.41	1.05%	362.366	23,535.66	13,782.07	2	123.20	-0.40%	1.05%	0/0
CAT	Long	5/21/2007	66.9	5/24/2007	67.42	0.78%	184.85	0.78%	355.487	23,782.07	13,966.92	4	46.21	-0.71%	3.21%	0/0
CAT	Long	6/7/2007	69.1	6/13/2007	71.44	3.39%	811.62	3.39%	346.844	23,966.92	14,778.54	5	162.32	-0.34%	3.39%	0/0
CAT	Long	6/25/2007	72.13	7/10/2007	72.59	0.64%	158.02	0.64%	343.526	24,778.54	14,936.56	11	14.37	-4.39%	1.09%	0/0
CAT	Long	7/23/2007	73.6	8/2/2007	72.4	-1.63%	-406.57	-1.63%	338.812	24,936.56	14,529.98	9	-45.17	-8.24%	1.21%	0/0
CAT	Long	8/10/2007	69.6	8/23/2007	67.47	-3.06%	-750.70	-3.06%	352.442	24,529.98	13,779.28	10	-75.07	-8.97%	2.27%	0/0
CAT	Long	8/30/2007	67.01	9/4/2007	68.98	2.94%	699.08	2.94%	354.862	23,779.28	14,478.36	3	233.03	-0.62%	2.94%	0/0
CAT	Long	9/10/2007	66.18	9/18/2007	69.52	5.05%	1,235.38	5.05%	369.875	24,478.36	15,713.74	7	176.48	-2.09%	5.05%	0/0
CAT	Long	9/26/2007	68.8	10/1/2007	71.41	3.79%	975.48	3.79%	373.746	25,713.74	16,689.22	4	243.87	-1.67%	3.79%	0/0
CAT	Long	10/15/2007	70.76	10/26/2007	67.67	-4.37%	-1,165.48	-4.37%	377.18	26,689.22	15,523.74	10	-116.55	-8.71%	1.60%	0/0
CAT	Long	10/30/2007	66.24	11/6/2007	67.56	1.99%	508.63	1.99%	385.322	25,523.74	16,032.36	6	84.77	-1.21%	2.32%	0/0
CAT	Long	11/8/2007	65.31	11/28/2007	64.2	-1.70%	-442.44	-1.70%	398.597	26,032.36	15,589.92	14	-31.60	-7.50%	1.93%	0/0
CAT	Long	12/14/2007	66.18	12/24/2007	65.56	-0.94%	-239.74	-0.94%	386.672	25,589.92	15,350.18	7	-34.25	-4.52%	1.54%	0/0

An Important Observation

Take another look at the results of the optimization. The length of the "fast" moving average is 7; the length of the "slow" moving average is 4. Traditional technical analysis tells us that we should buy when the fast moving average crosses up through the slow moving average. The result of the optimization run in this example suggests that is not always the case. In this example, we are buying when the fast moving average crosses down through the slow moving average.

To buy when the fast moving average is rising is a trend following system -- buy when recent prices are rising faster than older prices. This works well for very stable price series, such as high-yield bond funds.

To buy when the fast moving average is falling is a mean reversion system -- buy when prices are weak, anticipating that they will return to their mean. This works well for very active price series. See my companion book, *Mean Reversion Trading Systems*, for more discussion of mean reversion systems.

Exit AmiBroker.

Example 9 - Perform Walk Forward Validation

In this example, you will use the walk-forward technique to validate a trading system.

In applying technical analysis, several assumptions are made:

- Patterns exist that precede profitable trading opportunities.
- The trading system can detect those patterns.
- The patterns persist beyond the in-sample period used to develop the system for a long enough time that we can trade profitably.

When any trading system is developed, the process involves searching through a set of data, looking for the patterns. The data that is searched is called the in-sample data. The results obtained by using the system on the in-sample data are always good. The system developer does not stop looking until the results are good.

The proof that a trading system has found a recognizable pattern, and has not just mined the data, is performance on data that has never been used to develop the system. That never-before-used data is called the out-of-sample data.

The walk forward process consists of several iterations of:

1. Developing a system over a set of in-sample data.
2. Testing the profitability over a set of out-of-sample data.
3. Stepping forward in time to a new set of in-sample data and a new set of out-of-sample data.

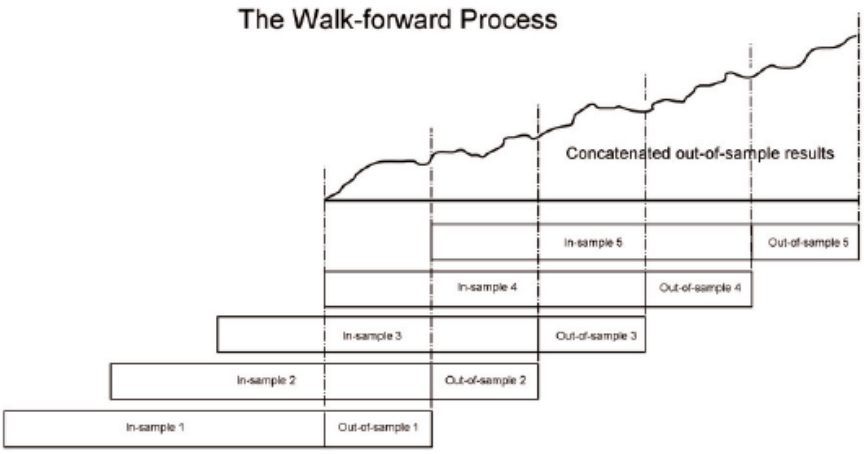
The combined results of all the out-of-sample data are used to decide whether the system recognizes profit potential or not.

There are several major advantages to using walk forward testing:

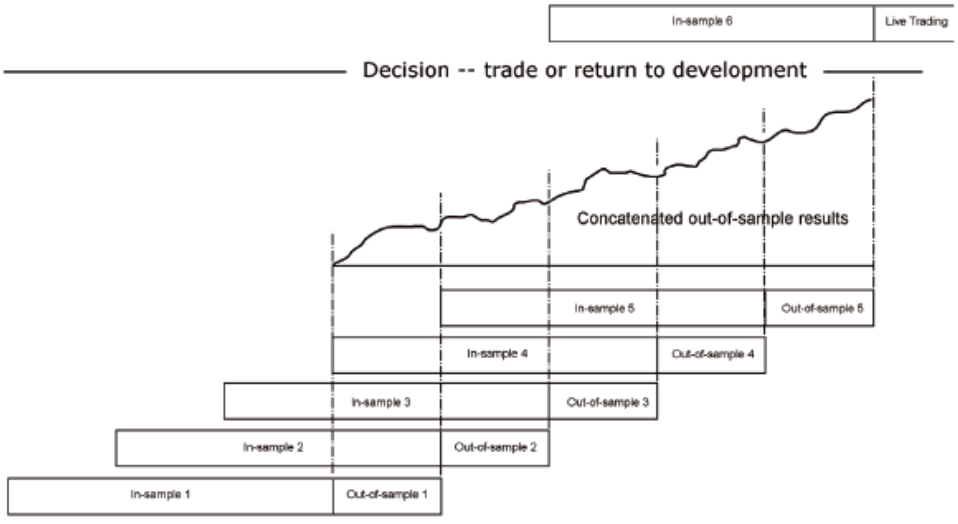
- The in-sample period can be relatively short, giving the system an opportunity to stay in synchronization with the underlying market.
- Each transition from in-sample to out-of-sample gives the developer one more data point. Eventually, when trades are made with real money, the decision to trade is a decision to trust the in-sample results. The greater the number of steps in the walk forward process, the greater the number of transitions from in-sample to out-of-sample the developer has seen. Each successful transition raises confidence that the system is a good one.
- The out-of-sample results from the walk forward run are the best estimate of future performance of the system. They provide the information used in analyzing the reward and risk profiles, and in determining the position size that will

produce greatest account growth while holding drawdown to a level of tolerance determined bythe trader.

The following graphics might help understand the process.



The decision whether to trade the system or go back to the drawing board is made after reviewing all the out-of-sample results.



Start AmiBroker.

Load AXP using daily data. My data is 1/1/2000 through 8/3/2012, downloaded from Yahoo using AmiQuote.

Load Example8B.afl. No changes to the afl code are necessary. Send it to Analysis.

On the Analysis tab, Settings, Walk Forward tab.

Select Easy Mode (EOD).

In-sample Start date: 7/1/2000

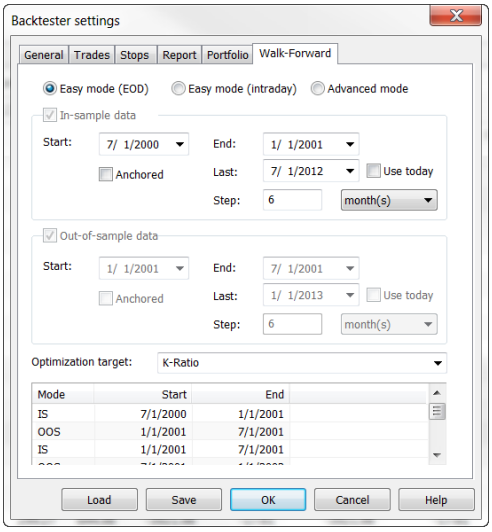
End: 1/1/2001

Last: 7/1/2012

Step: 6 months

Optimization target: K-Ratio

Note the table at the bottom where the in-sample and out-of-sample periods have been computed. The period 7/1/2012 to 1/1/2013 is the final out-of-sample (as of this writing).



Click OK.

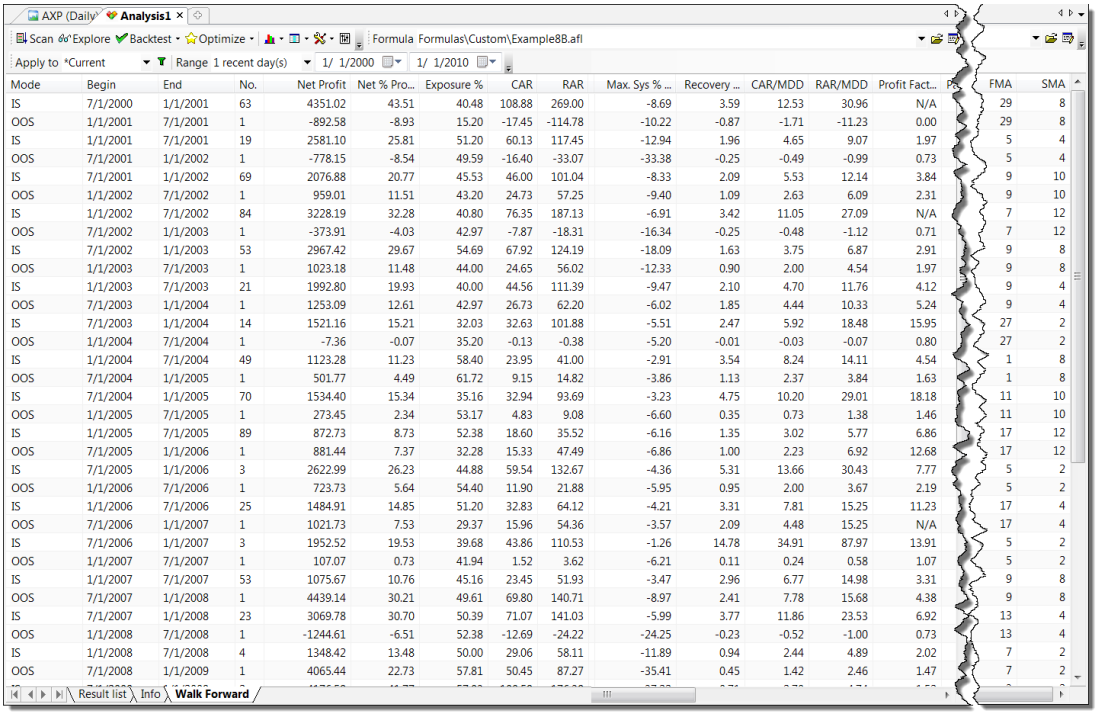
Back on the Analysis tab.

Apply to: Current.

Using the Optimize pull-down menu, click Walk Forward.

The optimization runs will start. There will be 24 iterations of in-sample and out-of-sample. Each in-sample period is six months long. The first in-sample period starts 7/1/2000. The final (24th) in-sample period begins 1/1/2012. The out-of-sample periods are also six months long and each immediately follows an in-sample period. There are 23 full out-of-sample periods and one partial. The final out-of-sample period begins 7/1/2012 and ends with the latest data.

As the walk forward runs progress, pairs of lines are created and displayed in a window on the Walk Forward tab. The first line of each pair gives the results for the in-sample period. The second line gives the results for the associated out-of-sample period.



Mode	Begin	End	No.	Net Profit	Net % Pro...	Exposure %	CAR	RAR	Max. Sys % ...	Recovery ...	CAR/MDD	RAR/MDD	Profit Fact...	FMA	SMA
IS	7/1/2000	1/1/2001	63	4351.02	43.51	40.48	108.88	269.00	-8.69	3.59	12.53	30.96	N/A	29	8
OOS	1/1/2001	7/1/2001	1	-892.58	-8.93	15.20	-17.45	-114.78	-10.22	-0.87	-1.71	-11.23	0.00	29	8
IS	1/1/2001	7/1/2001	19	2581.10	25.81	51.20	60.13	117.45	-12.94	1.96	4.65	9.07	1.97	5	4
OOS	7/1/2001	1/1/2002	1	-778.15	-8.54	49.59	-16.40	-33.07	-33.38	-0.25	-0.49	-0.99	0.73	5	4
IS	7/1/2001	1/1/2002	69	2076.88	20.77	45.53	46.00	101.04	-8.33	2.09	5.53	12.14	3.84	9	10
OOS	1/1/2002	7/1/2002	1	959.01	11.51	43.20	24.73	57.25	-9.40	1.09	2.63	6.09	2.31	9	10
IS	1/1/2002	7/1/2002	84	3228.19	32.28	40.80	76.35	187.13	-6.91	3.42	11.05	27.09	N/A	7	12
OOS	7/1/2002	1/1/2003	1	-373.91	-4.03	42.97	-7.87	-18.31	-16.34	-0.25	-0.48	-1.12	0.71	7	12
IS	7/1/2002	1/1/2003	53	2967.42	29.67	54.69	67.92	124.19	-18.09	1.63	3.75	6.87	2.91	9	8
OOS	1/1/2003	7/1/2003	1	1023.18	11.48	44.00	24.65	56.02	-12.33	0.90	2.00	4.54	1.97	9	8
IS	1/1/2003	7/1/2003	21	1992.80	19.93	40.00	44.56	111.39	-9.47	2.10	4.70	11.76	4.12	9	4
OOS	7/1/2003	1/1/2004	1	1253.09	12.61	42.97	26.73	62.20	-6.02	1.85	4.44	10.33	5.24	9	4
IS	7/1/2003	1/1/2004	14	1521.16	15.21	32.03	32.63	101.88	-5.51	2.47	5.92	18.48	15.95	27	2
OOS	1/1/2004	7/1/2004	1	-7.36	-0.07	35.20	-0.13	-0.38	-5.20	-0.01	-0.03	-0.07	0.80	27	2
IS	1/1/2004	7/1/2004	49	1123.28	11.23	58.40	23.95	41.00	-2.91	3.54	8.24	14.11	4.54	1	8
OOS	7/1/2004	1/1/2005	1	501.77	4.49	61.72	9.15	14.82	-3.86	1.13	2.37	3.84	1.63	1	8
IS	7/1/2004	1/1/2005	70	1534.40	15.34	35.16	32.94	93.69	-3.23	4.75	10.20	29.01	18.18	11	10
OOS	1/1/2005	7/1/2005	1	273.45	2.34	53.17	4.83	9.08	-6.60	0.35	0.73	1.38	1.46	11	10
IS	1/1/2005	7/1/2005	89	872.73	8.73	52.38	18.60	35.52	-6.16	1.35	3.02	5.77	6.86	17	12
OOS	7/1/2005	1/1/2006	1	881.44	7.37	32.28	15.33	47.49	-6.86	1.00	2.23	6.92	12.68	17	12
IS	7/1/2005	1/1/2006	3	2622.99	26.23	44.88	59.54	132.67	-4.36	5.31	13.66	30.43	7.77	5	2
OOS	1/1/2006	7/1/2006	1	723.73	5.64	54.40	11.90	21.88	-5.95	0.95	2.00	3.67	2.19	5	2
IS	1/1/2006	7/1/2006	25	1484.91	14.85	51.20	32.83	64.12	-4.21	3.31	7.81	15.25	11.23	17	4
OOS	7/1/2006	1/1/2007	1	1021.73	7.53	29.37	15.96	54.36	-3.57	2.09	4.48	15.25	N/A	17	4
IS	7/1/2006	1/1/2007	3	1952.52	19.53	39.68	43.86	110.53	-1.26	14.78	34.91	87.97	13.91	5	2
OOS	1/1/2007	7/1/2007	1	107.07	0.73	41.94	1.52	3.62	-6.21	0.11	0.24	0.58	1.07	5	2
IS	1/1/2007	7/1/2007	53	1075.67	10.76	45.16	23.45	51.93	-3.47	2.96	6.77	14.98	3.31	9	8
OOS	7/1/2007	1/1/2008	1	4439.14	30.21	49.61	69.80	140.71	-8.97	2.41	7.78	15.68	4.38	9	8
IS	7/1/2007	1/1/2008	23	3069.78	30.70	50.39	71.07	141.03	-5.99	3.77	11.86	23.53	6.92	13	4
OOS	1/1/2008	7/1/2008	1	-1244.61	-6.51	52.38	-12.69	-24.22	-24.25	-0.23	-0.52	-1.00	0.73	13	4
IS	1/1/2008	7/1/2008	4	1348.42	13.48	50.00	29.06	58.11	-11.89	0.94	2.44	4.89	2.02	7	2
OOS	7/1/2008	1/1/2009	1	4065.44	22.73	57.81	50.45	87.27	-35.41	0.45	1.42	2.46	1.47	7	2

Ignore in-sample profitability.

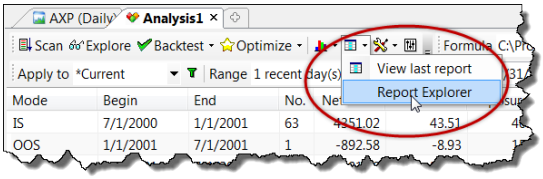
Examine the results for the out-of-sample periods -- 17 of 24 are profitable.

Look at the values of the two parameters (at the right side of the report). In 20 of 24 periods, the best parameters had the lengths of the moving average “reversed.” That is, the

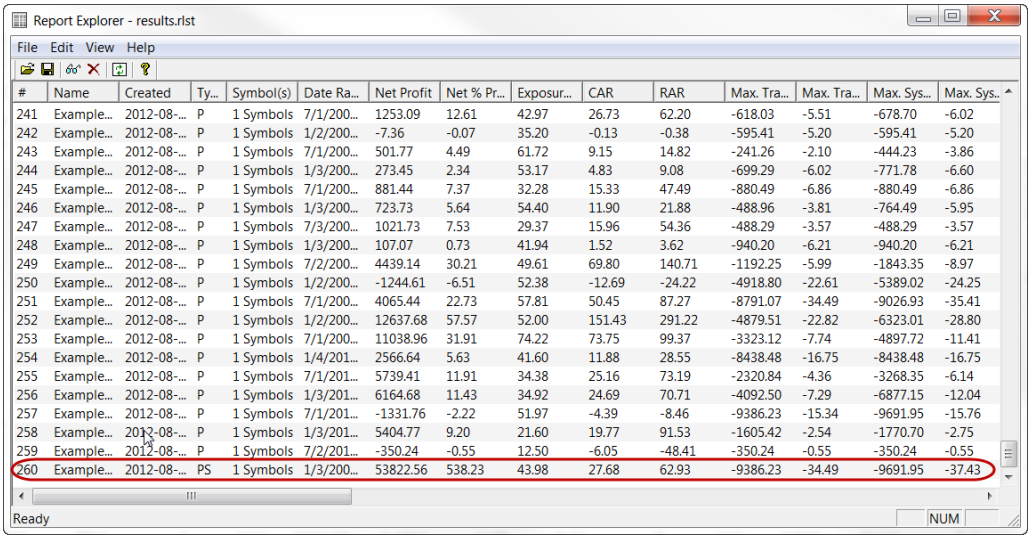
system acts as a mean reversion system, buying when the fast moving average crosses down through the slow moving average.

Note the values of the lengths of the moving averages in the periods. They range from 1 to 29, with some period-to-period changes small and others large. This is a graphic demonstration that trading systems are dynamic. The logic and parameters must adapt to the data as conditions change. There is no single pair of moving average lengths that are best for an extended period of time.

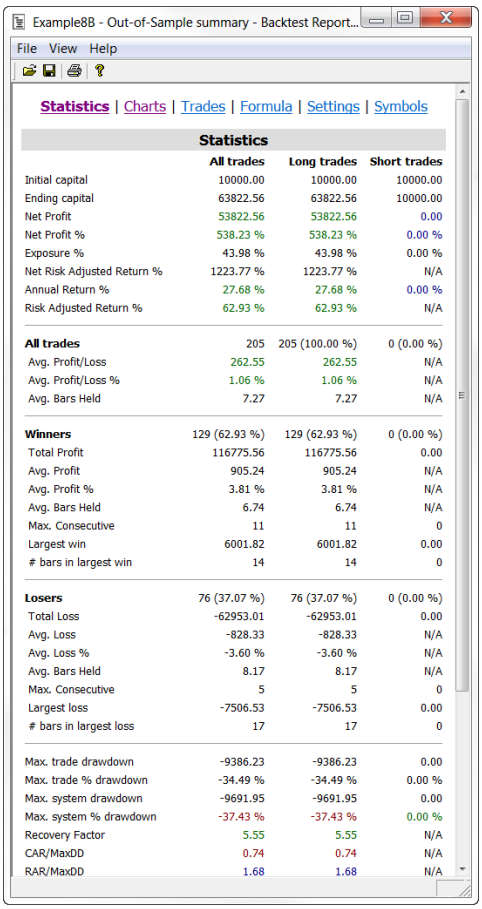
Use the pull-down menu next to the Report icon and select Report Explorer.

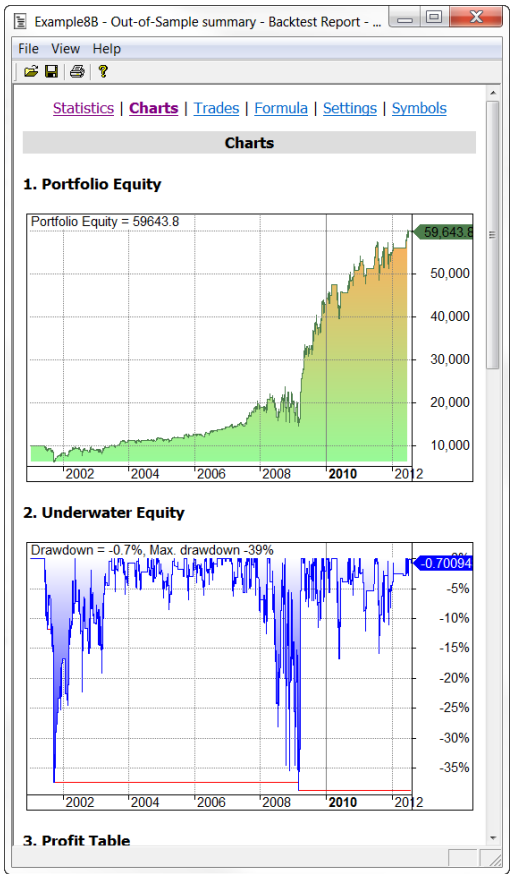


A list of test results opens. Each line is a summary of a backtest run. The final line is a report constructed by concatenating the trading system results of all of the out-of-sample periods (a feature released in AmiBroker 5.56.0, August 2012).



Double click the final line. A backtest report will open. It has the now familiar pages showing the Statistics and Charts.





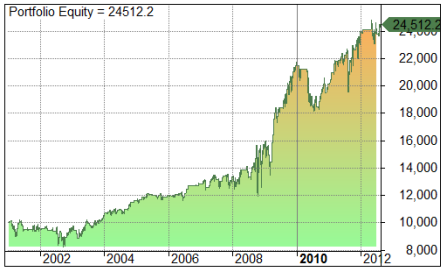
This is an example, not a ready-to-use trading system. Although the results of this example are reasonable, there are many factors beyond those described that should be taken into consideration in system design, testing, validation, and analysis.

Do NOT trade this system without performing your own tests, validation, and analysis.

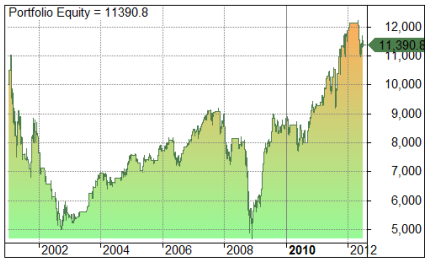
Exit AmiBroker.

While the results are good when trading AXP, they are not that good for all issues. I often test systems using major ETFs. Results for SPY, QQQ, IWM, and EEM are shown below. All of these are the out-of-sample trades from automatic walk forward tests.

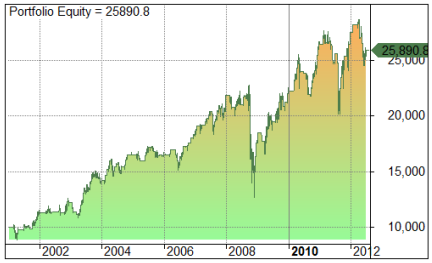
SPY



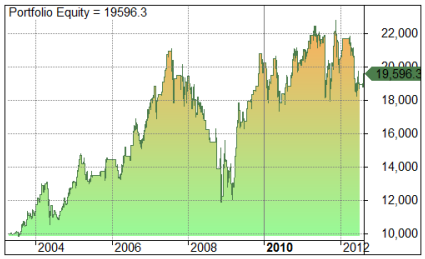
QQQ



IWM



EEM



Readers interested in more discussion might enjoy any or all of my companion books:

Quantitative Trading Systems -- The design, testing, and validation of trading systems.

Modeling Trading System Performance -- Analysis of the health of trading systems, and dynamic determination of position size.

Mean Reversion Trading Systems -- Features of, and considerations for, trading systems that fade the trend.

And my blog site at *BlueOwlPress.com* with more articles and discussions about trading system development. Specifically, read my paper, *Backtesting and Cognitive Dissonance,* for a more detailed discussion of the use of backtesting and the reliability of in-sample results.

Exercise 10 – Scan for Buy and Sell Signals

In this example, you will use the Scan feature to check for buy or sell signals.

Due to the limitations of the trial mode, it will be difficult to demonstrate realistic results of the Scan feature. Assume that Example8B is a validated trading system for all of the symbols in watchlist Basic Materials.

Scan combines a trading system with a watchlist to give a report telling you what buy and sell signals were generated for the symbols in the watchlist. The period of time this report covers is under your control, but it is usually just one or two days.

Start AmiBroker.

Load IP (or any of the issues in the Basic Materials watchlist) using daily data.

Load Example8B.afl. Send it to Analysis.

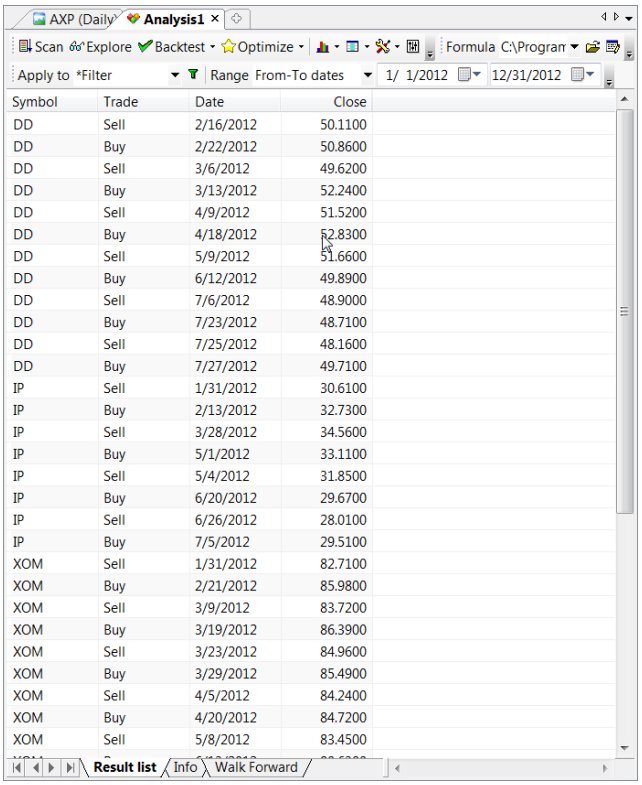
On the Analysis tab:

Apply to: Filter
Watchlist: Basic Materials
Range: From-To dates
From: 1/1/2012
To: The date of the last data bar.

Select Results List tab.

Click Scan.

A list of all Buy and Sell (and Short and Cover, when the system includes them) signals for the issues and date range selected will be generated.



Symbol	Trade	Date	Close
DD	Sell	2/16/2012	50.1100
DD	Buy	2/22/2012	50.8600
DD	Sell	3/6/2012	49.6200
DD	Buy	3/13/2012	52.2400
DD	Sell	4/9/2012	51.5200
DD	Buy	4/18/2012	52.8300
DD	Sell	5/9/2012	51.6600
DD	Buy	6/12/2012	49.8900
DD	Sell	7/6/2012	48.9000
DD	Buy	7/23/2012	48.7100
DD	Sell	7/25/2012	48.1600
DD	Buy	7/27/2012	49.7100
IP	Sell	1/31/2012	30.6100
IP	Buy	2/13/2012	32.7300
IP	Sell	3/28/2012	34.5600
IP	Buy	5/1/2012	33.1100
IP	Sell	5/4/2012	31.8500
IP	Buy	6/20/2012	29.6700
IP	Sell	6/26/2012	28.0100
IP	Buy	7/5/2012	29.5100
XOM	Sell	1/31/2012	82.7100
XOM	Buy	2/21/2012	85.9800
XOM	Sell	3/9/2012	83.7200
XOM	Buy	3/19/2012	86.3900
XOM	Sell	3/23/2012	84.9600
XOM	Buy	3/29/2012	85.4900
XOM	Sell	4/5/2012	84.2400
XOM	Buy	4/20/2012	84.7200
XOM	Sell	5/8/2012	83.4500

Exit AmiBroker.

Chapter 4

AmiBroker Chart Structure

In the earlier portion of this book, rather loose reference was made to databases, layouts, windows, charts, panes, drawing objects, and so forth. To fully understand AmiBroker, it is necessary to understand the structure into which these components fit.

- The top level component is a **Layout**.
- Each **Layout** is associated with a single **Database** when that layout is active.
- Each **Layout** consists of one or more **Windows**.
- Each **Window** has data for one **Symbol**.
- Each **Window** displays with one **Periodicity**.
- Each **Window** contains one or more **Sheets**.
- Each **Sheet** contains one or more **Panes**.
- Each **Pane** contains one or more **Chart**, **Formula**, or **Tool**.

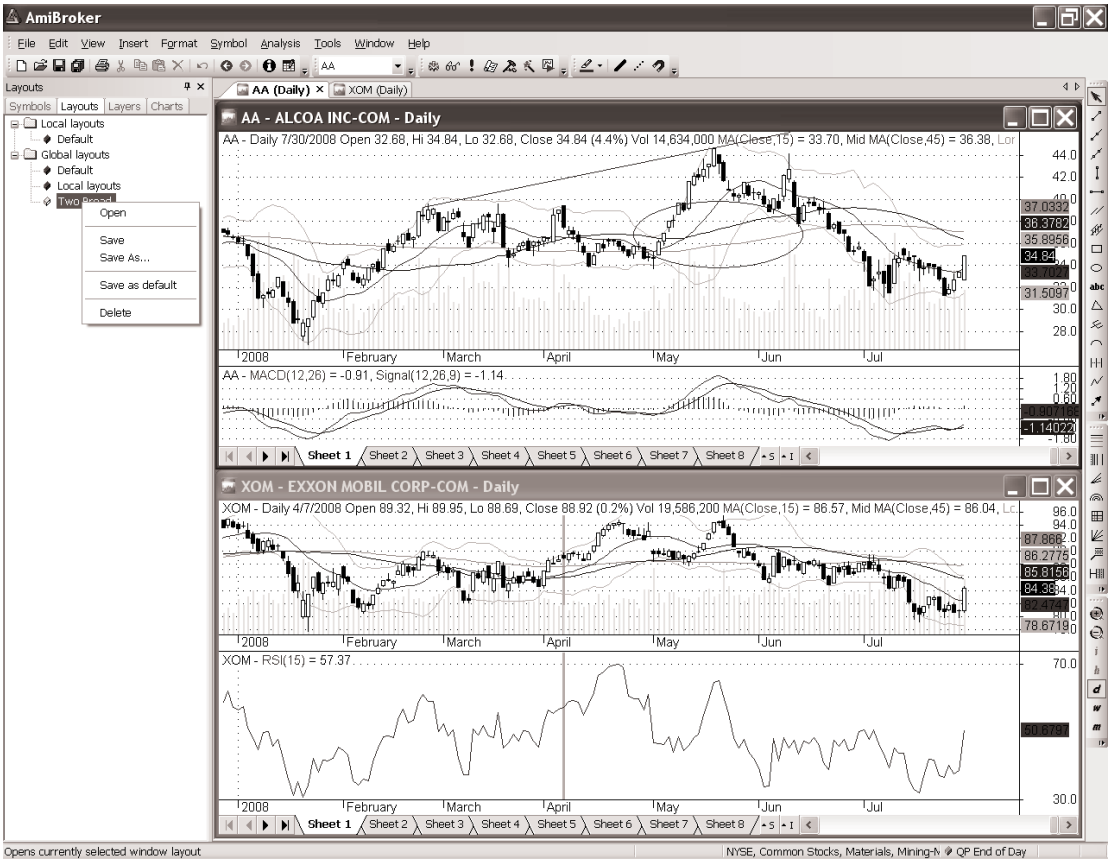
You can have as many layouts as you wish. But only one layout can be active (open) at a time.

Layouts can be *local* or *global*. Local layouts are associated only with a specific database. Each local layout is stored in the database directory it is associated with, in the file named broker.workspace. Global layouts can be used with more than one database. Global layouts are stored in the AmiBroker\Layouts subdirectory.

Window Layouts

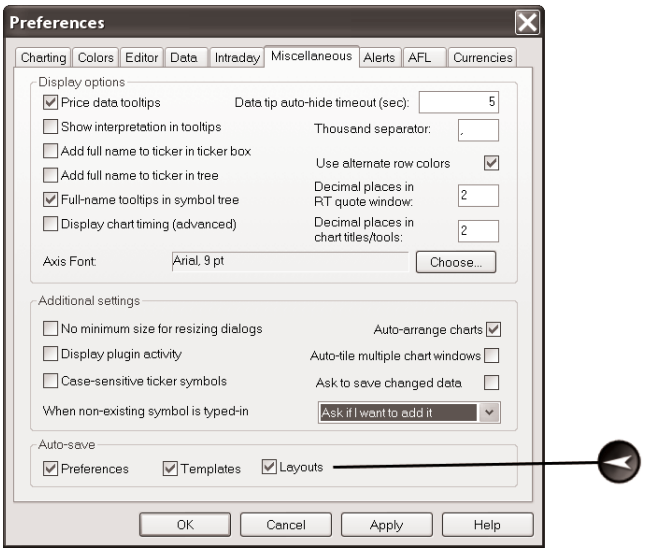
A window layout (that is, an AmiBroker window layout) is a complete set of one or more windows. Each window displays data for its own symbol, has its own display interval, its own size on the monitor, its own set of charts, its own panes -- each of which has its own charts and formulas.

The image below shows a 2-window layout. Each window has a different stock and a different set of indicator panes.



You can have an unlimited number of single or multiple-window layouts. Each can be a custom set up; they do not need to share anything. Switch between them by double clicking on the layout's name in the layouts tab.

The active layout can be automatically saved whenever the database is switched or AmiBroker is exited. To set this, use the Tools menu > Preferences > Miscellaneous. In the Auto-save area, Check the Layouts box. Information saved in layouts include: window sizes and positions, bar interval, symbol, chart sheets, panes, charts, formulas.



You can switch between multiple windows (within a layout) using either the Window menu and selecting the desired window from the list at the bottom of the menu, or using the MDI tabs.

To begin with, a single window layout will probably be sufficient. You will need a multi-window layout if you want to :

- Display charts for more than one ticker symbol
- Display charts of different periodicity
- Display charts showing different range of bars.

While not necessary, you may want a multi-window layout in order to be able to float a window and move it to a second monitor.

Most of the concepts and techniques that are described in the pages that follow are applied to the active pane and are independent of the layout structure.

Block Diagram

On the next page is a block diagram view of layouts, their components, and their relationship to databases.

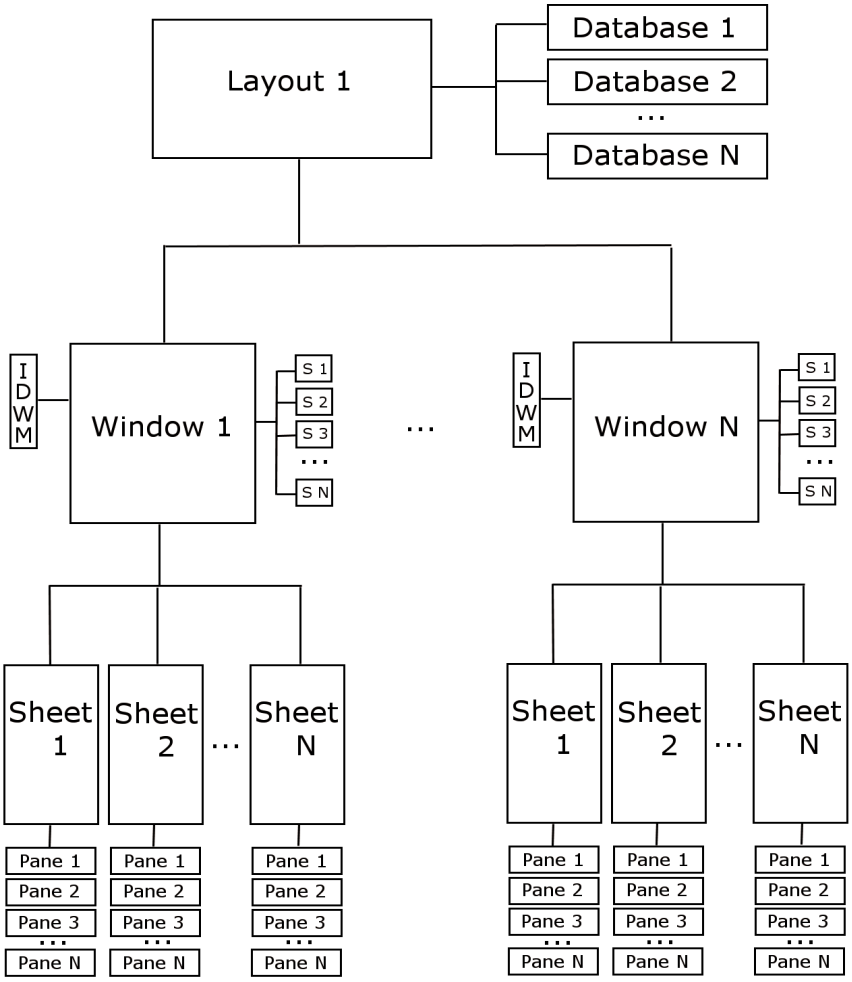
One layout is shown. You can have as many layouts as you want. Only one layout can be open and active at a time. To make another active:

- Save the current database
- Open the new database
- Select the layout you want to use

Everything shown on the block diagram will change to reflect the new layout / database combination - all windows, sheets, panes, tools, watchlists.

To reiterate:

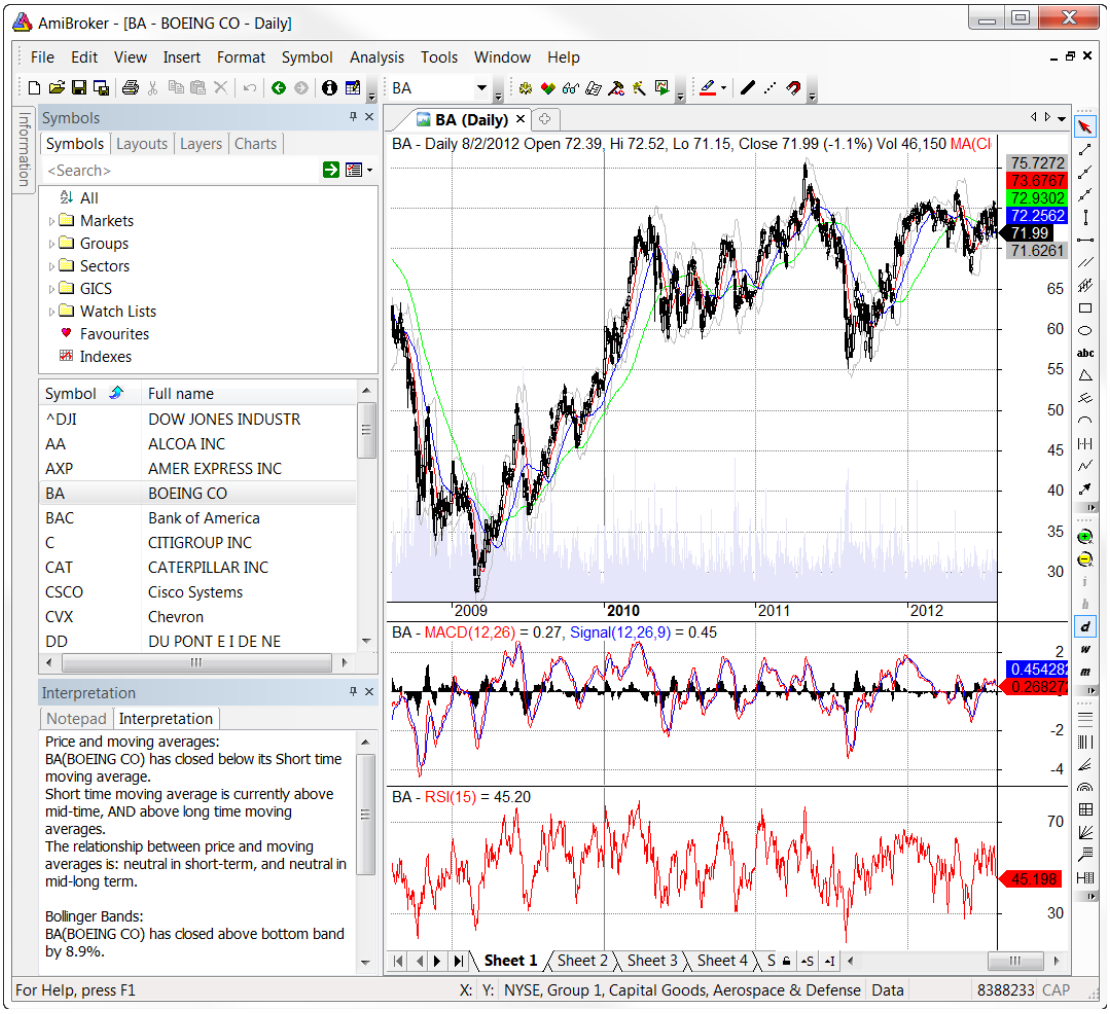
- The top level component of the AmiBroker structure is the layout.
- Each layout can be associated with any number of databases (there is no limit). Only one database is open and active when a layout is active.
- Some layouts - local layouts - are associated with only one database. Other layouts - global layouts - are associated with two or more databases, but only one at a time.
- Each layout consists of one or more windows. Only one window can be active at a time. All of the windows associated with the layout are listed in the windows menu. If you choose a multi-window layout, all the windows will be visible; otherwise only the active window is visible. You can change back and forth from window to window as you wish.
- Each window displays data for one symbol at a time.
- Each window displays data in one time frame at a time.
- Each window contains one or more sheets (there is no limit). Only one sheet can be active at a time. To make another sheet active, click its tab. All sheets display the same range of dates.
- Each sheet contains one or more panes (there is no limit).
- Each pane contains one or more displays of price data, manually applied tools, or formulas. Only one pane can be active at a time. To make another pane active, click anywhere in it.



Chapter 5

The GUI - Graphical User Interface

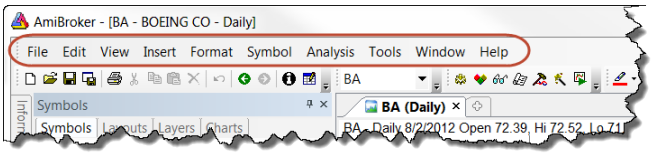
When you start AmiBroker using an end-of-day database, here is what you see.



The AmiBroker User's Guide has excellent descriptions of the features of the GUI. The next few figures identify the components so that you will be able to efficiently search for information about them.

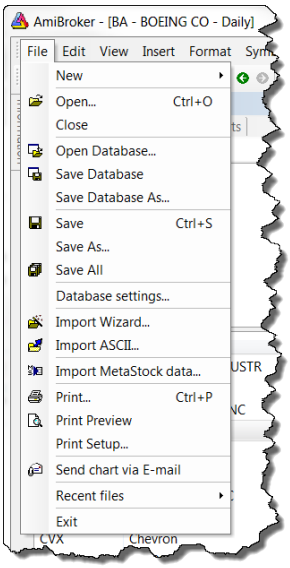
In keeping with Windows standards, the components can be resized, detached, redocked, etc.

Pull Down Menus

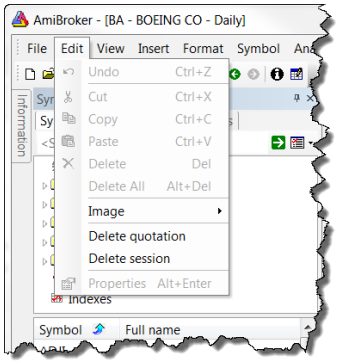


The menus and their functions are pretty self explanatory.

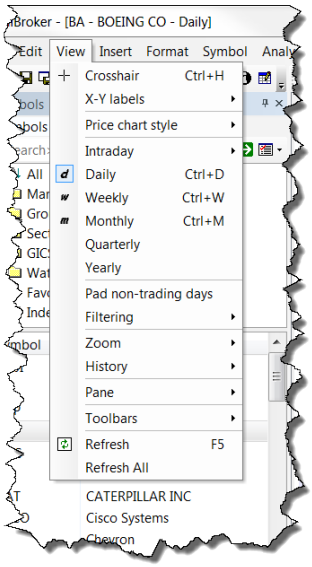
File



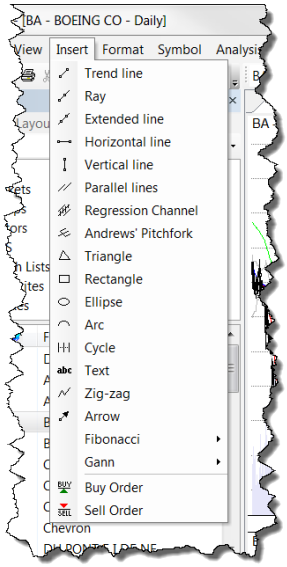
Edit



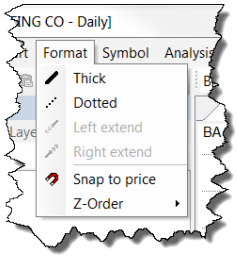
View



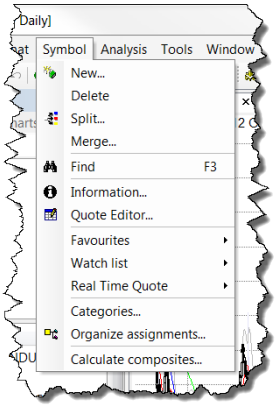
Insert



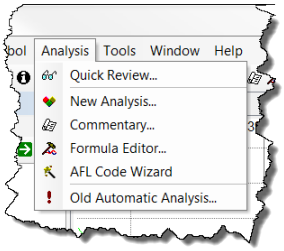
Format



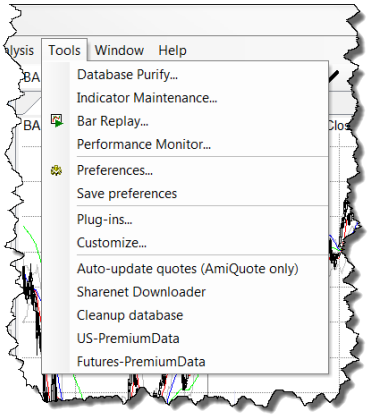
Symbol



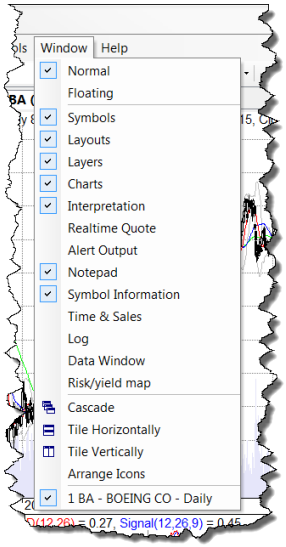
Analysis



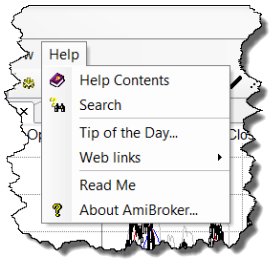
Tools



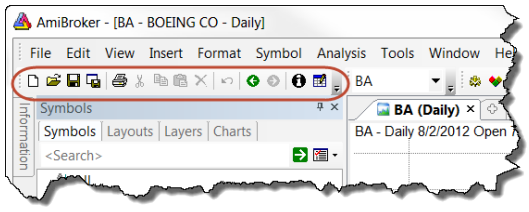
Window



HELP

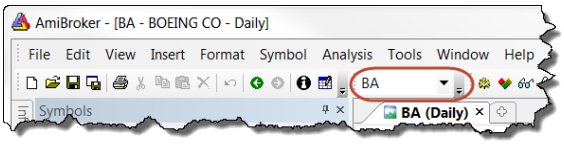


TOOLBAR



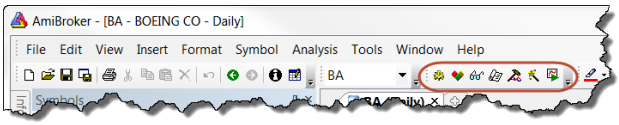
The Toolbar Menu contains shortcuts to File menu, plus other shortcuts. The toolbar can be customized. Use the pulldown menu at the right end of the toolbar to add or remove buttons.

Current Symbol



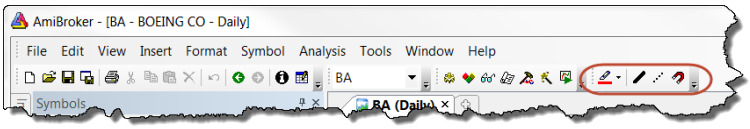
The Current Symbol specifies the symbol plotted in the top window, and the price series used by the backtester when Current is selected.

Shortcuts to Tools



Use these as an alternative to pulling down the tools menu.

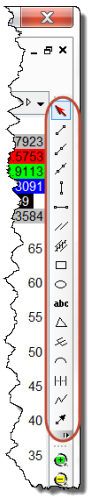
Shortcuts to Format

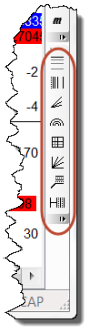


Use these as an alternative to pulling down the Format menu.

Drawing Tools

Use these to draw trendlines, symbols, text. The menus can be detached and redocked. (All of them can be.)

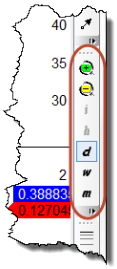




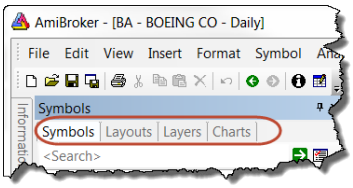
Zoom and Periodicity

Use the plus to view fewer bars, but in more detail. Use minus to view more bars.

Click one of D / W / M to display the price series in Daily, Weekly, or Monthly bars.

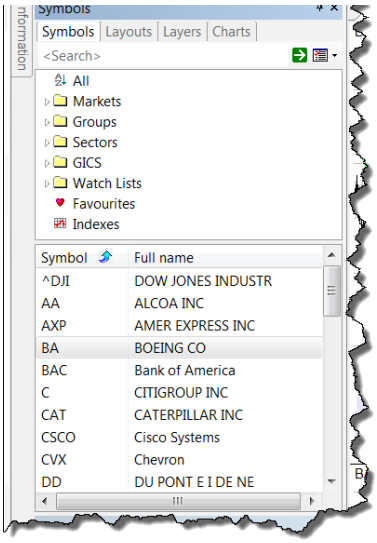


Tabbed Menus

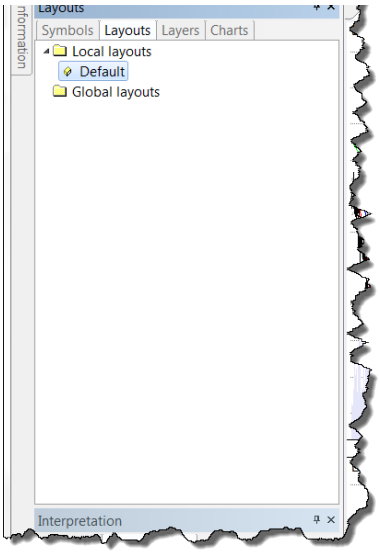


When selected, each tab displays the dialog associated with it in the tabbed menu window.

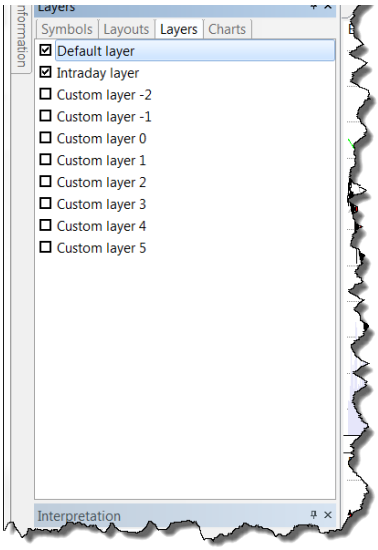
Symbols



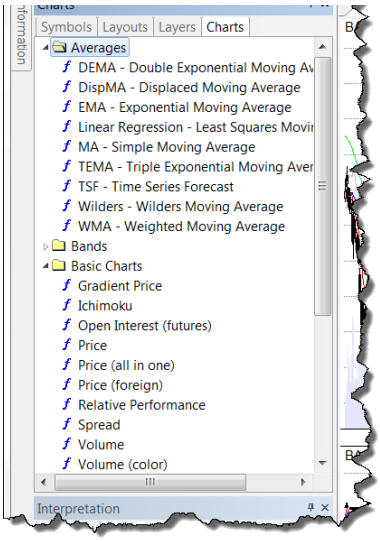
Layouts



Layers



Charts



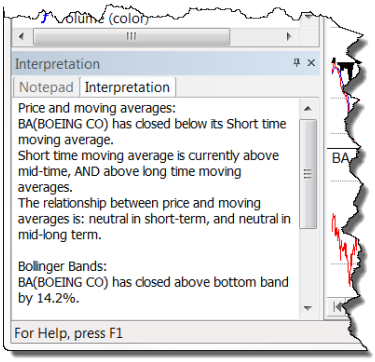
Use the Symbols menu to select the issue to be plotted.

Use the Layouts menu to save and retrieve chart layouts.

Use the Layers menu to select the layer the indicator is drawn on.

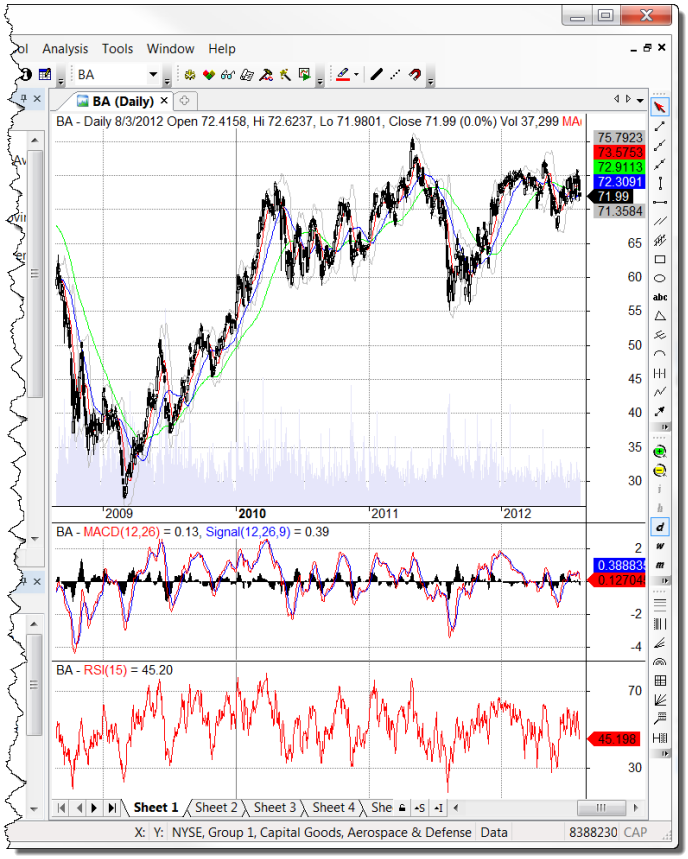
Use the Charts menu to select indicators to apply.

Interpretation Window



You can write your own commentary based on conditions you detect in the data.

Chart Area



Refer back to the Block Diagram in Chapter 4. The Chart Area has one or more sheets, each of which contains one or more panes. Each pane contains the display of one or more data series, tools, or indicators.

There are many options, preferences, properties, and parameters. See the *User's Guide* for details.

Chapter 6

Technical Analysis

Decisions to buy and sell anything - stocks, mutual funds, cars, houses, jewelry - are made for a variety of reasons and use a variety of decision making techniques. When it comes to securities, there are three broad categories of decision making:

- Fundamental analysis
- Chart analysis
- Quantitative analysis

Fundamental analysis uses information about the financial health of the organization, derived from financial statements, reports by the management, estimates of future revenue and expenses, and so forth. The efficient market hypothesis postulates that all the information that can be known about an organization has already been taken into account and is always already reflected in the price of its shares. Trading decisions are made using one of the models based on the market being efficient, such as the Capital Asset Pricing Model.

The two branches of technical analysis - chart analysis and quantitative analysis - believe that the markets are somewhat inefficient, and that there is information in the historical price and volume data that can be recognized and can lead to profitable transactions. For both, several things must be true in order to be profitable trading:

- The markets must be inefficient.
- There must be patterns in the price and volume that precede profitable trading opportunities.
- The patterns must occur often enough to be profitable enough to overcome trading costs.
- The analyst must be able to recognize those patterns, either visually or through computer programs.
- The patterns must persist long enough beyond the period being analyzed so that trades can be made.

Chart analysis uses visual examination of the price and volume searching for patterns such as head-and-shoulders, trendlines, support and resistance levels. Often, the recognition of important patterns is a subjective judgement of the analyst. As demonstrated in the earlier chapters, AmiBroker has an outstanding complement of charting tools to aid the chart analysts.

Quantitative analysis, used in the sense of analysis of price and volume information, relies on writing a set of rules into a computer program, then running that program as a component of a trading system development platform. Trading system programs have the same characteristics as general purpose computer programs, in that they have variables and constants, read and write data, can perform numeric calculations and string manipulation, have flow control statements, have subroutines and functions, and so forth. In addition, trading system programs have special features to deal with financial data and trading. They can process the data streams with the daily sets of open, high, low, close, and volume. They have special functions to handle the buying of shares, selling of shares, and accounting for the profits.

AmiBroker is a trading system development platform. It has special features and specialized functions to deal with the price data, the purchase and sale of shares (or whatever is being analyzed and traded), the accounting and reporting related to the trading account.

Of particular importance is a trading system development platform's ability to search for patterns (backtest), its ability to automatically search for the best sets of rules (optimize), and its ability to perform statistical tests to measure the likelihood of a set of rules being profitable in the future (validation). AmiBroker excels at all these tasks, and each will be discussed in the chapters to come.

Chapter 7

Trading System Development

All traders need confidence that their system will be profitable when traded. For any trader, the way to build confidence is to practice.

One advantage that traders who use mechanical systems have over those who use graphical methods is that the mechanical system is (or, at least, can be) objective, rule-based, judgement free, and testable.

You will be writing a trading system, testing it using historical data, then following the rules to place actual trades. The period of time, and the data associated with that time, that you use to develop and test your system is called the in-sample period and the in-sample data. The period of time, and the data associated with that time, that follows the in-sample period and has never been tested or evaluated by the system is called the out-of-sample data. Actual trades are always out-of-sample.

The process of validating a trading system is one of observing the profitability and behavior of the system in the out-of-sample period after it leaves the in-sample period.

The transition you make going from testing your system in-sample to trading it out-of-sample is one data point in the validation of your system. Your confidence level will be much higher if you have observed many of these transitions.

The ten step outline presented here is designed to build your confidence that your system will be profitable. For a more extensive discussion of trading system development, including expansion of the topics presented in this outline, please see *Quantitative Trading Systems*.

1. Define the Objective Function

An objective function is a metric of your own choosing that you use to rank the relative performance of two or more alternative trading systems. The phrase "of your own choosing" is critical. Your objective function must accurately reflect what is important to you. If you prefer long holding periods and infrequent trading, your objective function must rate systems that hold for weeks to months higher than systems that hold for a few days. If you prefer high equity gain without regard to drawdown over lower, but smoother, gain your objective function must reflect that.

I believe the psychology of trading experts who try to help traders become comfortable with the systems they trade have it backwards. If you decide ahead of time what you want, and design trading systems that satisfy your wishes, and if you have the confidence built through the validation process, you are guaranteed to be comfortable with your system.

AmiBroker reports the score for many metrics with each test run. If one of these standard, built-in metrics meets your needs, use it. If you want something else, you can create a custom metric and use it. For many people, rewarding equity growth while penalizing drawdown is important. If you agree, consider using one of the standard metrics that does that: RAR/MDD (risk adjusted annual rate of return / maximum drawdown), CAR/MDD (compound annual rate of return / maximum drawdown), K-ratio, Ulcer Performance Index, RRR (risk-reward ratio).

2. Decide What to Trade and How to Trade It

If you have a day job, you probably want to use daily data, run your trading system in the evening, and place your trades the next day. Or, you may be able to watch the market during the day and want to place your trades intra-day or at the close. You may have a preferred group of stocks or mutual funds. Be certain your system works well with your choices.

3. Design the Trading System

A trading system is a combination of a model and a set of data. The model is contained in the AFL code you write. The data is the price data of the ticker symbol your code processes. The model contains the intelligence. It is looking for the patterns in the data and testing the profitability of buying and selling.

A model consists of several parts: filter or setup, entry, exit, trailing stop, position size, portfolio composition, and so forth. In much of the literature, the entry is emphasized. But the other components are more important.

A model is a static representation of a dynamic process. Once you are done coding and testing, the model does not change. It may be cleverly designed and have self-adapting parameters, but it is still static. The market being modeled is dynamic and ever changing. Your model is looking for a particular pattern or set of conditions, after which it expects a profitable trade. As long as the model and the market remain in synchronization, the system will be profitable. When the two fall out of sync, the system will be less profitable or unprofitable.

It is useful to think of the data as comprised of two components - signal and noise. The pattern your model is looking for is the signal portion of the data. Everything your model does not recognize is the noise portion of the data. Your goal, as a designer of trading systems, is to accurately recognize the signal and ignore the noise.

Our hope is that:

- We can build a model,
- That recognizes some inefficiency,
- And use that model to trade profitably,
- As long as the model and reality stay in sync.

4. Determine the Length of the In-Sample Period

There are two views about how much time and data should be used to develop the system.

Some want a long time. They feel that the system will experience a variety of conditions and be better able to handle changes in the future. The risk is that conditions vary so much in the in-sample period that the system will not learn any of it well.

Some want a short time. They feel that the system will be better able to synchronize itself and learn very well. The risk is that the system may learn a temporary pattern that does not persist beyond the in-sample period. Another way to view this is that the system has fit itself to the noise because the signal is too weak. A system that has fit to the noise, performs well in-sample, but does not perform well out-of-sample is sometimes described as a system that is curve-fit, or over-fit, to the data.

The proper length of the in-sample period is impossible to state in general. It is very much a function of both the model and the data. The only way to determine the length of the in-sample period is to run some tests.

5. Determine the Length of the Out-Of-Sample Period

The length of the out-of-sample period is: As long as the model and the market remain in sync and the system remains profitable. There is no general relationship between the length of the out-of-sample period and the length of the in-sample period.

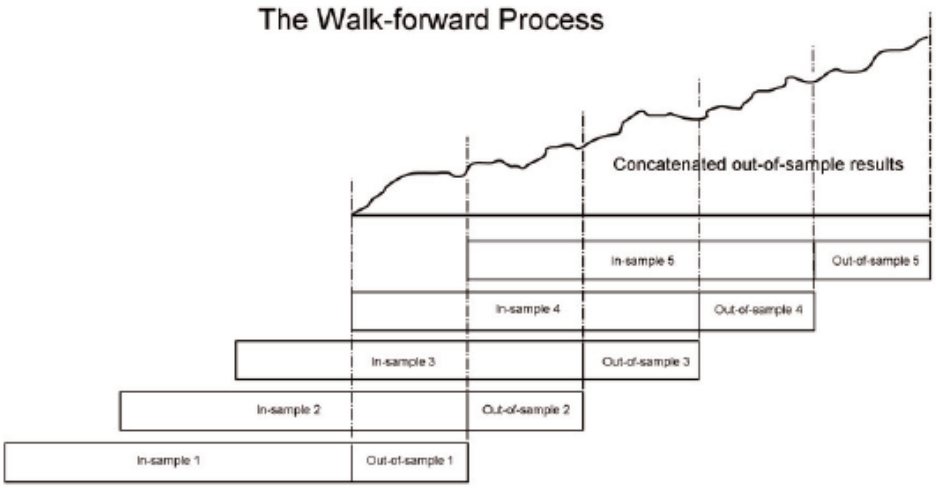
6. Decide What to Optimize

To optimize means to search a lot of alternatives and choose the best of them. Optimization is not a bad thing; in fact, it is a necessary step in system development. If you were to think up a new trading system and write the AFL code for it, it would include logic and parameters that were first guesses. You could test that system as it stands, and then either trade it or erase it and start over. But you are unlikely to do that. More likely, you will try other logic and other parameters values. If you are trying a few hand-picked alternatives, you might as well run an optimization and try thousands of alternatives. Give yourself a chance to find the best system.

Anything you would consider changing in your system - a different parameter value or an alternative logic statement - is a candidate to be optimized.

7. Perform Walk Forward Runs

While there is nothing special about optimizing, there is something very special about a walk forward run. You saw these diagrams in Chapter 3, Example 9, but they are worth looking at again.



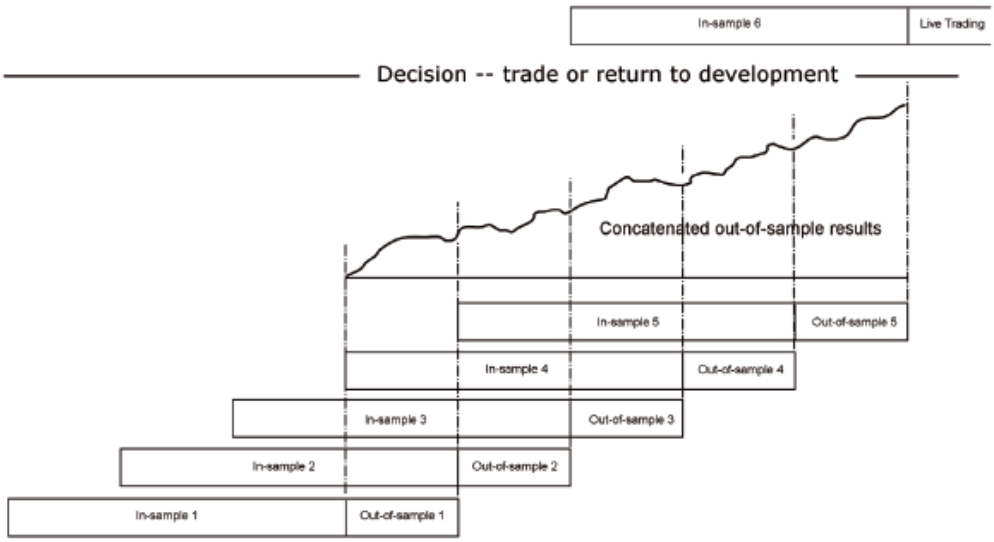
The walk forward process is several iterations of:

1. Optimize in-sample.
2. Choose the best according to your objective function.
3. Use those values and test out-of-sample.
4. Step forward by the length of the out-of-sample period.

Continue this process until you have used the last full in-sample period.

8. Evaluate the Out-Of-Sample Results

Either trade the system or send it back to development.



Note the importance of having an objective function you trust. The set of parameters used to test out-of-sample are the parameters that are at the top of the results list, after the list has been sorted by your objective function. The process is automatic and objective. All the decisions were made in earlier steps. You will never even see what the second choice is; the first choice is always used.

Each walk forward step is one data point you will use in the validation of your system.

9. Trade the System

Using the set of parameters that are at the top of the list after the last optimization, buy and sell when the system gives signals.

Take all the signals. If you have some way to decide which signals to take and which to skip, that logic belongs in the trading system and should go through the validation process.

On the last day of what would have been the out-of-sample period, re-optimize. Pick the top-ranked parameter values and continue to trade.

10. Monitor the Results

Each trade signaled after development has finished, and each trade you actually take, is an out-of-sample trade. You can compare the statistics for your trades with statistics for the out-of-sample results from the walk forward runs. If your results drop below what is statistically expected, stop trading the system. Either paper-trade it to see if it recovers; or re-optimize ahead of schedule, then paper-trade it and observe its performance.

Refer to *Modeling Trading System Performance* for detailed discussion of determining whether the system is working or broken, and for techniques for computing the best position size in order to maximize account growth while holding drawdown to a level within the tolerance of the trader.

In Summary

There are no guarantees. The best we can hope for is a high level of understanding and confidence gained through the validation process.

Chapter 8

AFL - AmiBroker Formula Language

AFL is AmiBroker Formula Language - the code that takes your ideas and puts them in the form that AmiBroker understands. Formulas, indicators, explorations, and trading systems are all written in AFL.

AFL is a component of AmiBroker, and is installed automatically in all versions of AmiBroker.

AFL is a procedural programming language, similar to Visual Basic, C, C++, Pascal, and Fortran. As such, it has language features for:

- Computation of numerical expressions.
- Evaluation of logical expressions.
- Declaration and use of data storage.
- Definition of data structures.
- Reading data into the program.
- Writing data out from the program.
- File manipulation.
- Program control.
- Built-in and user-defined functions.

AFL could be used for general purpose computing. But its unique capabilities come from the additional features designed to work with financial time series and trading systems. These include:

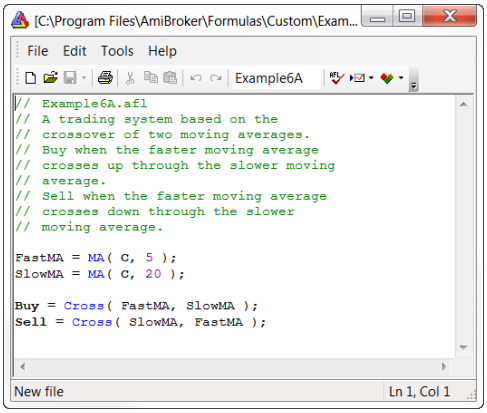
- Language and procedures to deal with financial quotations, such as open, high, low, close, volume, and open interest.
- Language and procedures to deal with trading system orders, such as buy, sell, short, and cover.
- The ability to plot or chart financial data, such as price, technical indicators, and buy and sell arrows.
- Powerful array processing capabilities. Much of the data used in trading systems is organized in time-ordered series, which AmiBroker stores in arrays and manipulates with array-oriented procedures.

AFL files are clear-text files, not proprietary or encoded in any way. While any text editor can be used to create and modify them, the AFL Formula Editor is designed specifically to work with AFL files. This chapter begins with a discussion of the AFL Editor, the Formula Editor.

Following that is a description and discussion of the AFL Language, including the language features.

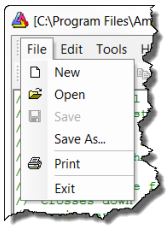
AFL Editor (Formula Editor)

The AFL Editor, or Formula Editor, is the editor that is built-in to AmiBroker. It is used to write AFL programs that will display custom indicators and test trading systems. It has four pull-down menus, a toolbar, and a context menu of its own.



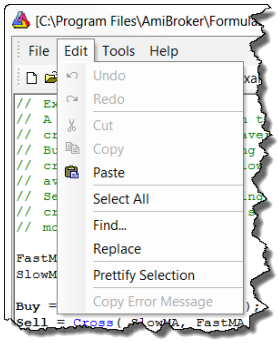
Pull-down menus

File Menu



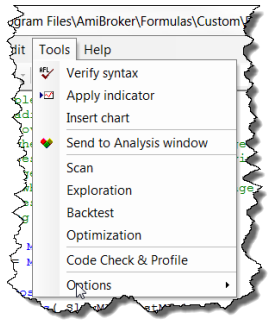
- New - Clears the AFL window
- Open - Opens an existing AFL file
- Save - Saves the contents of the window under the current name
- Save As - Saves the contents under a new name
- Print - Opens the Print dialog box to send the AFL to a printer
- Exit - Close the editor

Edit Menu



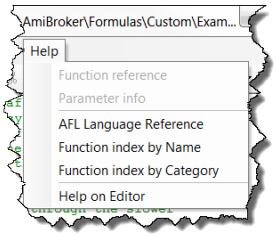
- Undo - un-does last action (multi-level)
- Redo - re-does recent action (multi level)
- Cut - Moves selected text to the clipboard
- Copy - Copies selected text to the clipboard
- Paste - Copies clipboard to cursor location
- Select All - Selects the entire file
- Find - Opens search dialog
- Replace - Opens search and replace dialog
- Prettify Selection - Reformats selected text to a consistent standard; adjusts indents and spacing
- Copy Error Message - After checking AFL syntax, if there is an error, copies that error message to the clipboard

Tools Menu



- Verify Syntax - Check AFL code for errors
- Apply Indicator - Saves the file, then applies it one time. (If you click again, no new pane is opened.)
- Insert Chart - Saves the file, then applies it many times. (If you click again, another new pane is opened.)
- Send to Analysis window - Saves the file and makes it the selected file for Analysis.
- Scan - Saves the file, makes it the selected file for Analysis, and runs the Scan process
- Exploration - Saves the file, makes it the selected file for Analysis, and runs the Explore process
- Backtest - Saves the file, makes it the selected file for Analysis, and runs the Backtest process
- Optimization - Saves the file, makes it the selected file for Analysis, and runs the Optimize process
- Code Check & Profile - Saves the file, makes it the selected file for Analysis, and runs the Code Check and Profile process. Code Check tests for "future leaks" where the code peaks into the future. Profile lists the operations required to run the code and estimates the execution time.
- Options - There is only one option - Autosave formula. When checked, the file will be saved before being sent to Analysis for processing.

Help Menu



- Function Reference - Displays the AFL reference page for the function highlighted. Pressing the F1 key is equivalent.
- Parameter info - Display a Tooltip containing the complete function prototype, including parameters.
- AFL Language Reference - Displays the AFL Language Reference section of the User's Guide, describing AFL in general.
- Function index by name - Displays the available AFL functions, sorted alphabetically.
- Function Index by Category - Displays the available AFL functions, grouped by category.
- Help on Editor - Displays User's Guide entry for the AFL Editor

AFL Language

Arrays

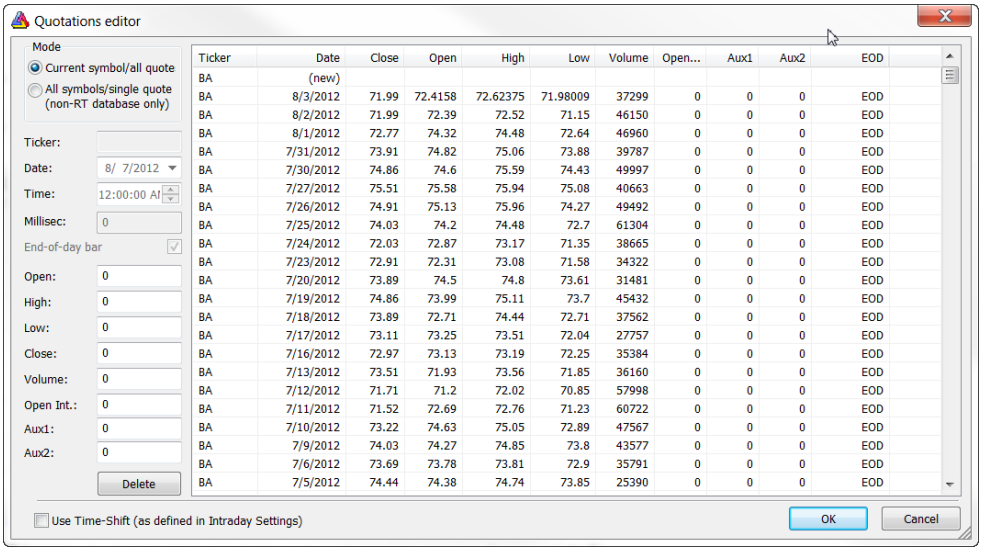
In computer science an array is a data structure consisting of a group of elements that are accessed by indexing. Each element has the same data type and the array occupies a contiguous area of storage.

Arrays are so important to AmiBroker that they will be introduced first - before other data structures and before the general description of the AFL language.

In AmiBroker, the Open, High, Low, Close, Volume, and Open Interest are stored in arrays. All other formulas and indicators that are used must be computed using the OHLCVI data.

Take a look at the data. Make Boeing, BA, the selected issue. Its price chart will appear. Using the **Symbol** menu, select **Quote Editor**.

There is a row for each bar and a column for each data field. When plotted on a chart, the date axis has the oldest at the left of the chart and most recent at the right.

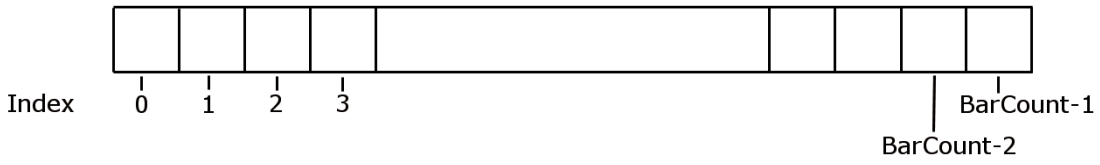


Ticker	Date	Close	Open	High	Low	Volume	Open...	Aux1	Aux2	EOD
BA	(new)									
BA	8/3/2012	71.99	72.4158	72.62375	71.98009	37299	0	0	0	EOD
BA	8/2/2012	71.99	72.39	72.52	71.15	46150	0	0	0	EOD
BA	8/1/2012	72.77	74.32	74.48	72.64	46960	0	0	0	EOD
BA	7/31/2012	73.91	74.82	75.06	73.88	39787	0	0	0	EOD
BA	7/30/2012	74.86	74.6	75.59	74.43	49997	0	0	0	EOD
BA	7/27/2012	75.51	75.58	75.94	75.08	40663	0	0	0	EOD
BA	7/26/2012	74.91	75.13	75.96	74.27	49492	0	0	0	EOD
BA	7/25/2012	74.03	74.2	74.48	72.7	61304	0	0	0	EOD
BA	7/24/2012	72.03	72.87	73.17	71.35	38665	0	0	0	EOD
BA	7/23/2012	72.91	72.31	73.08	71.58	34322	0	0	0	EOD
BA	7/20/2012	73.89	74.5	74.8	73.61	31481	0	0	0	EOD
BA	7/19/2012	74.86	73.99	75.11	73.7	45432	0	0	0	EOD
BA	7/18/2012	73.89	72.71	74.44	72.71	37562	0	0	0	EOD
BA	7/17/2012	73.11	73.25	73.51	72.04	27757	0	0	0	EOD
BA	7/16/2012	72.97	73.13	73.19	72.25	35384	0	0	0	EOD
BA	7/13/2012	73.51	71.93	73.56	71.85	36160	0	0	0	EOD
BA	7/12/2012	71.71	71.2	72.02	70.85	57998	0	0	0	EOD
BA	7/11/2012	71.52	72.69	72.76	71.23	60722	0	0	0	EOD
BA	7/10/2012	73.22	74.63	75.05	72.89	47567	0	0	0	EOD
BA	7/9/2012	74.03	74.27	74.85	73.8	43577	0	0	0	EOD
BA	7/6/2012	73.69	73.78	73.81	72.9	35791	0	0	0	EOD
BA	7/5/2012	74.44	74.38	74.74	73.85	25390	0	0	0	EOD

When the data is read from the database into memory, an array is created for each field - Date, O, H, L, C, V, OI. The array for each of them has one element for each bar. The elements are referenced (identified) by their position in the array - their index. The oldest element is stored in the first array element and has index 0. Each newer bar (newer element) has an index number 1 greater than its left neighbor. The set of all seven data fields for a given bar form a data record. All elements of a record have the same index number. When using end-of-day data, each bar holds the data for one day; the number of bars and the number of days will be the same.

BarCount

Look at the entire array, say for the closing price. When the data is loaded into memory, the number of data points is counted. That number is stored in a special variable named BarCount. The elements of the Close array are numbered from 0 to BarCount-1. There is no element whose index is BarCount. Trying to refer to an element whose index number is less than 0, or BarCount or greater, results in an "index out of bounds" error.



A total of BarCount elements, indexed from 0 to BarCount-1

When the price series is loaded from the database to memory, that established not only the length of those OHLCV arrays, but of all arrays. The default number of elements in every array is BarCount. When you compute an array indicator, such as the simple moving average, the resulting array also has BarCount elements; and they are also indexed from 0 through BarCount-1. When you program AFL and use looping, you will need to be aware of the array indexing and of the BarCount variable.

Array Operations

In order to use an indicator, like the 25-day moving average, the value of that average must be computed for each day.

The average is the sum of the 25 days up to (earlier than) and including the day being processed, divided by 25. Beginning with index number 24, every day has a 25-day moving average. For elements 0 through 23 there is not enough data to compute a 25 day average.

AmiBroker is unique in that it uses a very efficient array-processing technique to compute the 25-day moving average, giving the value for all the data elements in a single array in a single processing step. Computed once, they are available whenever they are needed for further analysis.

The formula to compute the moving average and store it so it can be used is:

```
MyAverage = MA ( C, 25 );
```

- MA is the AFL function that computes the simple moving average.
- C, or Close, tells the function to take the moving average of the Closing price.
- 25 is the number of bars (days) in the average.
- MyAverage is the location where the result is stored.

The key point is that this is an array operation. Close and MyAverage are arrays, and MA is a function that works with arrays (and only with arrays).

When people get confused with AFL, it is often over whether a function works with arrays or scalars (single values).

The description of MA, the function that computes the simple moving average, called the syntax of the function, tells us what it uses as input and what it returns as results.

Syntax:

MA (ARRAY, *periods*);

Returns:

ARRAY

Description:

Calculates a Simple Moving Average of length *periods* of ARRAY. Returns the result as an array.

For example:

```
MAofClose = MA ( Close, 20 );
```

Close is an array. MA is an array operation that computes the moving average for every element in the Close array. The result is also an array. The result returned from the function must be stored in a variable that is an array. All of these arrays have the same length - the length that was established when the data was read from the database and stored in the Close array.

Language Basics

Keep in mind that this is an introductory explanation. There are features of AmiBroker and AFL that are not explained, or even mentioned, here. For more information, please see the AmiBroker AFL Reference, which is a component of the AmiBroker User's Guide.

An AFL program consists of the following elements:

Tokens are word-like units recognized by the AFL language processor.

Whitespace is the collective name given to spaces, tabs, line breaks, and comments. Whitespace serves to separate tokens. Excess whitespace is discarded. That means that if a single whitespace character is permitted, you can have any number of them, including extra line breaks, to aid program readability.

Comments are text used to document a program. They are for the programmer's use only - they are stripped out of the code before it is processed. There are two ways to denote text as being a comment:

- // - two adjacent slashes - the remainder of the line is a comment.
- /* */ - two, two-symbol pairs - everything between the opening /* and the closing */ is a comment.

Comments may not be nested.

Tokens. AFL recognizes five classes of tokens:

- identifiers
- constants
- string-literals
- operators
- punctuators

Identifiers are arbitrary names, of any length, given to variables and functions. Identifiers must begin with a letter, may contain letters (a-z, A-Z), digits (0-9), and the underscore ("_"). AFL identifiers are not case sensitive. The identifiers MyMovingAverage and mymovingaverage are treated as being the same.

Constants are tokens representing fixed numeric or character values. Numeric constants consist of decimal integer and, optionally, decimal point and decimal fractional part. Negative numeric constants are preceded by the minus sign.

String literals, or string constants, are a sequence of any characters surrounded by double quotes ("). "This is a string literal"

Operators are the arithmetic, relational, and assignment operators, such as +, -, <. They will be covered in more detail below.

Punctuators are any of the following characters: ( ) , ; = .

Terms are constants, variables, or other terms that are operated upon by operators.

Expressions are combinations of constants, variables, terms, and operators that can be evaluated and result in a single value -- a scalar (single number), string, or array.

Statements consist of one or more expressions, an assignment operator, and a variable. The expressions are evaluated and assigned to the variable. Each statement must be terminated by a semicolon.

Operators

Comparison operators are used to test the relation between two expressions. The result of the test will be either True (1) or False (0). The operators are:

- < Less than
- > Greater than
- <= Less than or equal to
- >= Greater than or equal to
- == Equal to
- != Not equal to

Assignment operator is used to store the value on the right side into the variable on the left side. The operator is the = character. For example:

B = A + 14;

The right hand side is evaluated to a single result by adding A and 14 together. The result is stored in the variable B.

Note the difference between "=" and "==". The single = is an assignment. The double == tests equality.

Arithmetic operators combine one or more terms to produce a term or expression. They include:

- + Addition
- - Subtraction (including unary minus)
- * Multiplication
- / Division
- % Modulus (remainder)
- ^ Exponentiation (raise to a power)
- | Bit-wise Or
- & Bit-wise And

Logical operators work with logical constants and logical variables. They include:

- NOT (also !) Logical Not of a single operand
- AND (also &&) Logical And of two operands
- OR (also ||) Logical Or of two operands

Logical constants are True and False.

- **False** is represented by numeric zero .
- **True** is represented by numeric 1. Actually any non-zero value, positive or negative, evaluates to True.

Compound assignment operators perform an arithmetic operation followed by an assignment operation. The most commonly used are:

- ++ Post increment (increment the variable and store it back) i++ means add 1 to i. More completely, it means take the value of i, add 1, assign the result to i.
- -- Post decrement (decrement the variable and store it back) i-- means subtract 1 from i.
- += Add the value of the operand (variable) on the left plus the value of the operand on the right, then store the result back in the variable on the left. j += 2; Add 2 to j.
- -= Subtract the value of the operand on the right from the value of the operand (variable) on the left, then store the result in the variable on the left. k-= 5; Subtract 5 from k.
- *= Multiply the value of the operand (variable) on the left by the value of the operand on the right, then store the result back in the variable on the left. m *= i; Multiply m by i.

- /= Divide the value of the operand on the left (variable) by the value of the operand on the right, then store the result back in the variable on the left. x /= y; Divide x by y.

When in doubt, write out the calculation you want using ordinary arithmetic operators. The possible confusion is not worth the execution efficiency.

Operator Precedence

Parentheses can be used to control the order in which the operations take place. Those operations that are in the innermost parentheses are done first, then working outward. When there are no parentheses to explicitly define the order, this precedence is followed (highest ranked are listed first and done first):

1. ++ Post increment
2. -- Post decrement
3. [] Array element operator
4. ^ Exponentiation
5. - Unary minus
6. * Multiplication
7. / Division
8. % Modulo
9. + Addition
10. - Subtraction
11. < Less than
12. > Greater than
13. <= Less than or equal to
14. >= Greater than or equal to
15. == Equal to
16. != Not equal to
17. & Bit-wise And
18. | Bit-wise Or
19. NOT Logical Not
20. AND Logical And
21. OR Logical Or
22. = Assignment
23. *= Compound assignment (all operators)

In addition to having ++ be a post increment operator, it can be a pre increment operator. j = i++ uses ++ as a post increment operator. j is assigned the value of i, then i is incremented. j = ++i uses ++ as a pre increment operator. i is incremented, then the new value of i is assigned to j.

Use parentheses to insure that the calculations are carried out in the order you intend.

If H is 110 and L is 106, (H+L)/2 is 108. H+L/2 is 163, because the division takes place before the addition.

Array Subscript Operator

An entire array is referenced by using its name without qualification, as in MM = O;

To reference a particular element of an array, use the [] operator. For example, to reference the most recent closing price, use C[BarCount-1].

BarCount, as you will recall from the earlier discussion, is the number of elements in the OHLCV arrays, and all other arrays derived from those. The first element of every array is index 0, the final element is index BarCount-1.

Compound Statements (Blocks)

A compound statement consists of zero or more statements within curly brackets {}. A compound statement can be used anywhere a statement can be used.

Built-in Functions

In addition to the arithmetic operators, AmiBroker has over 70 built-in functions that perform mathematical operations. Functions accept zero or more arguments enclosed in parentheses, and return zero or one values.

Conditional If

The conditional if function has the form IIF(exp, Tp, Fp); Note that IIF is a function - it returns a value. The expression exp is evaluated to either True or False. If True, return Tp, if False, return Fp.

Note that IIF is Not a program flow control statement. That is, it does not operate like the traditional IF statement. IIF is a function and it returns a value.

Variables

A variable is an identifier and its associated data structure and storage.

- You can define and use as many variables as you wish; there is no limit.
- Each variable must be assigned a value before it can be used further.

- Variables used in the main program cannot be assigned values in functions.
- User-defined variables may not duplicate names already defined by built-in functions (such as EMA), predefined arrays (such as Open), keywords (such as If), or reserved words (such as Buy).
- It is good programming practice to define variables, give them meaningful names, assign values to the variables, then use the variables in subsequent calculations and function calls.

Reserved Variables

AmiBroker uses some variable names and reserves those. You may not use these identifiers for variables you define:

- buy
- sell
- short
- cover
- buyprice
- sellprice
- shortprice
- coverprice
- title
- tooltip
- graphxspace
- graphzorder
- exclude
- roundlotsize
- ticksize

User-Defined Functions, Procedures, Scope

User-definable functions make it possible to encapsulate user code into easy-to-use modules that can be used in many places without the need to copy the same code over and over again.

Functions must have a definition. The function definition includes the function body — the code that executes when the function is called.

The function definition establishes the name and parameters of a function. The function must be defined prior to the call to the function. The definition is a single line of code that begins with the keyword *function,* is followed by the function name, and then the list of

parameters enclosed in parentheses. The function body, enclosed in curly braces, follows using as many statements as you wish.

When the function is invoked or called by the calling program, control passes to the called function. Variables are passed for the arguments. Each argument corresponds to a parameter. The function performs its calculations, including executing the *return* statement. If there is a result to be returned, it is passed back to the calling program.

If the function does not have any return statement (does not return anything) then we call it a procedure.

Following is an example of a function, a 2nd order smoother:

```
// the first line is the definition of the function
function IIR2( input, f0, f1, f2 )
// the remaining lines are the body of the function
{
    result[ 0 ] = input[ 0 ];
    result[ 1 ] = input[ 1 ];

    for( i = 2; i < BarCount; i++ )
    {
       result[ i ] = f0 * input[ i ] +
                     f1 * result[ i - 1 ] +
                     f2 * result[ i - 2 ];
    }
   // this function returns an array
   return result;
}

// this is the main, or calling, program
Plot( Close, "Price", colorBlack, styleCandle );
Plot( IIR2( Close, 0.2, 1.4, -0.6 ), "function example", colorRed );
```

IIR2 is a user-defined function. There are four formal parameters: input, f0, f1, f2.

The calling program prepares actual arguments, one argument corresponding to each format parameter. At the time of the function call the values of arguments are passed (by value, not by name or by reference). Formal parameters behave like local variables and have the numeric values that were passed to them. They are undefined (have no numeric values) until the function is called. And when the return statement is executed, they become undefined again. They have no memory.

Within the body of the function, local variables named *result* and *i* are defined and used. Local variables are visible inside the function only. If any other function uses the same variable name the two will not interfere with each other.

Local, Global

AFL does not require that variables be declared before they are used. The first use of a variable determines whether it is *local* or *global*.

If a given identifier appears first inside a function, then it is treated as a local variable to that function.

If a given identifier appears first outside the function, but is used within the function, then it is treated as a global variable and it carries whatever value it had at the time of the call into the function.

Since this can cause problems that are difficult to debug, there are two keywords, local and global, that can be used to force the scope. See the following example.

```
k = 4; // this is a global variable because of its first use

function f( x )
{
   z = 3; // this is LOCAL variable
   return z * x * k; // 'k' here references global variable k
                     // (first used above outside function)
}

z = 5; // this is GLOBAL variable with the same name
       // as the local variable in function f

"The value of z before function call :" + WriteVal( z );

// Calling the function will not affect the
// the value of our global variable z
// because the global variable z and
// local variable z are separate and
// arguments are passed by value (not by reference)

"The result of f( z ) = " + WriteVal( f( z ) );

"The value of z after function call is unchanged : " + WriteVal( z );
```

Example 2: Using local and global keywords to override default visibility rules:

```
VariableA = 5; // implict global variable
function Test()
{
   local VariableA;  // explicit local variable
                     // with the same identifier as global
   global VariableB; // explicit global variable
                     // not defined earlier
                      // may be used to return more
                     // than one value from the function

   VariableA = 99;
   VariableB = 333;
}

VariableB = 1; // global variable

“Before function call”;
“VariableA = “ + VariableA;
“VariableB = “ + VariableB;

Test();

“After function call”;
“VariableA = “ + VariableA + “ (not affected by function call )”;
“VariableB = “ + VariableB + “ (affected by the function call )”
```

At the end of the function we can see the *return* statement that is used to return the result to the caller. Currently the return statement must be placed at the very end of the function.

It is also possible to write a procedure (a function that returns nothing (void))

```
procedure SinePlotter( Freq, ColorIndex )
{
   pname = “Line”+WriteVal(ColorIndex,1.0);
   array = sin( Cum( Freq * 0.01 ) );
   Plot( array, pname , colorRed + ColorIndex, styleThick );
}

for( n = 1; n < 10; n++ )
{
    SinePlotter( n/2+Cum(0.01), n );
}
```

Although there are two separate keywords *function* and *procedure,* AmiBroker currently treats them the same (they both accept return values but do not require them), but in the future the rules might be enforced. That will require use of the return statement only in

functions. So it is advisable to use the function keyword in the case where your function returns any value and the procedure keyword if it does not.

Note that recursion (having a function call itself from within itself) is not presently supported.

Flow Control

If you are using the array-oriented features of AFL, then most of the code you write will start at the top of the program and flow to the bottom of the program without interruption. None of the flow control statements or tests accept arrays as arguments. The flow control statements are:

- do ... while
- for
- if ... else
- switch
- while

There are two typical uses for flow control statements:

1. You want to perform some task that may or may not operate on entire arrays for each of several values of a variable. For example, write to a file the results of running tests with values for a particular argument of 5, 7, 9, and 11.
2. You want to step through the data that is in one or more of the arrays, such as the array of closing prices, and make some decision based on what you find. To do this requires that the program use looping.

Each of the five flow control statements will be illustrated with an example. The optional statements that can be used with the flow control are also explained.

break

The break keyword is a part of switch statement and an optional part of looping for, do-while, and while statements.

The break keyword terminates the smallest enclosing do, for, switch, or while statement in which it appears.

The syntax is:

break;

The break statement is used to exit an iteration or switch statement. It transfers control to the statement immediately following the iteration substatement or switch statement.

The break statement terminates only the most tightly enclosing loop or switch statement. In loops, break is used to terminate before the termination criteria evaluate to 0. In the switch statement, break is used to terminate sections of code — normally before a case label. The following example illustrates the use of the break statement in a for loop:

```
i = 0;
while ( i < 10 )
{
  i++;
  // break at step 5
  if( i == 5 )
  {
    break;
  }
  printf("Step " + i );
}
```

continue

The continue statement is an optional part of for, do-while, and while statements.

It stops the current iteration of a loop, and starts a new iteration.

The syntax is:

continue;

You can use the continue statement only inside a while, do...while, or for loop. Executing the continue statement stops the current iteration of the loop and continues program flow with the beginning of the loop. This has the following effects on the different types of loops:

- while and do...while loops test their condition, and if true, execute the loop again.
- for loops execute their increment expression, and if the test expression is true, execute the loop again.

The following example illustrates the use of the continue statement:

```
i = 0;
while ( i < 10 )
{
  i++;
  // Skip 5
  if( i == 5 )
  {
    continue;
  }
  printf("Step " + i );
}
```

do ... while

The syntax (general form) of the do ... while statement is:

do
{ statements }
while (expression);

do and while are keywords. They are always spelled exactly the same way and used in the same way.

{ statements } is any executable statement or statements, enclosed in a set of curly brackets. The curly brackets are mandatory.

(expression) is any expression that can be evaluated to either True or False. The parentheses are mandatory. The final semicolon is mandatory.

When this part of the program is reached, the statements are executed in order. There may be flow control statements included. Eventually the end of the statement code will be reached.

The expression is evaluated. If the result is True, then the statements in the body will be executed again, starting from the beginning, with the variables retaining their most recent values. If the result is False, then the do ... while is finished and program flow continues to the statement following it.

```
// initialize control variables to prepare for the do ... while
length = 5;
// the do ... while
do
{
  printf (“ something interesting “ + NumToStr(length) + ”\n”);
  length = length + 2;
}
while (length <= 13);
// whatever code comes after the do ... while
```

The output of the printf will have lines for lengths of 5, 7, 9, 11, and 13.

for

The syntax of the for statement is:

for (initial; conditional; loop-expression)
{ statements; }

for is a keyword.

(initial; conditional; loop-expression) control the loop. Before the first execution of the statements in the body, initial is executed. Immediately, and before every execution of the body, conditional is tested. If the result of the test is True, execute the statements in the body of the loop. If the result is False, then program flow continues to the statement following the for. After each execution of the statements in the body, execute the loop-expression.

```
// prepare for the for loop
myema[0] = Close[0];
// the for loop
for ( i=1; i < BarCount; i++ )
{
    myema[i] = 0.1 * Close[i] + 0.9 * myema[i-1];
}
// whatever code comes after the for loop
Plot (C, “C”, colorBlack, styleCandle);
Plot (myema, “MyEma”, colorRed,styleLine);
```

if ... else

The syntax for the if ... else statement is:

if (expression)
    statement1
[**else**
    statement2]

or, more symetrically:

if (expression)
{
    statement1;
}
else
{
    statement2;
}

if and else are keywords. The portion in the [ ] in the first explanation is optional (the brackets are not meant to indicate an array index). The else is optional. The second explanation includes the else portion, which will have an empty statement if there is no action to take.

(expression) is evaluated. If it is True, statement1 is executed. If it is False and there is an else clause, statement2 is executed. If it is False and there is no else clause, no action is taken.

For example:

```
i = 1;
if (i > 1)
   printf (“i is big \n”);
else
   printf (“i is small \n”);
```

if ... else statements can be nested. For example:

```
i = 2;
j = 4;
if (i > 1)
   if (j > i)
         printf (“i is big \n”);
else
   printf (“i is small \n”);
```

The question immediately arises: “which if gets the else?” The best answer, from good programming practice, is to use a full complement of braces so the question never arises. The answer to the question as it stands is: the else goes with the closest if that does not already have an else. The spacing on the page does not enter into the discussion at all. Don’t let ambiguity into your programs - use a full set of braces. The fully braced equivalent to the example above is:

```
i = 2;
j = 4;
if (i > 1)
{
   if (j > i)
   {
         printf (“i is big \n”);
   }
   else
   {
         printf (“i is small \n”);
   }
}
else
{
}
```

Run each of these. (Use the Commentary window for quick and dirty testing.) First as they stand, then with j set to 0.

Important note. The values being compared in the expression part of the if ... else are scalars - single values. If ... else can only be used on scalars, not on arrays. You can test to determine whether H[i] > H[i-1]. But you cannot test whether H > Ref(H,-1). The first

is testing two single numbers, and the result is clearly either True or False. The second is testing two arrays, and the result is not defined.

switch

The syntax of the switch statement is:

```
switch (expression)
{
    case constant-expression1: statement;
    case constant-expression2: statement;
    ...
    case constant-expressionN: statement;
    default: statement:
}
```

Control passes to the statement whose case constant-expression matches the value of switch (expression). The switch statement can include any number of case instances, but no two case constants within the same switch statement can have the same value. Execution of the statement body begins at the selected statement and proceeds until the end of the body or until a break statement transfers control out of the body.

You can use the break statement to end processing of a particular case within the switch statement and to branch to the end of the switch statement. Without break, the program continues to the next case, executing the statements until a break or the end of the statement is reached. In some situations, this continuation may be desirable.

The default statement is executed if no case constant-expression is equal to the value of switch (expression). If the default statement is omitted, and no case match is found, none of the statements in the switch body are executed. There can be at most one default statement. The default statement, if exists, must come at the end. Otherwise it may be executed before hitting conditions defined below it. A case or default label is allowed to appear only inside a switch statement.

The type of switch expression and case constant-expression can be any. The value of each case constant-expression must be unique within the statement body, otherwise the first match will be used.

Example:

```
for( n = 0; n < 10; n++ )
{
printf("Current n = %f\n", n );
switch(n) {
    case 0:
      printf("The number is zero.\n");
      break;
    case 3:
    case 5:
    case 7:
      printf("n is a prime number\n");
      break;
    case 2: printf("n is a prime number\n");
    case 4:
    case 6:
    case 8:
      printf("n is an even number\n");
      break;
    case 1:
    case 9:
      printf("n is a perfect square\n");
      break;
    default:
      printf("Only single-digit numbers are allowed\n");
    break;
   }
}
```

while

The while statement lets you repeat a statement until a specified expression becomes false.

The syntax of the while statement is:

while (expression)
    statement;

The expression must have an arithmetic (numeric or logical) type. Execution proceeds as follows:

1. The expression is evaluated.

A. If the expression is initially false, the body of the while statement is never executed, and control passes from the while statement to the next statement in the program.

B. If the expression is true (nonzero), the body of the statement is executed and the process is repeated beginning at step 1.

This is an example of the while statement:

```
i = 10;
while( i < 20 )
{
  Plot( MA( Close, i ), “MA” + WriteVal( i, 0 ),
                colorBlack + i );
  i = i + 1;
}
```

The example plots 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 - bar moving averages.

AFL Function Reference

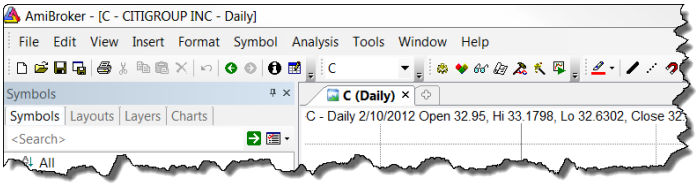
AmiBroker has several hundred functions available for you to use as you write your AFL programs. Rather than list them in this book, please see the *AmiBroker User's Guide*, AmiBroker Formula Language.

Chapter 9

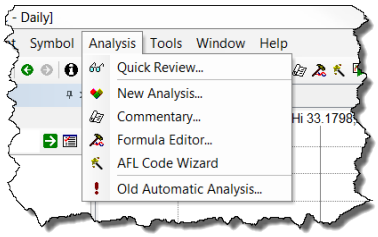
Analysis

The Analysis tab is host to the Analysis window and the major tools for writing, testing, refining, and validating trading systems; and for monitoring them for trading signals.

When you start AmiBroker, it opens the Chart window on the Chart tab.



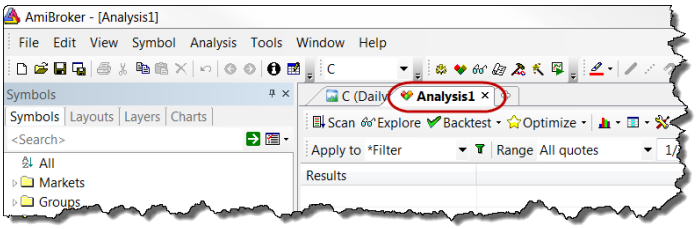
The Analysis pull-down menu looks like this:



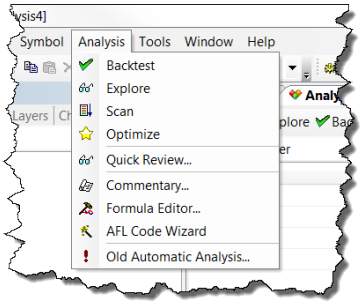
The Analysis window on the Analysis tab can be opened in several ways:

- Use the Analysis pull-down menu and click New Analysis.
- Use the File pull-down menu, then New > Analysis,
- Send a file to Analysis, for example from the Formula Editor.

The Analysis tab is displayed.

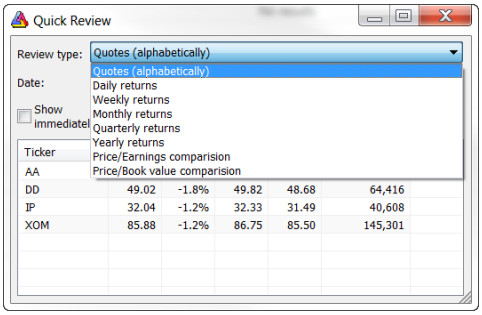


And the Analysis pull-down menu changes to this:

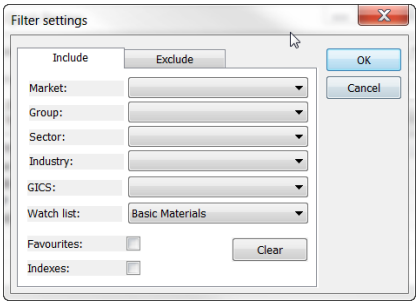


Quick Review

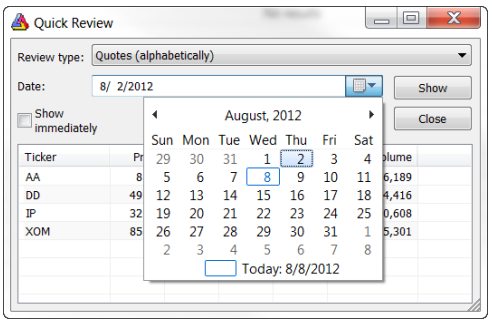
The Quick Review gives quick access to lists of daily price quotations and period-over-period price changes. The **Review Type** menu lets you select the type of report it will calculate.



Clicking the **Filter** button opens the Filter settings dialog. This allows you to specify the group of symbols that will be processed. There are two tabs, **Include** and **Exclude**, and they have the same controls. Using the pull-down menus for any of the categories, you can include whatever you want the report to show, then exclude whatever portion of those included that you do not want. This example shows including the watchlist containing the members of the Basic Industries.

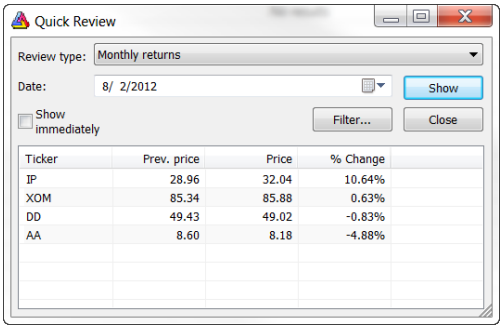


The **Date** menu opens a calendar tool that lets you select the report date - the date you want as the base date of the report. Or you can enter the date directly.



Quotes (Alphabetically) lists the closing price, day-over-day price change, high, low, and volume, as shown in the first figure in this section.

Daily, Weekly, Monthly, Quarterly, and Yearly Returns are all calculations of price change from one period ago to the report date. The report shows the price at the period earlier, the current price, and the percentage change. It is sorted into descending order by percentage change, but you can sort on any column by clicking the column header. Set the type report you want, then click the Show button. Here is an example of a monthly report:



Clicking the **Show** button starts the calculations that produce the report. Checking the **Show Immediately** box lets the results display as soon as they are calculated, rather than waiting until the entire report is complete before displaying results.

Getting results for the final two report type options:

- Price/Earnings comparison
- Price/Book value comparison

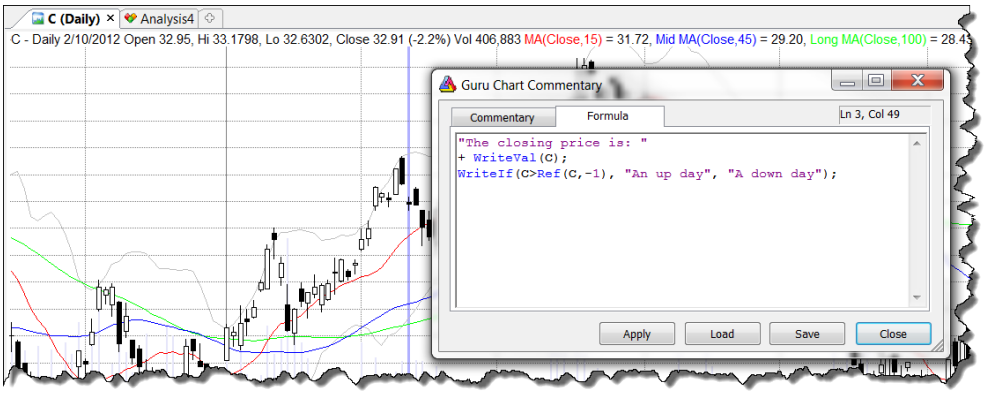
requires that your database has entries for earnings and book value.

Commentary and Interpretation

You can write commentary that will appear in the Commentary window or the Interpretation window. The commentary is specific to the active symbol, the active pane, and the selected bar. The explanation for Commentary is first, then Interpretation.

Commentary

Using the **Analysis** menu, select **Commentary**. This commentary window, called Guru Commentary, has two tabs - Commentary and Formula. Use the Formula tab to enter the AFL code for the commentary formula you want to use. The Commentary tab will display the output.



A commentary formula consists of two elements: static text and dynamic content.

Static Text

Static text elements are enclosed in quotes and terminated with a semicolon. The text is displayed in the commentary window exactly as it is written. Each statement will be placed on a new line. Including the two characters "\n" will force a line break in the output whenever that is necessary (and it will be necessary in Interpretation). A simple example of static text is:

```
"The closing price is: ";
```

Dynamic Content

Two AFL functions are useful for creating dynamic content: WriteVal() (or its equivalent function, NumToStr), and WriteIf().

WriteVal

Syntax:

WriteVal(NUMBER);

WriteVal(ARRAY);

Returns:

STRING

Description:

WriteVal does not really write anything; it converts either a number or an array to a string and returns the string value. WriteVal can only be used within a commentary.

Any expression that evaluates to a number or numeric array can be used as an argument to WriteVal. For example:

```
WriteVal(Close - Ref(Close,-1));
```

WriteVal always returns one single value of the array (not the entire array of values). In many cases this is the LastValue of the array, but in indicators it is "selected value" - the one that is selected by the vertical line.

WriteIf - Conditional Output

Syntax

WriteIf(Expression, "True Text", "False Text");

Returns

STRING

Description

The WriteIf function is very similar to the IIf function. An expression is evaluated. If the result is True, then whatever is in the True Text string is returned. If the result is False, then whetever is in the False Text string is returned. For example:

```
WriteIf(C>Ref(C,-1), “An up day”, “A down day”);
```

Concatenation means join together by appending one to the other. The concatenation operator is “+”. To join two strings together, put the “+” between them. Any expression that evaluates to a string can be concatenated into the output by using the “+” operator. For example:

```
“The closing price is: “ + WriteVal(Close);
```

The calculation and nesting can be as complex as you wish. For example:

```
Indicator1 = ...;
Indicator2 = ...;
Indicator3 = ...;
“The indicator status today is “ +
WriteIf(Indicator1>3, “Very Bullish”,
WriteIf(Indicator2>12, “Bullish”,
WriteIf(Indicator3<0, “Bearish”, “Neutral” )));
```

Use

Commentary code is written by typing it into the Formula tab of the Commentary window; no editor is needed. The **Save** button opens a Save As dialog and you can save the commentary just as you would save any other AFL program.

If you have already written a commentary, clicking the **Load** button opens a dialog box so you can locate and load the program you want to use.

Click the **Apply** button to run the code. Click the **Results** tab to see the commentary that applies to the bar that is selected.

Interpretation

To convert the code you have written for the commentary window so that it appears in the Interpretation window, enclose every string that is being written so that it becomes an argument to a Printf function.

Syntax

printf(formatstr);

Returns

Nothing

Description

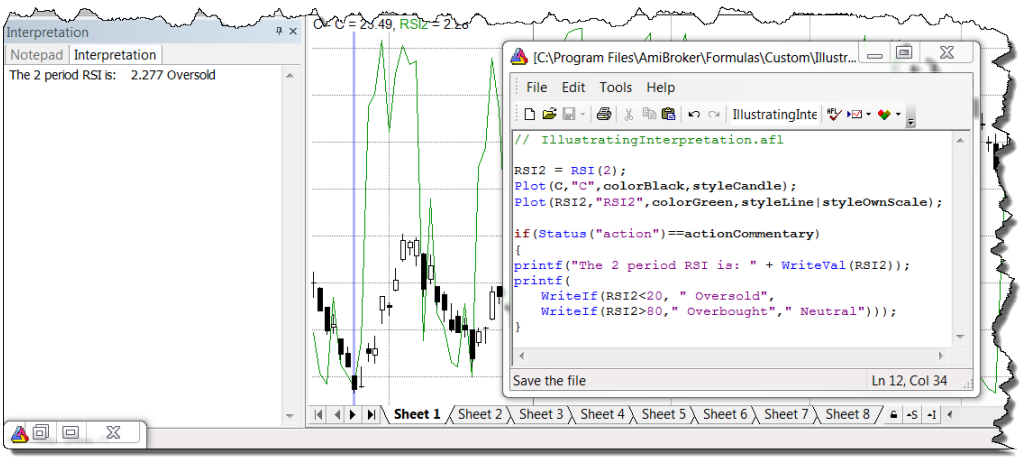
Printf sends the string portion of formatstr, formatted according to the format portion of formatstr, to the output.

In this case, the output is the Interpretation window. The code must be appended to the AFL code of the system or function that is displayed in a pane. In order to see the output, make the pane active. The commentary output will appear in the Interpretation window. As bars are selected, the output will be interpreted for each bar.

In the image that follows:

A. Add the commentary code to the AFL for a trading system that has a plot statement.
B. Use the Apply Indicator icon to open a new pane and plot the indicator.
C. Click in the pane that has the commentary code attached to it.
D. Read the commentary in the Interpretation window.

If you click a bar, you will see the interpretation for that bar. If you click a different pane, you will see the commentary associated with that pane.



Performance Tip

To get the best performance, enclose the commands that compute the values used only by the commentary and the statements that produce the output in a set of braces and precede them by a conditional test. For example:

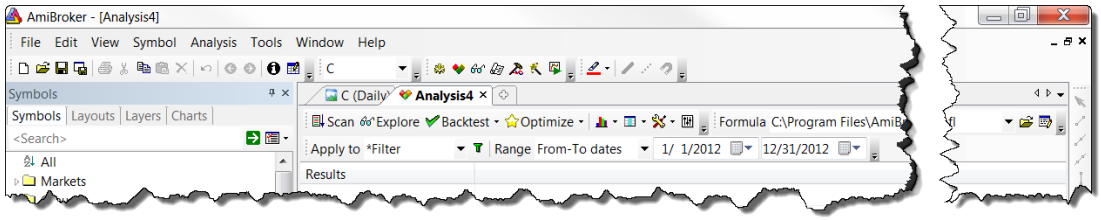
```
If(Status("action")==actionCommentary)
{
// computations needed only for the commentary

// WriteVal, WriteIf, and PrintIf statements

}
```

This section of code will only be executed if the commentary will be visible if produced, thus saving execution time.

Formula Editor

The formula editor was introduced in Chapter 3, Exercise 5. There is additional documentation in the *AmiBroker User's Guide section on Formula Editor*.

The backtest, explore, scan, and optimize features all make use of the common analysis controls. They were briefly discussed in Chapter 3, Exercise 5.



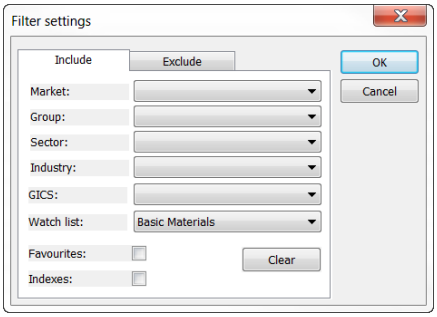
Apply To

Gives the choice of applying the analysis to:

- All symbols
- Current symbol
- Filter -- symbols defined by the include and exclude filters

Filter

Uses the Filter Settings Dialog box to select issues to be included and / or excluded.



Range

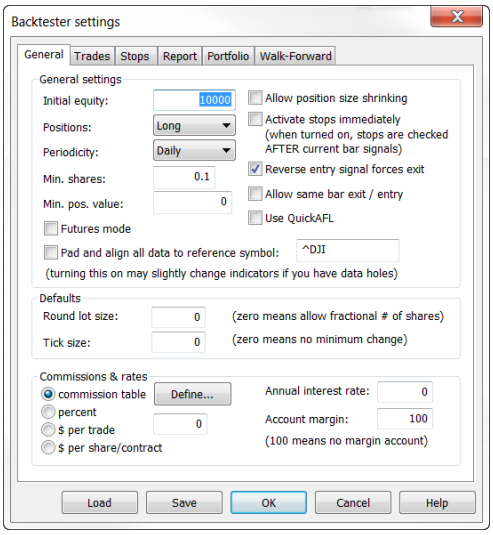
Gives the choice of applying the analysis to:

- All quotes
- 1 most recent bar
- 1 most recent day
- From To date range

Settings

Clicking the **Settings** button that looks like a little wrench opens the Settings dialog. It has six tabs. Each will be discussed.

General tab



Initial equity - Enter the starting balance for the trading account.

Positions - The menu gives three choices. Select whichever is meaningful for your trading system.

- Long
- Short
- Long and Short

Periodicity - The menu gives choices consistent with your database. If you have an end-of-day database, you will only see choices of daily and longer. If you have an intraday database, you will see shorter time periods.

Min Shares - The minimum number of shares required to open a position. If your account does not have enough cash to purchase this many shares, it will pass on the trade.

Min pos value - The minimum dollar amount required to open a position. If your account does not have at least this much cash, it will pass on the trade.

Futures mode - Check this box to have the trade accounting done using margin deposit and point value.

Pad and align all data to reference symbol - This is off by default. When checked, it will be turned on. When on, the quotes of all symbols are padded and aligned to the reference symbol. Padding and aligning insures that there will be data for exactly the same dates and times in both series. The symbol being tested will have data removed for periods when the reference symbol has no data, and will have data added (previous value copied) for periods when it has no data but the reference symbol does. Enter the ticker for the reference symbol in the box.

Allow position size shrinking - When checked, allows position size to shrink so that a partial position can be taken if cash is insufficient for a full position.

Activate stops immediately - When you trade on open and want to have built-in stops activated on the same bar, check this box. When you trade on close and want built-in stops to be activated from the next bar, uncheck this box.

Reverse entry signal forces exit - Check this box to turn on, uncheck to turn off. When on, a signal to enter a short position forces an exit to a long position being held, and vice versa. When off, an entry to a short position does not force the exit of a long position - the long position waits for its own exit signal.

Allow same bar exit - Check this box to turn on, uncheck to turn off. When on, the exit from a trade can occur on the same bar as the entry. When off, the earliest exit is the next bar.

Use QuickAFL - When checked, use QuickAFL. QuickAFL examines the visible charts and recomputes indicators using only as much data as necessary to redraw the visible portion or recompute the selected range.

Round lot size - The minimum number of shares, or contracts, that can be purchased. For mutual funds, where it is possible to purchase fractional shares, enter 0.

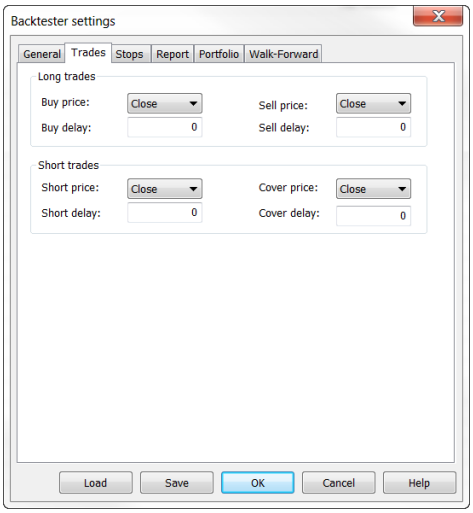
Tick size - The smallest change in the price that can be recorded. A setting made on this screen is a global setting and will apply to all symbols. You can enter a value for each symbol, which will override the global setting, using Symbol > Information.

Commissions & rates - Can be set from a table, or you can enter a number to be used as a percentage, a cost per trade, or a cost per share.

Annual interest rate - The rate you earn on funds that are in cash and not invested in a position.

Account margin - This setting is for your entire account. It is not related to margin for futures. It is the percentage of the funds in your trading account that comes from your cash. 100 means all of your account comes from your own money. 50 means that 50 percent of your account comes from your own money and you are borrowing the other 50 percent.

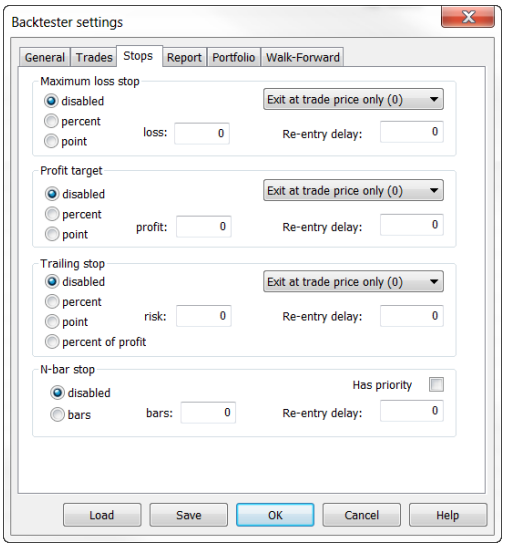
Trades tab



Buy Price - The menu gives you a choice of Close, Open, High, Low, or Average. This price will be used if your trading system issues a market order. That is, if it only signals Buy or Sell without specifying the BuyPrice or SellPrice. You can use an AFL statement (BuyPrice = xxx) to override settings made on this screen. Sell Price, Short Price, and Cover Price are handled the same.

Buy Delay - The number of bars between the generation of the signal and the execution of the trade. If you are using daily bars, computing signals in the evening, and trading the next day, use a value of 1. If you are watching the market intra-day, get a signal near the close, and make the trade on that same bar, use a value of 0. You can use an AFL statement (SetTradeDelays) to override settings made on this screen.

Stops tab



Maximum Loss Stop - You have a choice of:

- Disabled - No maximum loss stop is active.
- Percent - Enter a maximum loss stop as a percentage of the entry price. Enter the percentage number in the field.
- Point - Enter a maximum loss stop as a fixed number of points. Enter the number of points in the field.

The **Exit at** menu gives you three choices.

- Exit at trade price only - Use the price set on the Trades tab. If you have set your sell to be at the close, the stop will be executed at the close, at the closing price.
- Exit intraday at stop - Use the price computed by the AFL code.
- Exit next bar at trade price - Use the price set on the Trades tab, and exit on the next bar.

Re-entry delay - Number of bars to wait before taking a new position in this same issue.

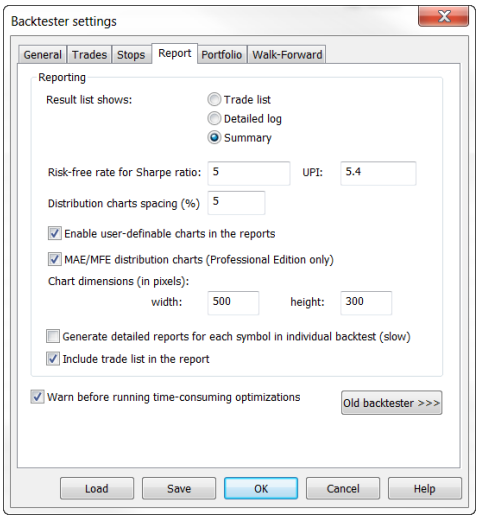
Profit target and **Trailing stop** are controlled in the same way.

N-bar stop - You have a choice of:

- Disabled - No N-bar stop in place.
- bars - Enter the number of bars in the field. An exit will be made after this number of bars. Both the entry bar and exit bar count. A position entered on Tuesday and exited on Wednesday will show that it has been held two days.

Has priority - When checked, the N-bar stop has priority over other exits that are signalled for the same bar. When unchecked, other stops are considered first, such as a profit target hit on the same bar as the N-bar exit stop.

Report tab



Result list shows - give you a choice of:

- Trade list - Report a listing with a row for every trade made.
- Detailed log - Report a listing with a row for every bar, including information about scores and positions.
- Summary - Report a listing with one entry per backtest.

Risk free rates - Interest rates used when computing Sharpe ratio and Ulcer Performance Index.

Distribution charts spacing - Controls the spacing between bars on some of the charts.

MAE/MFE distribution charts - Only available with Professional Edition of AmiBroker. When checked, produce the Maximum Adverse Excursion and Maximum Favorable Excursion distribution charts.

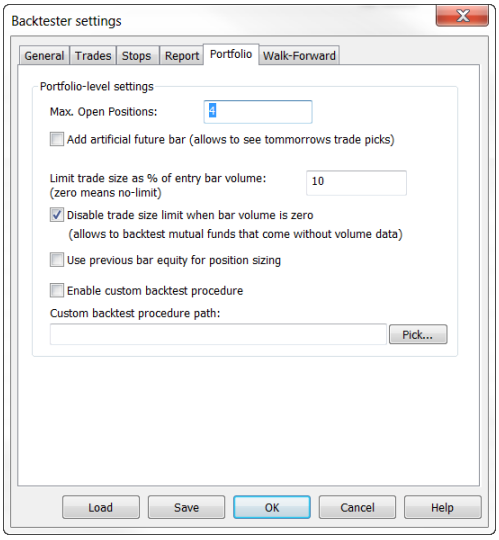
Generate detailed reports for each symbol in individual backtest - When checked, generate a full report for each symbol. This will increase the time of the run and increase the disk space used.

Include trade list in the report - Check to turn on, which is the default. When turned on, generates a trade list with every backtest.

Warn before time consuming operations - Check to turn on, which is the default. When turned on, a dialog box will pop up when a long run has been requested, asking for confirmation to continue.

Old Backtester - Provides access to a previous version of the backtester. At the introductory level, you probably do not need the old backtester.

Portfolio tab



Max open positions - Maximum simultaneous open positions. This can also be set by AFL code (SetOption("MaxOpenPositions", number)), which overrides the value entered on the Portfolio tab.

Add artificial future bars - Add empty bars onto which you can plot indicators. Useful for plotting the value of stops and profit targets to be used the next day.

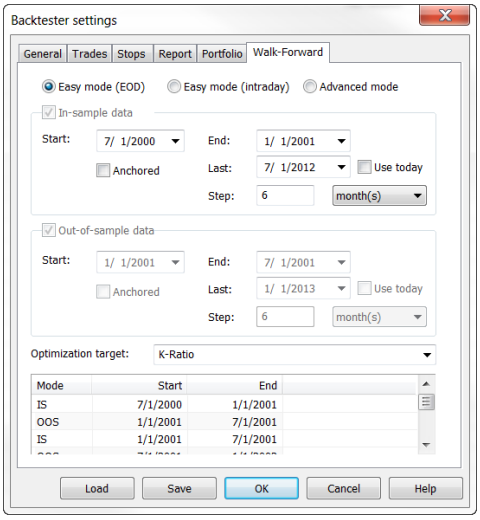
Limit trade size as percentage of entry bar volume - Set a limit so that your trade does not dominate the trading or try to take impossible positions.

Disable trade size limit when bar size is zero - Useful when trading mutual funds and indices that have no volume information.

Use previous bar equity for position sizing - When unchecked, which is the default, all funds are immediately available, so use intra-day equity when taking new positions.

Enable custom backtest procedure - Allows you to specify a procedure to use as the fitness function. This topic is beyond the introductory level, and is thoroughly described in *Quantitative Trading Systems*.

Walk-Forward tab



You have a choice of:

- **Easy mode (EOD)** - End-of-Day easy mode - AmiBroker handling the coordination of dates so in-sample and out-of-sample periods align properly.
- **Easy Mode (Intraday)** - Intraday easy mode - AmiBroker handling the coordination of dates so in-sample and out-of-sample periods align properly.
- **Advanced Mode** - You have complete control over the length of the in-sample period, the length of the out-of-sample period, and the alignment of the two.

This description is for the Easy mode EOD.

Start - The first date for the first in-sample period.

End - The last date for the first in-sample period.

Last - The last date for the last full in-sample period.

Step - The length of the out-of-sample period.

Anchored - When checked, the beginning of every in-sample period is the date of the first day of the first in-sample period. With each successive step, the length of the in-sample period grows longer. When unchecked, the in-sample periods are all the same number of days long, and the starting dates move forward in time by the length given for Step. [It is much better modeling and simulation technique to use a moving window of in-sample data than to use an anchored starting date. That is, leave anchored unchecked.]

Use today - When checked, do not accept a date for Last. Instead, continue the in-sample and out-of-sample periods using all the data available.

Optimization target - The menu gives a list of the built-in metrics that can be used as fitness functions. Developers who want to reward equity growth while penalizing drawdown might want to try one of the metrics that have that characteristic:

- RAR/MDD
- CAR/MDD
- RRR (Risk-Reward Ratio)
- Ulcer Performance Index
- K-Ratio

Advanced developers can:

- Use advanced mode to completely control the walk forward stepping.
- Write their own fitness function and instruct the backtester to use it.

Action buttons

There are four action buttons -- Explore, Backtest, Optimize, and Scan. Clicking a button initiates the action associated with that button. They are discussed in the order during development that they are most likely to be used.

Explore

The previous sections - Main Window and Settings Button - explained how to set up for Analysis. When you click the Explore Button on the main Analysis window, you start the Analysis run.

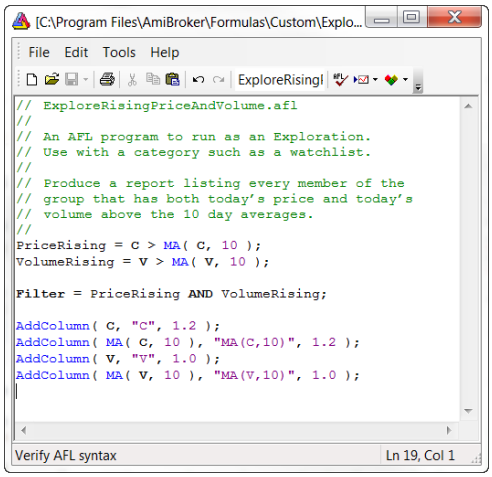
The exploration runs an AFL program that searches for conditions that you specify, then produces a report. In addition to producing reports, Explore is a valuable aid while programming and debugging trading systems. Use it to print out intermediate results at various checkpoints in your afl code.

Every program that will be run as an exploration has a Filter statement in it. In order for a result to appear in the report, that result must pass the filter. For example, you might want a report showing which members of the S&P 100 have both today's price and today's volume greater than their 10 day averages.

Follow these steps:

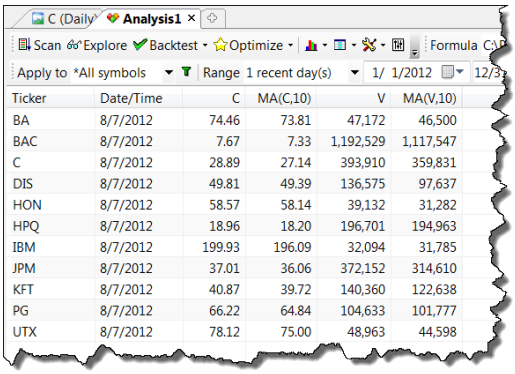
1. Use the Formula Editor to write the AFL code and save it in a file named ExploreRisingPriceAndVolume.afl.

```
// ExploreRisingPriceAndVolume.afl
//
// An AFL program to run as an Exploration.
// Use with a category such as a watchlist.
//
// Produce a report listing every member of the
// group that has both today's price and today's
// volume above the 10 day averages.
//
PriceRising = C > MA( C, 10 );
VolumeRising = V > MA( V, 10 );

Filter = PriceRising AND VolumeRising;

AddColumn( C, "C", 1.2 );
AddColumn( MA( C, 10 ), "MA(C,10)", 1.2 );
AddColumn( V, "V", 1.0 );
AddColumn( MA( V, 10 ), "MA(V,10)", 1.0 );
```



2. In **Analysis**, click **Pick**, navigate to the file you just created, and select it.
3. Click **Use filter**, click **Define**, select the category you want to use.
4. Click **1 Recent Day**.
5. Click **Explore**.

The list of issues in your category meeting the criteria in your AFL appear in the results window.



Ticker	Date/Time	C	MA(C,10)	V	MA(V,10)
BA	8/7/2012	74.46	73.81	47,172	46,500
BAC	8/7/2012	7.67	7.33	1,192,529	1,117,547
C	8/7/2012	28.89	27.14	393,910	359,831
DIS	8/7/2012	49.81	49.39	136,575	97,637
HON	8/7/2012	58.57	58.14	39,132	31,282
HPQ	8/7/2012	18.96	18.20	196,701	194,963
IBM	8/7/2012	199.93	196.09	32,094	31,785
JPM	8/7/2012	37.01	36.06	372,152	314,610
KFT	8/7/2012	40.87	39.72	140,360	122,638
PG	8/7/2012	66.22	64.84	104,633	101,777
UTX	8/7/2012	78.12	75.00	48,963	44,598

The elements of AFL code that are unique to explorations are:

- The Filter statement.
- The AddColumn statements.

The Filter statement has the form:

```
Filter = any expression that evaluates to True or False;
```

If you want an entry in the report for every symbol in the watchlist, use this:

```
Filter = 1;
```

The AddColumn statement has the form:

```
AddColumn (Array, “Name”, format);
```

Where Array is the value to be reported, “Name” is the heading for the column, and format is the format of the output.

Format is a pair of digits with a decimal point between them, such as 6.4.

The first digit is the number of spaces to use. 6.4 guarantees 6 spaces for the number, even if it requires fewer.

The second digit is the number of decimal places to display. 6.4 displays the result to 4 decimal places.

Backtest

The Backtest button begins execution of a backtest. That is, a test of the profitability of the buy and sell rules defined in an AFL program as it is applied to historical data. The AFL code that buys and sells is a trading system.

The backtest button has a pull-down menu that gives two choices:

- Portfolio Backtest - Apply the backtest program to a group of issues, all of which are candidates to be held. The backtester calculates the buy and sell signals for each individual issue in the group. Then it goes through the signals in order by time. Whenever there is a buy signal and there is cash available to take the trade, a position is taken. If there is not enough cash, the trade is passed. Whenever there is a sell signal, the position is sold and the cash made available for another purchase. The equity curve (running account balance) and report are produced for the portfolio as a whole.
- Individual Backtest - As above, apply the backtest to a group of issues, all of which are candidates to be held. The backtester calculates the buy and sell signals for each individual in the group. Then, differing from the portfolio method, it assumes each of the issues is the only candidate to be held, and calculates a report for each of the issues individually.

The same AFL code can be run as a portfolio backtest and an individual backtest.

In order to run a backtest on a portfolio, apply to **filter** and define a group. Then click **Backtest** (portfolio backtest is the default) or use the pull-down menu and choose **portfolio backtest**.

In order to run a backtest on a single issue, apply to **current symbol** and click **backtest**. The result will be titled <portfolio>. If you want the result to list the ticker symbol, use the menu and choose **individual backtest**.

In order to run a backtest on several issues in one pass, treating them as individuals rather than as a portfolio, apply to **filter**, use the menu and choose **individual backtest**. There will be a result for every symbol in the group, but no portfolio result.

In processing the AFL file, the backtester performs the calculations and assignment statements. The key component is the set of buy and sell (and, perhaps, short and cover) signals.

The format of a buy statement is:

```
Buy = whatever condition you want to signal taking a new position;
```

And of a sell statement:

```
Sell = whatever condition you want to signal exiting that position;
```

When the calculations that make up the condition to buy evaluate to True, or 1, that constitutes a buy signal.

When the calculations that make up the condition to sell evaluate to True, or 1, that constitutes a sell signal.

A very simple trading system, often used as an example because of its simplicity, not because of its profitability, is a moving average crossover.

Look at the code from Example 6B, Chapter 3.

```
// Example6B.afl
// A trading system based on the
// crossover of two moving averages.
// Buy when the faster moving average
// crosses up through the slower moving
// average.
// Sell when the faster moving average
// crosses down through the slower
// moving average.

FastMALength = 5;             //Guess that 5 is a good value
SlowMALength = 20;            //Guess that 20 is a good value

FastMA = MA( C, FastMALength );
SlowMA = MA( C, SlowMALength );

Buy = Cross( FastMA, SlowMA );
Sell = Cross( SlowMA, FastMA );
```

When ever the right-hand side of the Buy = statement evaluates to be True, that is a buy signal. The Cross function returns a value of true on every bar where the first argument is higher than the second argument on this bar, but the first argument was lower than the second argument on the previous bar. That is - where the first argument crossed up through the second argument.

In order to test the profitability of a trading system when it is applied to a portfolio, the system must have some way of controlling the number of positions held at any one time. It must also be able to allocate funds to each new position. There are AFL statements that do that.

The statement:

```
SetOption ("MaxOpenPositions", 3);
```

tells the backtester that there can be a maximum of 3 open positions. If 3 positions are being held and a buy signal comes from a symbol not being held, it is ignored and that trade

is passed. If fewer than 3 positions are being held and there are sufficient funds to take a new position, a position will be taken in the symbol that generated that buy signal.

Having the possibility of holding 3 positions implies that there is funding for 3 "tracks." Each new position is funded by one third of the account equity. As the equity grows, more money is available for purchases in each of the three tracks - they share the funds equally.

You might want a way to test the effect of allowing a differing number of maximum positions. These statements are equivalent to the ones above, but give the flexibility of changing the maximum number easily.

```
PosQty = 3;
SetOption( "MaxOpenPositions", PosQty );
SetPositionSize( 100 / PosQty, spsPercentOfEquity );
```

Put these statements at the beginning of the code from Example 6B, and save it under a new file name, say BacktestExample.afl.

```
// BacktestExample.afl
// A trading system based on the
// crossover of two moving averages.
// Buy when the faster moving average
// crosses up through the slower moving
// average.
// Sell when the faster moving average
// crosses down through the slower
// moving average.

PosQty = 3;
SetOption( "MaxOpenPositions", PosQty );
SetPositionSize( 100 / PosQty, spsPercentOfEquity );

FastMA = MA( C, 5 );
SlowMA = MA( C, 20 );

Buy = Cross( FastMA, SlowMA );
Sell = Cross( SlowMA, FastMA );

Plot( C, "C", colorBlack, styleCandle );
e = Equity();
Plot( e, "equity", colorGreen, styleLine | styleOwnScale );
```

When you have the program changed and are in the Formula Editor, send the program to **Analysis**. This causes the file being edited to become the active or selected file for the Analysis process.

Click the Analysis tab, apply to **filter**, then **Define**, and select a watchlist with 5 or more symbols in it. The 9 S&P sector exchange traded funds make a good watchlist for testing. The 9 symbols are: XLB, XLE, XLF, XLI, XLK, XLP, XLU, XLV, XLY.

Set the range to 1/1/2000 to 1/1/2010.

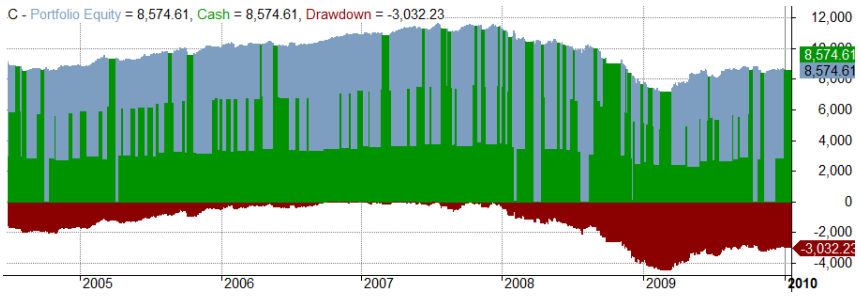
Click **Backtest**.

The Results area will show the results of running the system as a portfolio.

The results are poor, but perhaps they can be improved later.

Click the **Report** button. The Report will open. Expand it and look at the statistics on the first tab. Look at the trades on the Trades tab. Note that there are several positions held at a given time - between 0 and 3.

Click the **Equity** button and scroll back and forth. Blue areas are positions, green are cash. You can see when there were 0, 1, 2, and 3 positions held.



Use the pull-down menu at the right side of the Backtest button and select **Individual Backtest**. The Results area will show the results - one line for each of the members of the watchlist.

Context Menu

After an Analysis run where there are results in the results window, right-click anywhere in the window to bring up the Analysis Context menu. There are two rather different types of processing on the context menu:

1. Show arrows resulting from trades taken on a single security - the top three.
2. Send symbols resulting from tests made to a group of securities to watchlists - the next four.

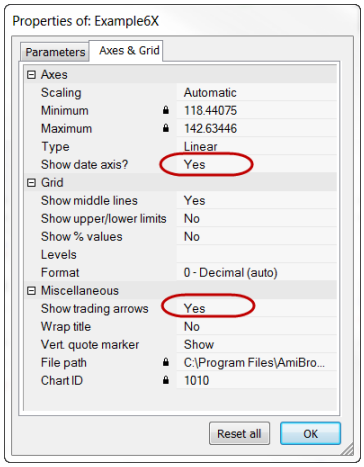
We'll demonstrate in two examples.

First example. Running a trading system on a single security will get us to the first set of three options. To get things set up, follow these steps:

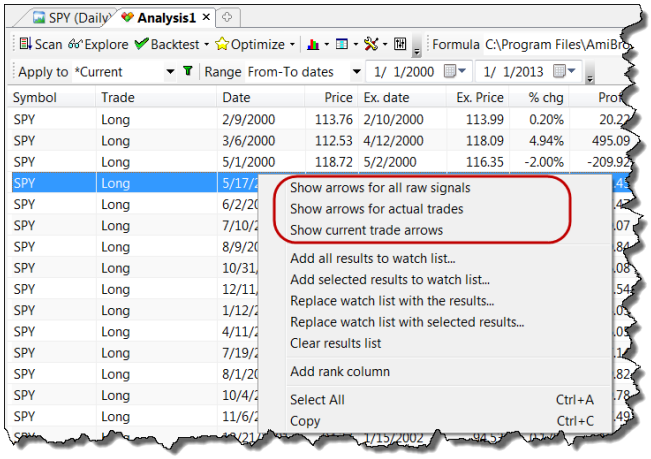
1. Using the **Analysis** tab. Click **Pick** and select **Example6B** again.
2. Click **Edit**, which will open the Formula Editor and display the AFL code of the system. Add these two lines to the bottom of the program:

```
Plot( FastMA, "FastMA", colorRed,styleLine);
Plot( SlowMA, "SlowMA", colorBlue, styleLine);
```

   Save the program as Example6X.afl.
3. Click the **Verify Syntax** icon to test for typing errors.
4. Using the **Apply** icon, drop down its menu and use the **Insert** option. A new pane will open with the price in a candlestick chart, a 20 period simple moving average, and a 5 period simple moving average.
5. Right-click the pane with the new chart, bringing up the Chart Pane Context menu. Click Parameters, then the Axes & Grid tab. Set the options to show the date axis and to show trading arrows. Click OK to leave the context menu.



6. Set some parameters for Analysis and run a Backtest. Anything will do.
   Use just the current symbol.
   Set the date range for a year or two.
   Positions to Long.
   Periodicity to Daily.
   Buy and Sell on the Close with Delay 0.
   Reporting show a Trade List.
7. Click **Backtest**. The Results window will show results.
8. Right-click the Results window to bring up the context menu.

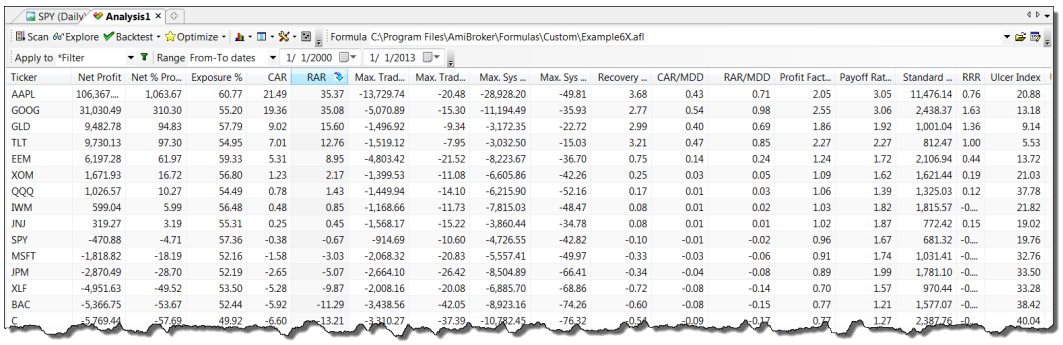


9. Since this run was made on a single symbol, the top three options are meaningful.
   - Show arrows for all raw signals - Sometimes there are signals that do not result in trades. This option shows all signals.
   - Show arrows for actual trades - Shows those signals that became trades.
   - Show current trade arrows - Each row of the results represents one trade. Select one trade. This option scrolls the chart and shows the arrows for that one trade.

     This chart shows arrows for actual trades. (The arrows drawn for the first option result in the same chart, because there are no extraneous signals.)

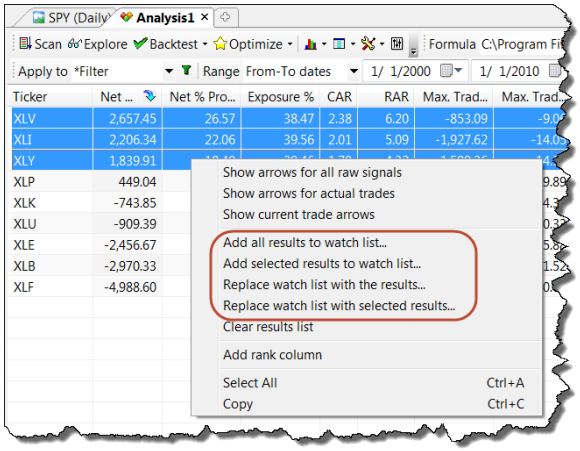
Second example. Options four through seven come after running the system using a group of stocks. Follow these steps (1 through 5 are the same as the previous example):

1. From the Analysis menu, select **Analysis**. Click **Pick** and select **Example6X**.
2. Click **Edit**, which will open the Formula Editor and display the AFL code of the system.
3. Click the **Verify Syntax** icon to test for typing errors.
4. Using the Apply icon, drop down its menu and use the **Insert** option. A new pane will open with the price in a candlestick chart, a 25 period simple moving average, and a 5 period simple moving average.
5. Right-click the pane with the new chart, bringing up the Chart Pane Context menu. Set the options to show the date axis and to show trading arrows. Click OK to leave the context menu.
6. Set some parameters for Analysis.
   Use filter and select a watchlist of your choosing.
   Set the date range for a year or two.
   Positions to Long.
   Periodicity to Daily.
   Buy and Sell on the Close with Delay 0.
   Reporting show Summary.
7. On BackTest's drop-down menu, select **individual backtest**. You will see a progress bar as the system is applied to each symbol in the watchlist.
   The Results window will show results. Sort the results as you wish by clicking a column header.



Ticker	Net Profit	Net % Pro...	Exposure %	CAR	RAR	Max. Trad...	Max. Trad...	Max. Sys ...	Max. Sys ...	Recovery ...	CAR/MDD	RAR/MDD	Profit Fact...	Payoff Rat...	Standard ...	RRR	Ulcer Index
AAPL	106,367....	1,063.67	60.77	21.49	35.37	-13,729.74	-20.48	-28,928.20	-49.81	3.68	0.43	0.71	2.05	3.05	11,476.14	0.76	20.88
GOOG	31,030.49	310.30	55.20	19.36	35.08	-5,070.89	-15.30	-11,194.49	-35.93	2.77	0.54	0.98	2.55	3.06	2,438.37	1.63	13.18
GLD	9,482.78	94.83	57.79	9.02	15.60	-1,496.92	-9.34	-3,172.35	-22.72	2.99	0.40	0.69	1.86	1.92	1,001.04	1.36	9.14
TLT	9,730.13	97.30	54.95	7.01	12.76	-1,519.12	-7.95	-3,032.50	-15.03	3.21	0.47	0.85	2.27	2.27	812.47	1.00	5.53
EEM	6,197.28	61.97	59.33	5.31	8.95	-4,803.42	-21.52	-8,223.67	-36.70	0.75	0.14	0.24	1.24	1.72	2,106.94	0.44	13.72
XOM	1,671.93	16.72	56.80	1.23	2.17	-1,399.53	-11.08	-6,605.86	-42.26	0.25	0.03	0.05	1.09	1.62	1,621.44	0.19	21.03
QQQ	1,026.57	10.27	54.49	0.78	1.43	-1,449.94	-14.10	-6,215.90	-52.16	0.17	0.01	0.03	1.06	1.39	1,325.03	0.12	37.78
IWM	599.04	5.99	56.48	0.48	0.85	-1,168.66	-11.73	-7,815.03	-48.47	0.08	0.01	0.02	1.03	1.82	1,815.57	-0....	21.82
JNJ	319.27	3.19	55.31	0.25	0.45	-1,568.17	-15.22	-3,860.44	-34.78	0.08	0.01	0.01	1.02	1.87	772.42	0.15	19.02
SPY	-470.88	-4.71	57.36	-0.38	-0.67	-914.69	-10.60	-4,726.55	-42.82	-0.10	-0.01	-0.02	0.96	1.67	681.32	-0....	19.76
MSFT	-1,818.82	-18.19	52.16	-1.58	-3.03	-2,068.32	-20.83	-5,557.41	-49.97	-0.33	-0.03	-0.06	0.91	1.74	1,031.41	-0....	32.76
JPM	-2,870.49	-28.70	52.19	-2.65	-5.07	-2,664.10	-26.42	-8,504.89	-66.41	-0.34	-0.04	-0.08	0.89	1.99	1,781.10	-0....	33.50
XLF	-4,951.63	-49.52	53.50	-5.28	-9.87	-2,008.16	-20.08	-6,885.70	-68.86	-0.72	-0.08	-0.14	0.70	1.57	970.44	-0....	33.28
BAC	-5,366.75	-53.67	52.44	-5.92	-11.29	-3,438.56	-42.05	-8,923.16	-74.26	-0.60	-0.08	-0.15	0.77	1.21	1,577.07	-0....	38.42
C	-5,769.44	-57.69	49.92	-6.60	-13.21	-3,310.27	-37.39	-10,782.45	-76.32	-0.54	-0.09	-0.17	0.77	1.27	2,387.76	-0....	40.04

8. Each row represents the result of running this system on an individual symbol. Select several rows by holding the CONTROL key while left-clicking the rows you want. Right-click the Results window to bring up the context menu.



9. The four options marked are applicable:
   - Add all results to watchlist - Adds all the symbols in the results list to the watchlist. Duplicates will be removed.
   - Add selected results to watchlist - Adds the symbols in the rows you have selected to the watchlist.
   - Replace watchlist with the results - Makes the watchlist empty, then adds all the symbols in the results list.
   - Replace watchlist with selected results - Makes the watchlist empty, then adds the symbols in the rows you have selected to the watchlist.

Optimize

To optimize is to make an organized search for the set of values for the parameters of a trading system that maximize the objective function you are using to compare alternative systems.

There is one statement that must be added to a trading system to allow it to be optimized.

```
var  =  Optimize ( "name" , def , min , max , step );
```

`var` is a variable in your AFL program.

`name` is a string literal that will be used as a label when reporting.

`def` is the default value given to var whenever the program is evaluated in a mode other than optimize.

`min` is the minimum value of the range you wish to search.

`max` is the maximum value of the range you wish to search.

`step` is the increment by which the value is increased for each optimization.

For example,

```
MALength = Optimize ( "MALength", 5, 1, 21, 2 );
```

The variable named MALength will be given values of 1, 3, 5, ... 21 in successive optimization passes.

Example 8, Chapter 3, ran an optimization. Review that example now.

Portfolio Optimization

If you click the Optimize button without pulling down the menu to make other choices, you are asking for a portfolio optimization. Review the section on the backtest button and note that it, too, defaults to a portfolio test.

If you have checked **use filter,** clicked **define,** and selected a watchlist or some other list of symbols, the backtest and the optimization are carried out on that group as a portfolio.

If the watchlist you have selected contains a single symbol, or if you checked **use current**, then your portfolio is a single issue, and the backtest or optimization is carried out on the single issue.

Optimization Basics

An optimization is simply a series of backtest runs, each run made with a different set of values for the parameters. If you were patient enough, you could type each variable in

your program individually and make the same series of backtests. The advantage of having the optimization done by AmiBroker is that the details are handled by the program, and the results are displayed in the Results window for your evaluation.

After the alternative runs have been made (whether you used the optimizer or hand-entered the values), you must decide which set of values to use; and that is a decision you must make for yourself. Your level of comfort with the system that emerges from your design and testing will reflect the method you used to choose the best alternative.

The period of time, and the data associated with that period, that is processed as each of the backtest or optimization runs is made is called the in-sample data. When searching for the best set of values for the parameters, the in-sample data is being searched. The in-sample results will be good — they are always good — we do not stop working with the system until they are good. The real test is see whether those results are due to a good model that correctly identified the signal portion of the data, the patterns, rather than the noise. To make that test, change the dates to a period that follows the in-sample data and that has not been used to select the parameter values. That new data and time period is called the out-of-sample data. If the results obtained by running the system on the out-of-sample data are good, then there is a possibility that the system can, indeed, identify patterns that precede profitable trading opportunities.

But please be careful. It is too easy to run an optimization (on in-sample data), pick the best values, run the system on the out-of-sample data, notice that an adjustment to the system would make the out-of-sample results much better, change the code for the system, and rerun the system. Every time that is done, the data that was previously out-of-sample has become increasingly in-sample. Eventually the results from both the in-sample and out-of-sample runs will be good; but that success is partially due to expanding the in-sample data set. A new, fresh, out-of-sample set of data is now required.

Objective Functions

AmiBroker has 20-some built-in metrics that are automatically computed and reported for each run. If the ranking produced by one of these metrics agrees with your own ranking, then you can use that metric as your objective function (also called a fitness function). The purpose of the objective function is to quantify the subjective preferences of the trader. There is an article, *Managing Subjectivity with Objective Functions*, discussing this in more detail on my website.

If you want a custom function, you can write the AFL that describes it, tell AmiBroker to compute a score using that function for each run, and rank the results by that function. There are examples of doing that in *Quantitative Trading Systems*.

Most trading system development platforms do not provide the power and flexibility in creation and use of custom objective functions that AmiBroker does.

In fact, most have a default objective function of net profit, and most have no way to specify any other. Unfortunately, net profit is a poor choice for an objective function.

Net profit is one of the metrics that tends to perform well in-sample, but poorly out-of-sample. That is, most trading systems can be adjusted until they produce high net profit results on the in-sample data, but profit for the out-of-sample data is often poor.

Better choices of metrics are those that reward equity growth while penalizing drawdown. AmiBroker has several of those as built-in metrics. They include:

- RAR/MDD - Risk adjusted Annual Return / Maximum DrawDown.
- CAR/MDD - Compound Annual Return / Maximum DrawDown.
- K-ratio - a metric developed by Lars Kestner.
- RRR - Risk - Reward Ratio.
- Ulcer Performance Index.

Exhaustive versus Non-Exhaustive Optimizing

Unless you specifically request otherwise, when you code your optimize functions and then click the optimize button, AmiBroker will use every combination of parameter values you have given - it will perform an exhaustive search. If one variable goes from 2 to 100 in steps of 2, that is 50 values; if another goes from 1 to 20 in steps of 1, that is 20 values. An exhaustive optimization would run a backtest for each of the 1000 (20 times 50) alternatives. After the 1000 runs, AmiBroker will sort them into descending order according to the metric you chose and display the results. Since every possible combination has been evaluated, the best one is at the top of the list.

AmiBroker has alternative methods of carrying out the search that will probably find the best alternative, and will do it in much less time because it does not search the entire space. The search space, as the universe of all possible combinations of parameter values is called, will have some regions where the objective function values are high, and other regions where they are low. There are some optimization methods that focus their search efforts on those areas that are more promising and skip lightly over the less promising areas. These are called non-exhaustive search methods. The one implemented in AmiBroker is CMAE (Covariance Matrix Adaptation Evolutionary Strategy).

The absolute best score of all the alternatives, and the set of values for the parameters that are associated with it, is called the global maximum. It is the best solution. An exhaustive search will find the global maximum. Other regions will have scores higher than the surrounding areas, but will not be the absolute highest. They are called local maximums.

Non-exhaustive search methods sometimes find local maximums, but not the global maximum; although usually they do find the global maximum.

There are advantages to using the non-exhaustive methods.

- One is that they are quicker to converge — quicker to find the region that contains the maximum. If your optimization has only a few thousand alternatives to consider, you may not notice the difference. If it has 100,000 or more, a run that would take hours or days to complete, you will definitely notice the difference. There is a posting on my website that discusses *Optimization and the Curse of Dimensionality.*
- Another is that the algorithm used to evaluate the objective function can be clever and can take into account the surrounding points. One of the goals is to pick a set of values that score well, but that also are stable and robust. A robust area of a search space has good scores not only at the maximum, but also in the surrounding area.

To use the non-exhaustive method, place this statement at the start of your program:

```
OptimizerSetEngine("cmae");
```

Walk Forward

To start a walk forward run, pull down the menu next to the Optimize button and select Walk-Forward Optimization.

Example 9, Chapter 3, ran a walk forward. Review that example now.

Background and Justification

While there is nothing special about the optimization process - it is just an organized way to look at a lot of alternatives - there is something very special about the walk forward process. The walk forward process is the best tool available to you for trading system validation. While you are developing your trading system, you have access to all the data. Using your best efforts, you try to develop your system using in-sample data and test it using out-of-sample data. But what will happen when you start actually trading the system? The walk forward process can give you an idea of what might happen.

In the development phase, you used the in-sample data and ran optimizations to search for patterns, then tested on out-of-sample data. The walk forward process automates this for you and provides several important things:

- It automates the process of optimizing over a set of in-sample data.
- It automates selection of the best set of values.
- It automates testing over the out-of-sample data.

- For each walk forward step, it gives you a look at what happens when your system goes from in-sample to out-of-sample.
- It gives you an idea of how long your system remains in synchronization with the market it is modeling.
- It provides a set of out-of-sample results that you can use as a baseline to compare the actual results of your own trading.

The walk forward process is the culmination of the steps of:

- Choose your own objective function.
- Optimize in-sample.
- Test out-of-sample.

Note that in the walk forward process, the best set of values for the parameters is chosen, and the out-of-sample results are calculated using those values. At this point, you do not choose which set to use; you have already incorporated the criteria that are important to you into your objective function. For a deeper discussion of objective functions, see *Quantitative Trading Systems*.

Mechanics of Walk Forward

By the time you get to the walk forward stage, you have a trading system you like and you know what tradables it models well. You are looking for three things:

- How long is the in-sample period?
- How long is the out-of-sample period?
- What is the out-of-sample performance?

The underlying market is changing as the fundamentals upon which it is based change. These could be interest rates, the strength of the currency, the phase of the economic cycle, the profitability of the sector and of the specific company, and so forth. Your model is able to identify the important characteristics and identify profitable trading opportunities, but the model and the market drift from being in synchronization to out. The length of the in-sample period must be shorter than the period of synchronization. Your system sets its parameter values in the in-sample period and trades in the out-of-sample period. So, your system must sync-up during the in-sample period, then trade for as long as the synchronization remains strong. There is an article on my website that discusses *Logic, Data, and Synchronization*.

Open **Analysis**, click the **Settings** button, and the **Walk-Forward** tab. Review Chapter 3, Example 9, for screen capture images of these steps.

1. Click **Easy mode (EOD)**.

Decide when to begin your walk forward. Of course, you must have data from the starting date to the present. But you may decide not to use all of your historical data. By using your judgement, or by running some tests, decide whether the oldest data you have has the same characteristics as the more recent data. For example, the characteristics of the entire equities market changed after October 1987, so you may not want to use data earlier than 1988. If your preferred markets ran up sharply in the late 1990s and fell off dramatically in the early 2000s, you may want to include that data, or you may want to exclude it.

2. Set your starting date in the **Start** box.

3. Leave **Anchored** unchecked.

Each model and market combination has a unique relationship. There is no way to tell in advance what the optimum length of the in-sample period is. You must run tests to determine it. By the time you are running the walk forward tests, you should have a very good idea of what length in-sample period to use.

4. Set that length as your in-sample length on the walk forward tab. That is, set the **End** date so that the length of time between Start and End is the length of your in-sample period.

5. Decide what date you want as the last day of the last in-sample period. (You may want to withhold some data from the walk-forward process, just as you withheld some data from the backtesting runs you made earlier.) Enter that date in the **Last** field.

Alternatively, check **Use today**. AmiBroker will use the final period that includes today as the final out-of-sample period.

Your system will remain profitable as long as the model and the market remain in sync during the out-of-sample period. The length you set as the step sets both the re-optimization period and the out-of-sample length. By running tests, you can determine how long, typically, the system remains profitable in the out-of-sample period. That is the longest you can wait between re-optimizations.

6. Set the length you want to use for your out-of-sample and re-optimization in the **Step** field, and use the pull-down menu to select the time increment.

The dates that will be used for each in-sample run and each out-of sample run will be computed and entered into the table. Scroll up and down and review these dates. Be certain they reflect what you want.

7. Select your optimization target from the pull-down list. You should already have selected an objective function and been using it all along during your system development and backtesting. K-Ratio, CAR/MDD, RAR/MDD, RRR, and UPI are all good choices.

8. Click **OK**. This exits the Walk Forward setup dialog.

Each of the walk forward runs is evaluated as it takes place. After an in-sample optimization, the best set of parameter values is chosen and a single test run is made using those values and that specific in-sample period. The trade summary and equity curve for that in-sample run are calculated. Next, a single test run is made, using those same values, for the out-of-sample period that immediately follows. The trade summary and equity curve for that out-of-sample run are calculated.

There are two reports you can watch as the walk forward runs take place.

The first is the set of in-sample and out-of-sample summaries. These are produced automatically and displayed on the walk forward tab.

The second is the concatenation of the in-sample equity curves and out-of-sample equity curves. These are displayed in a pane of the main chart window. But you must first insert the indicator that will display them. Using the charts tabbed menu, expand the equity category, right-click **IS and OOS Equity**, then left-click **Insert**.

As the walk forward progresses, you can click on the main chart tab and watch the equity curves develop.

Use the Report icon's pull down menu and open Report Explorer. Scroll down to the last entry. It is the consolidation of the out-of-sample results. Open that report. Look at the summary statistics and charts. If they look good, your system is ready for trading.

System Health and Position Size

The variation in parameter values from step to step illustrates the dynamic nature of trading systems. It is very rare that one set of parameter values is best over a long period of time. As characteristics of the data change, the model -- the rules and parameter values -- must be allowed to change to keep the system synchronized. Through the life of a trading system, it will pass through periods of close synchronization and high performance, and through periods of weak synchronization and low performance. Just as the model must be allowed to adapt, there is no single position size that is correct for a long period. The question of whether a system is healthy or is broken, and what the correct position size is for current conditions, is critical to trading success. My book, *Modeling Trading System Performance*, addresses this in detail. It explains how to use the "best estimate" of future trading performance to determine the position size that will result in maximum account growth, while holding drawdown to a level determined by the trader.

Synchronization is dynamic. Position size must also be dynamic. The correct position size for a system that is broken is zero.

File Button

The file button has a pull-down menu that gives two choices.

- Export - Export the contents of the Results window to a file in a directory of your choice. The format of the saved file can be either .csv, which can be opened in a spreadsheet for further analysis, or .html, which can be printed, e-mailed, or imported back into the results window.
- Import - Import a file in .html format that was previously exported. The import works best if the file being imported was created by AmiBroker using the Export command.

If the File button is clicked without pulling down the menu, the default operation is Export.

Equity Button

The equity button has a pull-down menu that gives two choices.

- Portfolio Equity - Opens a new pane and displays the equity for the most recent run using the portfolio format.
- Individual Equity - Opens a new pane and displays the equity for the most recent run using the current symbol as the price series tested. Changing the current symbol, either by clicking a row in the results window or by selecting a new symbol, changes the equity plot to reflect testing that symbol.

Report Button

The report button has a pull-down menu that gives two choices.

- View last report - Opens the report for the most recent backtest. The report is in .html format, with one html page per tab of the report. Reports are saved in the directory C:\Program Files\AmiBroker\Reports. They stay there until you remove them.
- Report Explorer - Opens the Report Explorer (which runs as a separate program), and loads a summary line for each report in the Reports directory. You can sort the list by clicking on the header of any column.

  If you have previously turned on the feature that generates detailed reports (on Analysis > Settings > Report, click Generate detailed reports for each symbol in individual backtest), clicking one of the rows in the report listing will open the detailed report.

The menus for the Report Explorer give a method for selecting and deleting unneeded reports. Use the Edit menu. Either select and delete individual rows. Or select all and then delete.

Chapter 10

Write It Yourself

The previous chapters have shown examples of functions, indicators, and trading systems. This chapter explains in more detail how you go about writing your own. The examples in this chapter all assume that the data is end-of-day and comes from the exchange. The data bars are daily bars and have at least a closing price, perhaps open, high, and low. It can be stocks, mutual funds, exchange traded funds, or commodities - the techniques apply to all and we will not distinguish.

Before you start coding

A few questions need answers before you start.

When will you be collecting the data, running your trading system, and issuing your signals? When will you place your orders? When will the orders be executed?

The answers will help you specify the BuyPrice and Trade Delay.

Trading Next Day Open (NDO)

Assume you have a day job and you are doing your data collection in the evening, running your trading systems in the evening, generating signals from the values of the most

recent bar, placing orders in the evening, and anticipating execution at the opening price the next trading day.

You must specify Trade Delay of 1. Your signals come today, your trades happen tomorrow - one day later.

If you plan to issue Market On Open orders, you must specify your BuyPrice as Open.

Note: If you are trading mutual funds and the only price point and trading opportunity tomorrow is the close, then you must specify you BuyPrice as Close.

There are two methods of telling AmiBroker about those choices.

The first is using the Analysis tab > Settings > Trades, and filling in the BuyPrice and BuyDelay. Entering your preferences there sets them as defaults. The second is in your AFL code.

These lines go at the top of your code.

```
SetTradeDelays(1,1,1,1);
BuyPrice = Open;
SellPrice = Open;
```

The SetTradeDelays procedure has four numbers. They represent the trade delay for the Buy, Sell, Short, and Cover, respectively.

The BuyPrice = Open; line tells AmiBroker that you want market orders executed at the open. In AmiBroker, a market order is any order that does not explicitly state a price and a condition. Buy = Cross(C,MA(C,5)); issues a market order, since no price or other condition is specified.

Trading Same Day Close

If you have an accurate and profitable short term trading system - one that holds only a few days - you will probably find that the price change between the close of the bar that generates your signal and the open on the next bar is in the same direction as your signal. That overnight move is often the most profitable portion of the holding period. In order to capture it, you must be able to anticipate the price that will give you a signal before the close of trading, then execute your trade and take your position on the close.

Your TradeDelay is 0, and your BuyPrice is Close.

There are several ways to anticipate the signal.

- Use real-time quotes, either snapshot or streaming, to estimate the closing price just before the closing time. Use the formula for the code that generates your

signal to precalculate what price would trigger your signal. Either enter that as a Limit On Close order, or be ready to place the order yourself.

- Use a root-finding algorithm to precalculate the price necessary for any condition, whether you have the formula or not. For example, you can precalculate the price at which a moving average will cross through a 2 ATR band, and have your order ready when that condition happens. This procedure is explained in detail in *Quantitative Trading Systems*.

Whatever your trading delays and buy price are, if you use AFL statements to set them, the values given in the AFL override the values set in Settings.

Portfolio Considerations

You may be planning a system that trades only one thing - perhaps a broad market index. In that case, the number of positions you hold will always be either zero or one. And you need to make no special arrangements for portfolio management.

You may also be planning to write several trading systems, each for a single tradable, and manage the position allocation outside AmiBroker. Again, you need no special portfolio code.

You may be planing a trading system that keeps track of a universe of potential holdings - say the Fidelity Select Funds, or the S&P Sector ETFs. Your system monitors each of them every day. Whenever there is a buy signal, you want AmiBroker to take a position, but only if there is room in your holdings. In this case you do need special arrangements to handle the portfolio.

There are no settings that let you manage a portfolio - this can only be done in AFL code. Expanding on the previous example, add these lines at the top of your code:

```
MaxPositions = 3;
SetOption("MaxOpenPositions", MaxPositions );
SetPositionSize( 100 / MaxPositions, spsPercentOfEquity);
```

MaxPositions is not a reserved variable. It is one of your variables, and it is used so that you can easily change its value, or optimize it.

SetOption, using the "MaxOpenPositions" option and MaxPositions identifier, tells AmiBroker than the maximum number of open positions is found in the variable MaxPositions.

SetPositionSize tells AmiBroker how to allocate funds. If the second argument is spsPercentOfEquity, treat whatever number the first argument is as a percentage. Allocate

that percentage of the equity to each new position taken. With MaxPositions having a value of 3, each new position gets 33.33% of the equity.

If the second argument is spsValue, then that number of dollars will be used.

If there is not enough in cash to make the full purchase, AmiBroker checks the Setting to see whether **Allow Position Size Shrinking** has been checked. If it has been checked, then all the remaining cash will be used to take the new position.

As you progress, there are additional statements you will want to use to be certain that AmiBroker manages the system as you want it to. For now, let's move on.

Enter a Long Position - Buy

The AFL statement that tells AmiBroker to issue a Buy signal is:

```
Buy = something;
```

Buy is a reserved variable. It is an array and it has values for every day. Those days when your system does not generate a Buy signal, the value for that day is False, or 0. Those days when your system does generate a Buy signal, the value for that day is True, or 1. Whether the value is a 0 or a 1, False or True, depends on the result of evaluating the right hand side - the something.

Whenever there is a Buy Signal and no position is held, there will be a purchase. If there is already a position, the signal will be ignored.

Recall that every statement of this type has an expression on the right hand side and a variable on the left hand side separated by the "=" sign. The "=" sign is the assignment operator.

Processing the statement requires two steps: evaluating the right hand side so that it returns a unique result of the same type as the variable on the left hand side; then assigning that value to (or storing that value in) the variable.

Since Buy needs to be assigned the value of True, or 1, whenever the system generates a Buy signal, the right hand side is usually an expression that is either True or False, 1 or 0.

Some simple examples:

```
Buy = DayOfWeek() == 2;
```

DayOfWeek is a built-in function that evaluates what day of the week the current bar is. It returns a value of 0 for Sunday, 1 for Monday, 2 for Tuesday, and so forth. The "==" is a comparison operator that checks what is on its left and what is on its right and returned a value of either True, the two are the same, or False,

the two are different. If the bar being evaluated is Tuesday, DayOfWeek() returns 2, and the "==" operator returns True - the two are the same. So the value of True, or 1, for the right hand side is assigned to, or stored in, Buy. AmiBroker sees that Buy is True and issues a Buy signal. If you are running a backtest, a new position is taken. If you are running a scan, the report tells you that there is a new Buy signal at the close of today's bar.

```
Buy = Cross ( C, MA(C,5) );
```

Cross is a built-in function that returns a value of True when the first argument, C, has a greater value after this bar than the second variable, MA(C,5); and the first variable had a lower value than the second variable after the previous bar. Said differently, Cross returns True when the first variable crosses up through the second variable. The two being compared do not need to be variables; constants and expressions are also acceptable.

The first variable is the current bar's close. The second variable is the current bar's 5 day simple moving average. If the close was below the 5 day moving average yesterday and is above it today, then Cross returns a True. The evaluation of the right hand side is complete and the True is assigned to Buy. A Buy Signal is generated.

This is called an impulse signal. It is True for one bar, then False. After a few bars it may be True again, but only for one bar.

```
SetUp = C > MA(C,5);
StochSignal = Cross(StochD(),30);
Buy = SetUp AND StochSignal;
```

This represents a setup and a trigger. The first line is the setup. The right hand side evaluates to True on every bar where the close is above its 5 day simple moving average. That will be one day or several days in a row. This type of signal is called a state signal. The second line evaluates to True on those bars when the StochD indicator crosses up though the 30 line. The Buy Signal is generated when both are True - the And operator evaluates to True when both its right hand side and its left hand side are true.

Exit a Long Position - Sell

Sell works the same as Buy.

```
Sell = whatever;
```

Whenever the right hand side evaluates to True, a Sell signal is generated. If there is a position, it will be sold. If there is no position, the signal will be ignored.

Some simple examples:

```
Sell = DayOfWeek() == 5;
```

Sell on Friday.

```
Sell = Cross ( MA(C,5),C );
```

Cross is True when the first argument, MA(C,5), has a greater value after this bar than the second variable, C,; and the first variable had a lower value after the previous bar. That is, when MA(C,5) rose through C — or when C fell through MA(C,5).

Exit a Long Position via Stop

There are several ways to exit a long position:

- A Sell signal, as was just described.
- A timed exit, or n-bar stop.
- A profit target, or profit stop.
- A trailing stop.
- A maximum loss stop.

AmiBroker has built-in stops. Review the sections of this book and the AmiBroker User's Guide for descriptions of stops.

A Timed Exit

For trading systems that hold only a short period - perhaps a few days - the timed exit may be valuable. This stop exits after 5 days.

```
ApplyStop( stopTypeNBar, stopModeBars, 5 );
```

A Profit Target

Also for systems that hold only a short period, exit when the trade has a profit. This stop exits when it can book a 5 percent profit.

```
ApplyStop( stopTypeProfit, stopModePercent, 5 );
```

A Trailing Stop

Trailing stops work well for positions that are held a longer time. They follow along behind a rising price - rising when the price goes up, but never declining. If the price declines enough to cross through the stop price, the position is sold. This stop, called a chandelier stop, trails at 3 times the Average True Range below the price.

```
ApplyStop(stopTypeTrailing, stopModePoint, 3*ATR(14), True, True );
```

A Maximum Loss Stop

The maximum loss stop is the last resort. Well designed systems seldom have positions exited because the price hit a maximum loss stop. This is a 15 percent maximum loss stop.

```
ApplyStop( stopTypeLoss, stopModePercent, 15 );
```

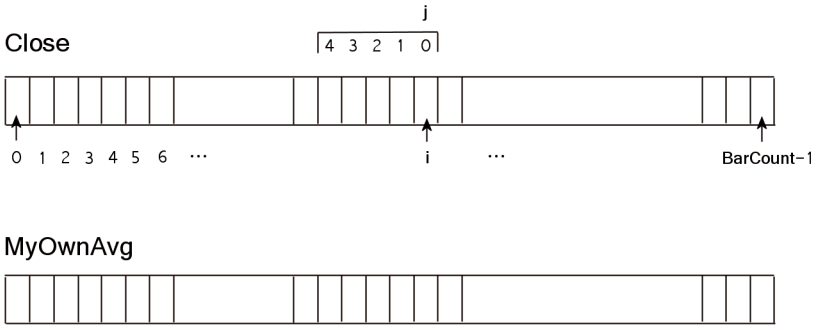
Write Looping Code

Often, you can do everything you want to with the functions built-in to AmiBroker. Occasionally, you need to examine the data bar-by-bar. A very simple example will be used to illustrate looping — the calculation of a 5 day simple moving average of the closing price for every bar of data.

The AmiBroker code is easy:

```
MABuiltIn = MA(C,5);
```

To explain the looping code, a diagram will help. You will see a small box for every day of data. There is an arrow pointing to an arbitrary index position labeled i. Index i is the day having its average computed. i starts at 0, the first bar, and goes up through BarCount-1, the last bar. There is also an arrow pointing to a group of 5 bars that will be used to compute the 5 day moving average. A separate index, j, will be used to move through these 5 bars.



The AFL code to compute a 5-day simple moving average of the Close, and store that average in MyOwnAvg, is:

```
for ( i = 0; i < 4; i++ )
{
    MyOwnAvg[i] = C[i];
}

for ( i = 4; i < BarCount; i++ )
{
    Summer = 0;

    for ( j = 0; j <= 4; j++ )
    {
        Summer = Summer + C[i-j];
    }

    MyOwnAvg[i] = Summer / 5;
}
```

The first for loop handles the first part of the series where there is not enough data to compute a 5 day average. MyOwnAvg is set to the same value as Close for those 4 cells.

The second for loop starts at the 5th cell, the one whose index is 4 (array index numbers start at 0). And it goes up to and includes BarCount-1, the end of the series. The variable i is the index of the cell being worked on at any given time.

To compute the 5 day simple moving average, the values of the current cell and the previous 4 cells are added together, then that sum is divided by 5. The third for loop does that. Note that the j variable used to add up the 5 values counts backwards; that makes the coding easier.

Good programming practices strongly discourage use of specific numbers, such as 4 and 5, in programs. And it would be impractical to have to write a new program for every different moving average length. Here is the same code written using a variable for the length of the moving average:

```
MALen = 5;

for ( i = 0; i < MALen; i++ )
{
    MyOwnAvg2[i] = C[i];
}

for ( i = MALen - 1; i < BarCount; i++ )
{
    Summer = 0;
```

```
    for ( j = 0; j < MALen; j++ )
    {
        Summer = Summer + C[i-j];
    }

    MyOwnAvg2[i] = Summer / MALen;
}
```

Write a Custom Function

AmiBroker has hundreds of built-in functions. But eventually you will come across an indicator or a function that you want to use that is not built-in. The code that follows implements the simple moving average again, this time as a function.

```
// MyOwnAvgFunction.afl
//

function MACustom ( Price, Length )
{

    for ( i = 0; i < Length; i++ )
    {
        Result[i] = Price[i];
    }

    for ( i = Length - 1; i < BarCount; i++ )
    {
        Summer = 0;

        for ( j = 0; j < Length; j++ )
        {
            Summer = Summer + Price[i-j];
        }

        Result[i] = Summer / Length;
    }

    return Result;
}

BuiltInAvg = MA( C, 5 );
MyOwnAvg = MACustom( C, 5 );

Plot( C, "C", colorBlack, styleCandle );

Plot( MyOwnAvg, "MyOwnAvg", colorRed, styleLine );
Plot( BuiltInAvg, "BuiltInAvg", colorBlue, styleLine );
```

Variable Length Parameter

The MyOwnAvgFunction has two parameters - Price and Length. Price is an array, perhaps of closing prices, perhaps of some other indicator. It is expected that there will be variation among the values in that array. Length is constant throughout the calculation of the average. The next example illustrates that the length can be an array, and its elements variable.

There must be some way of determining what values to put in the array. The ADX indicator is sometimes used to indicate the strength of a trend. High values of ADX correspond with long cycle periods, low values of ADX with short periods. This example uses the value of the ADX indicator as the length of the moving average. Readers will want to experiment with using other indicators of period or cycle length.

```
// MyOwnAvgVariableFunction.afl
//
function MACustomVariable ( Price, Length )
{
    Ignore = int( LastValue( Highest( Length ) ) );

    for ( i = 0; i < Ignore; i++ )
    {
        Result[i] = Price[i];
    }

    for ( i = Ignore; i < BarCount; i++ )
    {
        ThisLength = int( 0.5 * Length[i] );
        Summer = 0;

        for ( j = 0; j < ThisLength; j++ )
        {
            Summer = Summer + Price[i-j];
        }

        Result[i] = Summer / ThisLength;
    }
    return Result;
}

ADXVal = ADX( 14 );
//Plot(ADXVal,"ADXVal",colorBlue,styleLine|styleOwnScale);

BuiltInAvg = MA( C, 10 );
MyOwnAvg = MACustomVariable( C, ADXVal );

Plot( C, "C", colorBlack, styleCandle );

Plot( MyOwnAvg, "MyOwnAvg", colorRed, styleLine );
Plot( BuiltInAvg, "BuiltInAvg", colorBlue, styleLine );
```

This line computes how many elements at the beginning of the series should be ignored. There are more sophisticated ways to compute that, but the point is that it must be considered.

```
Ignore = int( LastValue( Highest( Length ) ) );
```

Instead of Length being constant throughout, it varies. The value needed is in the cell with the same index number as the series being averaged.

This line extracts the value of the variable length and makes it an integer, so it can be used as an index into the arrays. The 0.5 factor is a fudge factor. Since there is so much delay in the ADX indicator, its value is cut in half to make it more responsive.

```
ThisLength = int( 0.5 * Length[i] );
```

Note how the built-in average stays about the same distance from the closing prices, but the custom average with the variable lengths tightens up when the direction is changing, and lags behind when there is a trend.

Two example trading systems follow.

One is a moving average crossover system that is trend following when the length of the first moving average is shorter than the length of the second.

The other is a mean reversion system that buys when the price is far (as measured by the number of standard deviations) below the average.

Neither of these systems is particularly good. They are meant to illustrate some ideas and to be starting points for your own research.

A Trend Following Trading System

```
// TrendFollowing.afl
//
// Using end of day data,
// generating signals in the evening,
// placing trades for execution
// the next day on the open.
//
// Takes only long positions.
//
// Holds up to 3 positions.
//
// Moving average crossover to enter,
// moving average crossover to exit,
// chandelier trailing stop.
//
// Variables are set up for optimization
// and walk forward testing.
//
// Try this using a watchlist of the nine
// S&P sector exchange traded funds.
//
// This is just an example.
// Do Not trade this system.
// Do you own research.
//

OptimizerSetEngine( “cmae” );

SetTradeDelays( 1, 1, 1, 1 );
BuyPrice = Open;
SellPrice = Open;

PosQty = Optimize( “PosQty”, 2, 1, 3, 1 );
SetOption ( “maxopenpositions”, PosQty );
SetPositionSize( 100 / PosQty, spsPercentOfEquity );

MAL1 = Optimize( “MAL1”, 11, 1, 51, 2 );
MAL2 = Optimize( “MAL2”, 30, 2, 52, 2 );

MA1 = MA( C, MAL1 );
MA2 = MA( C, MAL2 );

Buy = Cross( MA1, MA2 );
Sell = Cross ( MA2, MA1 );

ATRMult = Optimize( “ATRMult”, 3.75, 1, 5, 0.25 );

ApplyStop( stopTypeTrailing, stopModePoint,
           ATRMult * ATR( 14 ), True, True );
```

A Mean Reversion Trading System

```
// MeanReversion.afl
//
// Using end of day data,
// generating signals just before the close,
// placing trades for execution
// the same day at the close.
//
// Takes only long positions.
//
// Holds up to 3 positions.
//
// Moving average crossover to enter,
// moving average crossover to exit,
// chandelier trailing stop.
//
// Variables are set up for optimization
// and walk forward testing.
//
// Try this using a watchlist of the nine
// S&P sector exchange traded funds.
//
// This is just an example.
// Do Not trade this system.
// Do you own research.
//

function zscore( price, Lookback )
{
    zs = ( price - MA( price, Lookback ) ) / StDev( price, Lookback );
    return zs;
}

OptimizerSetEngine( “cmae” );

SetTradeDelays( 0, 0, 0, 0 );
BuyPrice = Close;
SellPrice = Close;

PosQty = Optimize( “PosQty”, 1, 1, 3, 1 );
SetOption ( “maxopenpositions”, PosQty );
SetPositionSize( 100 / PosQty, spsPercentOfEquity );

ZSLength = Optimize( “ZSLength”, 34, 4, 50 , 1 );
zs = zscore( C, ZSLength );

BuyLevel = Optimize( “BuyLevel”, -2.0, -4.0, 0.0, 0.25 );

Buy = Cross( BuyLevel, zs );
Sell = Cross( zs, 0 );
```

```
HoldDays = Optimize( “HoldDays”, 9, 1, 10, 1 );
ApplyStop( stopTypeNBar, stopModeBars, HoldDays );

ProfitPercent = Optimize( “ProfitPercent”, 3, 0.5, 5, 0.25 );
ApplyStop ( stopTypeProfit, stopModePercent, ProfitPercent );
```

Schematic Diagram

Thanks to Robert Grigg, a colleague from Australia, who campaigns tirelessly — spreading the word about the importance of using proper trading system design and testing techniques.

Robert has graciously permitted use of his AmiBroker Schematic diagram. The schematic illustrates the relationships among and between the major components of AmiBroker. Although it is labeled unofficial, Tomasz Janeczko has complemented its accuracy.